Path Planning and Collision Avoidance Algorithms for Small RPAS

Juliana Maria Medeiros Alves
juliana.alves@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

June 2017

Abstract

The development of Remotely Piloted Aircraft Systems (RPAS) for civil applications has been rapidly growing over the past years. This work presents a solution for the generation of optimal trajectories for RPAS subject to manoeuvrability and collision avoidance constraints. To achieve this task a two-layered approach is proposed. In the first stage, classical path planning techniques are implemented to generate safe and flyable paths in a known static environment. The A* algorithm and Ant Colony Optimization (ACO) are used to find an optimal sequence of waypoints in a discrete environment. To ensure that the path is flyable and complies with curvature constraints, an optimization of Rational Bezier curves is implemented. The second stage is developed for real-time implementation and potential fields methods are used to replan the initial path when new obstacles are detected. For the global path planning stage the best results were found to be provided by using ACO to optimize waypoint order, A* to connect the waypoints and rational Bezier curves with constraint restriction. The Potential Fields method is computationally inexpensive proving to be a feasible solution for real-time implementation. It is shown that the algorithms perform reasonably well in several scenarios.

Keywords: RPAS, Obstacle Detection, Optimal Trajectory, Heuristic Search, Ant Colony Optimization, Potential Fields, Bezier curves.

1. Introduction

Safety remains the most important issue in the aviation field and Remotely Piloted Aircraft Systems (RPAS) can present a hazard to other aircraft, people and property. Current civil applications for RPAS include infrastructures and traffic monitoring, search and rescue operations among many others. These tasks require that the Remotely Piloted Aircraft (RPA) can autonomously go through specific waypoints while avoiding collisions on the way. Path planning with obstacle avoidance is a fundamental aspect of autonomous vehicles operations. To this day, several solutions have been developed to tackle this problem. The proposed methods so far are divided into two main categories: global and local path planning. Global path planning requires a known static environment and is generally performed offline before the mission begins. Local path planning methods are implemented during mission execution and are responsible for the replaning of the original path when new obstacles are detected.

Graph search algorithms are one of the most popular methods used in robot path planning. These methods are heavily based on the Dijkstra's algorithm [1] where starting at one vertex a graph is searched by exploring adjacent nodes until the goal state is reached, with the intent of finding the optimal path. In [2] a variation of the A* algorithm is proposed for path planning of fixed-wing UAVs in 3D environments, providing a feasible solution for offline path planning with turning and climbing angles constraints. Rapidly Exploring Random Tree (RRT) is a popular search algorithm when dealing with high dimensional spaces. In [3] a greedy version of closed-loop RRT is used to plan the collision avoidance path. The collision with aircrafts is predicted based on the RPAS current flight route and the aircrafts ADS-B data. The Ant Colony Optimization algorithm has also been applied to the UAV global path planning problem [4] [5]. The solutions however are only applied to 2D environments, considering a constant flying height, which is not suitable for many applications of flying vehicles. The Artificial Potential Field methods are an approach inspired by physical potential fields. These methods are generally used for reactive collision avoidance systems [6] and are a good solution for online implementation. In [7] this approach was applied to formation flights. Velocity Obstacles are another approach for local path planning. This

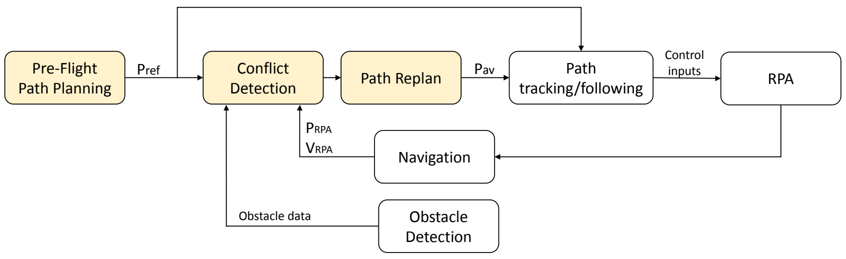


Figure 1: System architeture.

method was initially developed for ground vehicles but have since been applied to UAVs[8] [9] [10]. The method also allows cooperative maneuvers[11].

This work presents a two stage path planning architecture. In the first stage the global planning module, which assumes a known static environment, determines a collision free path from a given start to goal configurations. This path is given as a reference for the mission execution stage and as new threats are detected by the on-board sensors, the local planning module must replan the path to avoid these new obstacles.

2. Path Planning framework

Figure 1 illustrates the proposed framework for the path planning system and its integration with the other system modules.

The navigation module is responsible for the estimation of the RPA state, which comprises its position and velocity.

The obstacle detection module contains the sensors and algorithms necessary to detect and estimate the obstacles state. In this work, the type of sensors used will not be specified, but it is assumed that there is a working method of sensor fusion to obtain the necessary information about the environment. For the purpose of collision avoidance, a safety volume is defined around the obstacles. Due to its simplicity and ability to encompass a wide variety of obstacle types a cylindrical model is used to represent obstacles.

The pre-flight path planning module is used offline to find an optimal path. During mission execution the planned path P_{ref}, is given as a reference to the path tracking or path following module that, in conjunction with low level controllers, has the task of finding the necessary control inputs for the RPA to follow the given path. If during mission execution new obstacles are detected the path replan module is activated and an avoidance segment P_{av} is planned.

3. Pre-Flight Path Planning

The fundamental problem of path planning consists of finding a sequence of actions for an agent that can take it from one location to another while avoiding any obstacles on the way.

3.1. Configuration Space

A key concept of path planning is the representation of the physical world where the RPAS will operate. The environment model includes several natural conditions such as terrain, weather and obstacles.

In this work the configuration space will be defined as a regular grid. This is a conceptually simple representation, easy to construct and by properly defining the grid resolution it is possible to find kinematically feasible paths. This representation is also convenient as an action space can be independently built and a set of common actions that can be applied to any of the states in the configuration space. When deciding on the grid size some limitations of the RPA must be considered. Looking at Fig. 2 is possible to deduce for the grid resolution,

$$\Delta x = \Delta y = \frac{R_{curv}}{\sqrt{2}}, \tag{1}$$

where R_{curv} is the minimum curvature radius.

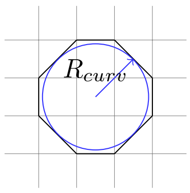


Figure 2: Grid resolution.

Fixed-wing platforms are not allowed to climb at an angle superior to the maximum climb angle, γ_{max}, hence the resolution along the vertical plane is defined according to this limit as

$$\Delta z = \Delta x \times \arctan(\gamma_{max}). \tag{2}$$

3.2. Constraints

Some of the kinematic constraints of the vehicle, minimum turning radius and maximum climb angle, were already included in the definition of the search space. Other constraints in the vehicle maneuverability can be included in the process of node expansion during the search process through the graph.

Distinct expansion rules are defined for multirotor and fixed-wing platforms. For a non-holonomic vehicle, or multirotor platforms, any of the 26 neighboring nodes in a regular grid can be reached as ilustrated in Fig. 3.

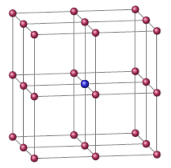


Figure 3: Expansion rules for multirotor.

Fixed-wing platforms have a forward only motion and cannot make sharp turns or climbs. To incorporate maneuverability restrictions, a set of expansion rules is defined as seen in Fig. 4.

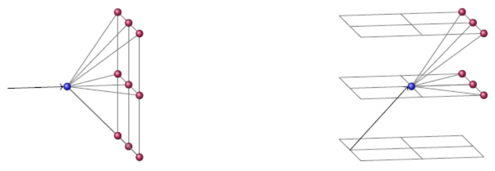


Figure 4: Expansion rules for fixed-wing.

Other constraints are the minimum safety distance. Given the relative distance between the RPA position and the obstacle center d_o, the collision avoidance constraints is

$$d_o = ||P_{RPAS} - P_{Obs}|| \geq R_s = R_{obstacle} + d_{min} \quad (3)$$

where R_s is the obstacle radius plus the minimum allowed distance between vehicle and obstacles and the minimum distance d_{min}, is defined by considering possible deviations that may occur during the execution of the path.

The mission constraints are the waypoints given to the path planner which the RPAS must visit given as

$$WPs = \{\mathbf{P_1}, \ldots, \mathbf{P_n}\}, \quad \mathbf{P_i} = [x_i, y_i, z_i]. \quad (4)$$

3.3. Cost Function

Depending on the mission objectives, different cost functions can be considered. RPAS have limited range and endurance so when planning paths a broadly used criteria is the minimum distance. Looking at the problem of limited on-board energy another important objective would be to plan for least energy cost paths.

Minimum Distance

For the minimum distance paths the cost function is simply given by the sum of Euclidean distance between all points. Considering a path $P = \{\mathbf{P_1} \ldots \mathbf{P_N}\}$ of N waypoints, the cost is given by

$$F_d = \sum_{i=1}^{N-1} \|\mathbf{P_{i+1}} - \mathbf{P_i}\|. \quad (5)$$

Minimum Energy

To formulate the energy minimization problem, an energy balance is considered. Considering a point mass model for the RPA, its motion can be analyzed using the work and energy method. The energy balance is a statement about how energy is spent when the RPA moves from point i to point j,

$$E_{i\to j} = \frac{1}{2}m(v_j^2 - v_i^2) + D\Delta s + mg\Delta h \quad (6)$$

where m is the RPA mass, v_i and v_j the vehicle airspeed in the points i and j, D the drag component, Δs the air displacment, g the gravity component and Δh the height variation between the two points. The cost function for minimum energy paths is then given by

$$F_e = \sum_{i=1}^{N-1} E_{i\to i+1}. \quad (7)$$

This simplified model can be applied to either fixed-wing platforms or multirotors. In the latter case the drag component tends to be negligible.

3.4. Path Search

To generate an optimal path for the RPAS two algorithms are considered: A* and ACO.

3.4.1 A* Algorithm

The A* algorithm[2] works by systematically searching the graph by applying the transition function and choosing the states that minimize the cost function, given by

$$f(n) = g(n) + h(n), \quad (8)$$

while keeping track of the visited nodes so that no redundant exploration occurs. In Eq. 8, $g(n)$ denotes the cost to reach the node and $h(n)$ represents the cost of getting from the node to the goal. This method is known to be complete (it always finds a solution if one exists) and optimal (the solution found is the optimal one) if the heuristic function is admissible (never overestimates the solution cost) and consistent (for every node n and every successor n' the cost of reaching the goal from n is less than the step cost from n to n' plus the cost from n' to the goal).

3.4.2 ACO

Ant Colony Optimization[4][5] is a metaheuristic method derived from the observation of real ants behaviour that use a pheromone trail to mark paths from the nest to the food source. A set of ants is placed on the departure node and following a probabilistic model they transition between nodes until all the required waypoints have been visited. Once each ant has found a solution, the pheromone trail is updated giving more emphasis to the best solution found so far. While ants construct their solution the transition probability of the k ant move from node i to node j is given by a random proportional rule,

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha . [\eta_{ij}]^\beta}{\sum [\tau_{ij}]^\alpha . [\eta_{ij}]^\beta}, \tag{9}$$

where n_{ij} is the heuristic value, τ_{ij} the pheromone value and α and β determine the influence of the pheromone trail and the heuristic information.

Pheromone trail:: the pheromone trail represents the desirability of visiting one node after the other. Generally pheromones are deposited in the edges connecting the graph nodes, however when planning in a large tri-dimensional grid it is infeasible to define each possible edge connecting nodes so in this implementation pheromones will be deposited in each node instead of the edges.

Heuristic information: to ensure that ants reach the target point, the heuristic value isd defined either as a measure of distance or a measurure of energy expenditure. The heuristic information is computed according to

$$\eta_{ij} = \left(\frac{1}{d_{ij}}\right)^c \left(\frac{1}{e_{ij}}\right)^{1-c}, \tag{10}$$

where d_{ij} represents the Euclidean distance between node i and node j and e_{ij} represents the energy spend in the transition between nodes. The energy is calculate using Eq. 6. The constant c is 1 when distance is minimized and zero if energy is minimized.

3.5. Path Smoothing

The paths obtained with A* and ACO consists of straight line segments between waypoints. This paths cannot be exactly followed by RPA with dynamic and kinematic constraints. Bezier curves are used to generate a flyable path for the RPA. Bezier curves are a type of parametric curves designed to provide a smooth path that passes exactly through the initial and final waypoints and is influenced by the other waypoints on the way, which are defined as control points. A particular case of this curves are Rational Bezier curves[12]. These curves are generate by attributing a weight to each control point, pulling or pushing the curve away from the point. They allow a better control over the curve shape. These curves are given by

$$P_R(t) = \frac{\sum_{i=0}^n B_i^n(t).w_i.P_i}{\sum_{i=0}^n B_i^n(t).w_i}, \quad t \in [0,1] \tag{11}$$

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad i \in \{0,1,\ldots,n\} \tag{12}$$

where $B_i(t)$ is the Bernstein polynomial, P_i the control points given, by A* and ACO, and w_i the curve weights. The curvature of a parametric curve $P(t)$ can be calculated as

$$\kappa(t) = \frac{|P'(t) \times P''(t)|}{|P'(t)|^3}. \tag{13}$$

The defined problem is to optimize the weights of a rational Bezier curve. The optimization problem is formulated as

$$\begin{aligned} \text{minimize} \quad & F_c && (14)\\ \text{subject to} \quad & d_o \geq R_s && (15)\\ & P_R(t) = f(w) && (16)\\ & |k| \leq k_{max} && (17)\\ & w_{min} \leq w_i \leq w_{max} && (18) \end{aligned}$$

The cost function F_c to be minimized is given either by Eq. (5) or Eq. (6). If the constraints are to be satisfied without optimizing the cost function F_c is set to zero. The constraint defined by Eq.(15) imposes a minimum distance between the robot and the obstacle. Eq.(17) ensures that, given the RPAS curvature limits, the path is flyable.

A generic constrained optimization solver, $fmincon$, provided by MATLAB is used to solve the problem. The safety distance and curvature are calculated for each point on the Bezier curve, but the number of points on the curve differ from the number of optimization variables, so a constraints lumping method is performed to attribute to each optimization variable the maximum constraint value of the closest point.

High degree curves are generally not efficient to process and situations where a solution cannot be found can easily arise when planning a long range mission on an area densely populated with obstacles. To solve this issue the curve is successively divided and each segment is optimized until the constraints are satisfied. Algorithm 1 describes the overall pre-flight path planning method.

4. Real-Time Path Planning

In this section the problem of replanning a reference path when new obstacles are detected, while considering the Rules of the Air. To develop the module

Algorithm 1: Pre-flight Path Planning.

```
Input: Constraints, cost function, reference waypoints, obstacles and departure heading and flight path angle;
Output: Optimal path from star to goal, $P_{op}$;
Find control points $CPs = \{P_1, \ldots, P_N\}$, using graph search (A*/ACO);
Set unitary weights and calculate initial Bezier curve, $B_i$, using eq. (11);
Current curve $\longleftarrow B_i$;
while Solution is not found do
    Find optimal weights, $w_o$, for the current Bezier curve;
    Current curve $\longleftarrow P_R(w_o)$;
    if constraints are satisfied then
        $B_o \longleftarrow$ Current curve;
        return $B_o$
    else
        Divide current curve
```

for real-time path planning, some safety distances are defined: first a detection radius R_d, is dependent on the types of sensors available on the platform, that is the distance at which the object is considered by the path replanning system; another zone to be defined is the action radius R_a, that is the distance from which the replanned path begins to depart from the original path given by the global planner; the safety radius R_s, define the required safety distance that must be maintained; and a collision is said to occur when the obstacle breaches the collision radius R_c. These distances are ilustarted in Fig. 5.

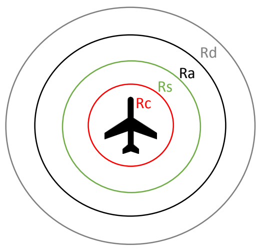


Figure 5: Safety radius defined around the obstacle.

4.1. Rules of the Air

Regarding the avoidance of collisions for manned flight, four main points are stated [13]:

1. On a head-on encounter, both aircraft should deviate to the right;
2. On a converging scenario, the aircraft with the other on its right-hand side has to give way and turn right;
3. In an overtaking event, the faster aircraft must overcome on the right hand side of the slower one;
4. An aircraft should avoid passing over, under or in front of other.

To comply with the Rules of the Air when a moving obstacle is detected, the type of enconter must be evaluated. Depending on the type of encounter different resolutions are adopted:

- When the intruder is found to be in a head-on collision course or to the right of the RPAS, the avoidance should be made by turning right;
- If the intruder is approaching from the left, and the RPA is on leveled flight, turning right will put the RPAS in front of the intruder. To avoid this scenario, the avoidance maneuver is made by turning left and going behind it;
- If the RPA is climbing, the avoidance is made by leveling the flight until the intruder is overcome;
- If the RPA is descending, the aircraft could be leveled off, but due to inertia this would be riskier than increasing the descent rate (unless the value is at its maximum).

If a static obstacle appears in the way, different paths are achieved depending on the direction of the avoidance. In this situation, there are no rules commanding the vehicle to behave a certain way, so the following strategy is adopted:

- If any side of the obstcale is blocked, the rotation is set to the opposite direction;
- Considering the line joining the RPA position and the obstacle center, if the goal point in the path is to the left the rotation is made counterclockwise and vise versa. If the point or path direction is along the line, the swirl direction can be arbitrarily chosen.

4.2. Collision Detection

To detect possible collisions, the concept of the Closest Point of Approach (CPA) is used. When a conflict with multiple intruders occurs, threats must be prioritized. As intruders will have different speeds and bearings, using the distances to the collision point is not enough, so the time to collision, t_{CPA}, is used instead. Higher priority will be given to intruders with the smallest t_{CPA} and the conflicts are resolved in a sequential manner.

[14]

4.3. Potential Fields

In this approach the obstacles and the goal position are treated as charged particles[6]. A repulsive force is attributed to the obstacles and an attractive force to the goal point. The sum of those forces is used to generate the direction of motion. The proposed fields are generated in a similar way to [14][15].

The attractive potential is responsible for directing the RPA towards the desired destination. If the objective is to direct the vehicle to a single goal waypoint, this potential function is simply given by the direction from the current position, **P**, to the desired waypoint, $\mathbf{P}_{WP}$. This potential is depicted in Fig. 6.

$$\mathbf{F}_{at} = \frac{\mathbf{P}_{WP} - \mathbf{P}}{\|\mathbf{P}_{WP} - \mathbf{P}\|} \tag{19}$$

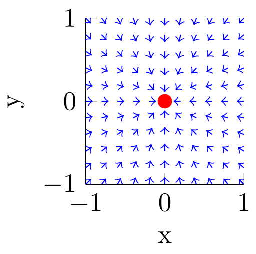


Figure 6: Attractive potential field for a waypoint.

When the mission consists on following a pre-planned path, the potential function must take into account two terms: one that brings the RPA close to the given path and other that makes the vehicle follow the path direction. To obtain the first term, the closest point on the path, $\mathbf{P}_{close}$, to the current position, **P**, is found and the direction between both is taken. The path following term is obtained from the direction from the closest point on path to the next point on the path, $\mathbf{P}_{next}$.

$$\mathbf{F}_{at} = \alpha_{PF}\frac{\mathbf{P}_{close} - \mathbf{P}}{\|\mathbf{P}_{close} - \mathbf{P}\|} + (1-\alpha_{PF})\frac{\mathbf{P}_{next} - \mathbf{P}_{close}}{\|\mathbf{P}_{next} - \mathbf{P}_{close}\|} \tag{20}$$

By selecting the values of α_{PF} more importance can be given to the path following or the path approaching direction. This potential field can be seen in Fig. 7.

The repulsive force, which keeps the vehicle away from obstacles, is given by

$$\mathbf{F}_{rep} = \begin{cases} 0, & \text{if } d_o \leq R_a \ \ or \ \ ang \geq \theta_{cut} \\ -\frac{d_o}{\|d_o\|}\left|\frac{R_t - d_o}{R_t}\right|\mathbf{S}, & \text{if } R_c \geq d_o \leq R_a \\ \infty, & \text{if } d_o \leq R_c \end{cases} \tag{21}$$

If the distance to the obstacle is greater than the action radius, the obstacle has no influence and the potential is zero. For distances inferior to the collision radius, the potential is infinite and points in the opposite direction of the vector connecting the current point to the obstacle center. Between the action and collision radius, the potential is dependent on three terms: the first one keeps the RPAS at a distance and depends on the distance to the obstacle center, the second term increases the field intensity as the vehicle gets closer to the obstacle and the last term induces a swirling motion to provide a smooth movement.

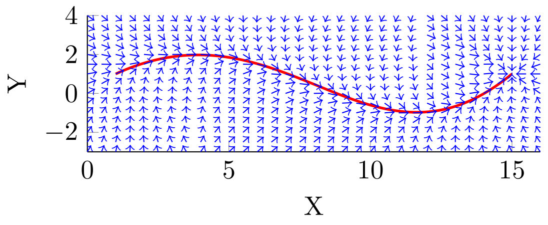


Figure 7: Attractive potential field for a path ($\alpha = 0.5$).

A cutoff angle, θ_{cut}, is defined to reduce the repulsive potential once the obstacle is overcome to avoid the RPA from being trapped around the obstacle. The angle between the desired direction of motion, D, and the relative position between the RPAS and the obstacle is given by

$$ang = \frac{acos(D \cdot d_o)}{\|D\|\|d_o\|}. \tag{22}$$

To comply with the avoidance logic the swirling direction, S, is defined according to the type of encounter. An example of the repulsive field can be seen in Fig. 8.

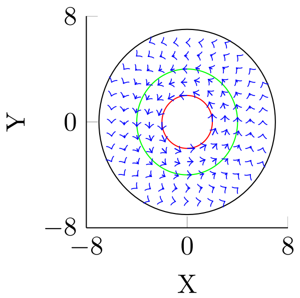


Figure 8: Repulsive potential flow.

The total potential flow force, which determines the movement direction, is given by

$$\mathbf{F}_{tot} = \mathbf{F}_{at} + \sum \mathbf{F}_{rep}. \tag{23}$$

From the total field vector the required heading and flight path angles to avoid the obstacle are obtained, from which, knowing the current direction of motion, a series of waypoints are generated until the obstacle has been cleared.

However, the combination of both the attractive and repulsive potential can lead to heading changes not feasible by the RPA, so the angle between the platform current heading and $\mathbf{F}_{tot}$ is taken. If this angle is greater than the maximum turning angle, the angle is scaled to the maximum allowable value. The same applies to the climb angle.

One issue may arise when an intruder obstructs one of the required waypoints defined during the mission planning stage. If the obstacle is static there is no way to go through the required waypoint without violating the safety distance, however if the obstacle is moving it is possible to return to the required waypoint once the collision has been avoided. To do so instead of returning to the global path once the threat is overcome, the attractive potential function for a waypoint is activated and a path that directs the RPA towards the missed waypoint is computed. When the waypoint has been passed over, or within a predefined distance, the RPA returns to the global path.

5. Results

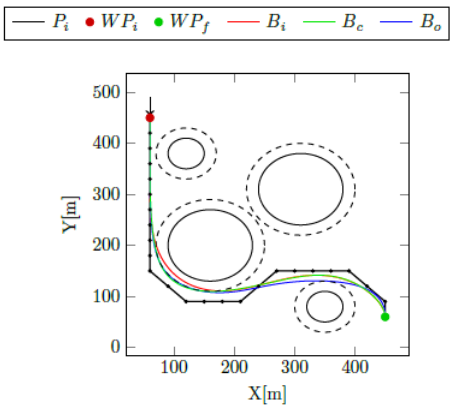
All examples are obtained with MATLAB R2016a running on a Intel Core i5 with a CPU of 2.4 GHz, 4Gb RAM and Windows 7.

5.1. Pre-Flight Path Planning

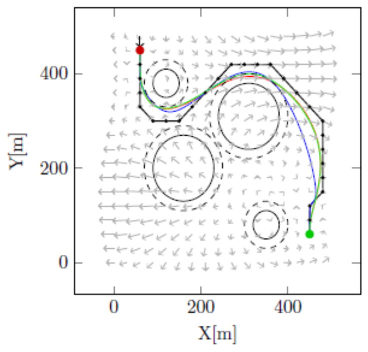
For the pre-flight path planning stage, three examples are presented: minimum distance paths and minimum energy paths in a wind environment between two waypoints and minimum distance paths between three waypoints. The following parameters are used for the ACO algorithm: $\alpha = 1$, $\beta = 2$, number of ants=10, $q_0 = 0.9$, $\tau_0 = 1$, $\rho = 0.3$, $\eta = 0.9$, Iterations=500, $\tau_{min} = 0.1$ and $\tau_{max} = 10$. The results are presented for a fixed-wing RPAS with the following parameters: $v = 16m/s$ (airspeed), $m = 2kg$ (mass), $S = 1.5m^2$ (wing span),$R_{curv} = 10m$ (minimum curvature radius) and $\theta_{max} = 30°$ (maximum climb angle).

5.1.1 Planning with 2 Waypoints

In the first case, an example of minimum distance and minimum energy path planning between two waypoints is presented. Figure 9(a) depicts the minimum distance path. In fig. 9(b) a minimum energy path is planned in the presence of a wind field. The figures also show the RPAS departure heading. The presented results are obtained with the A* algorithm, which provided better results than ACO. Three curves determined from the A* points, $\mathbf{P}_i$, are also depicted: B_i (the initial Bezier curve with unitary weights), B_c (the curve with adjusted weights to satisfy constraints) and B_o (the curve obtained from constrained optimization of the cost function).



(a)



(b)

Figure 9: (a) Minimum distance paths and (b) minimum energy paths for a fixed-wing platform given by the A* algorithm. The dashed lines represent the safety distance.

Path	Distance[m]	CPU time[s]
A^*	794	0.043
B_i	715	0.038
B_c	725	12.97
B_o	722	70.27

Table 1: 2 waypoints results for minimum distance paths.

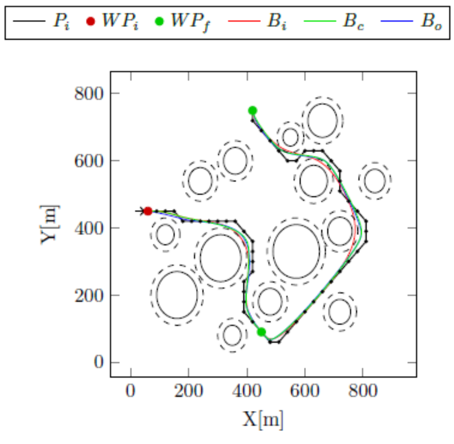
Path	Energy[kJ]	CPU time[s]
A^*	4.02	0.120
B_i	3.5	0.027
B_c	3.51	11.91
B_o	3.43	79.84

Table 2: 2 waypoints results for minimum energy paths.

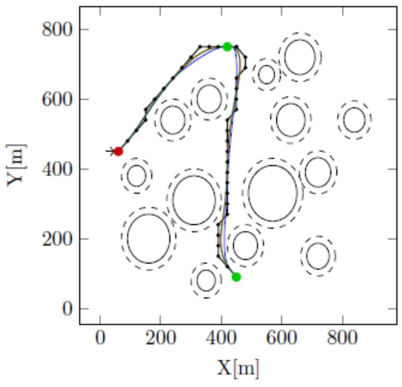
If a minimum energy path is planned without wind the same results as in Fig. 9(a) are obtained. It can be seen that the inclusion of a wind field influences the obtained path. From the results of both Tab. 1 and Tab. 2 it is seen that the optimized Bezier curve provides the path of minimum cost, but it has a longer computation time and provides little improvement over the curve that is only computed to satisfy constraints.

5.1.2 Planning with 3 Waypoints

In the following example, three waypoints are considered.



(a)



(b)

Figure 10: 3 waypoints minimum distance paths: (a) A*, (b) A* and ACO.

Path	Distance[km]	CPU time[s]
A^*	1.865	0.110
B_i	1.666	0.052
B_c	1.703	58.97
B_o	1.701	212.3

Table 3: 3 waypoints results for minimum distance paths.

In Fig. 10(a) a random waypoint order is given to the A* algorithm and in Fig. 10(b) the ACO is used

Path	Distance[km]	CPU time[s]
$A^*\&ACO$	1.266	110.15
B_i	1.191	0.038
B_c	1.191	15.6
B_o	1.175	198.56

Table 4: Results for minimum distance paths.

to find the optimal waypoint order and the A* algorithm is used to connect the waypoints. When comparing the results of using only the A* algorithm, Tab. 3, and a combination of A* and ACO, Tab. 4, it is concluded that the offline planner provides the best results when A* is used to plan between waypoints and ACO used to optimize waypoint order.

5.2. Path replan

In this section, a set of situations where a segment of the original path must be replanned to avoid new obstacles are presented. In these cases the potential fields approach was used with the following parameters: $R_{detection} = 50m$, $\alpha_{PF} = 0.5$, $\theta_{cutoff} = 30°$. It is assumed that any detected moving obstacle will maintain its course of motion.

5.2.1 Converging scenario

In this first case, the RPA is on a climbing trajectory when a moving intruder is detected. The replanned path leads to the levelling off the RPA, as depicted in Fig. 11, to avoid the obstacle. The safety distance is maintained, as seen in Tab. 5.

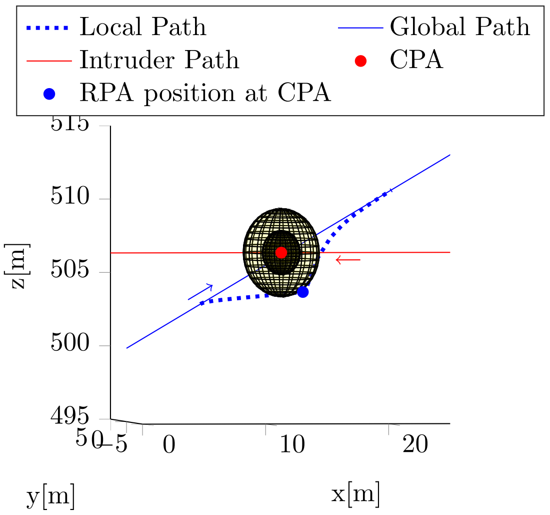


Figure 11: Converging encounter. The obstacle is depicted with the safety and collision spheres.

Minimum distance to obstacle[m]	Safety distance[m]	CPU time[s]
2.7230	3	0.019

Table 5: Climb encounter results.

5.2.2 Missed waypoint

In this example, the moving obstacle blocks one of the points of obligatory passage. As seen in Fig. 12, the RPA follows the right-hand rules to avoid the obstacle without compromising the safety distance, as verified in Tab. 6. Once the obstacle is avoided, the RPA is conducted to the missed waypoint and once the waypoint is visited it returns to the reference path.

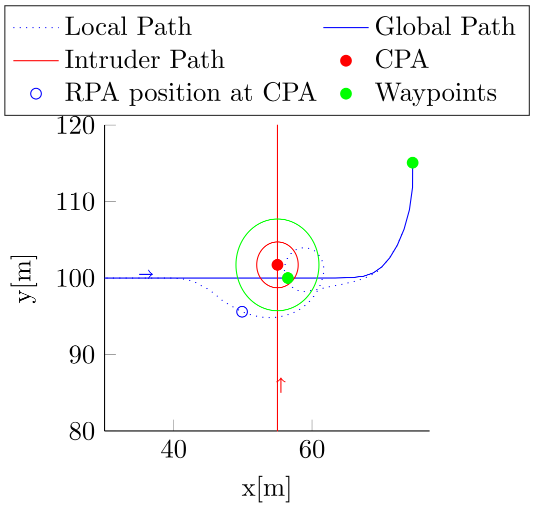


Figure 12: Converging encounter with missed waypoint. The obstacle is depicted with the safety radius(green) and collision radius(red).

Minimum distance to obstacle[m]	Safety distance[m]	CPU time[s]
7.985	6	0.015

Table 6: Missed waypoint scenario results.

The computation time of the online stage is fast enough to be suited for real-time implementation. When returning to a missed waypoint the curvature radius can prevent the RPA from exactly passing on the waypoint so an admissible radius of passage should be defined.

5.2.3 Replan with moving and static obstacles

In the final example, the RPA encounters two moving intruders, with one of them blocking a reference waypoint, and a static obstacle while following the reference path. The resulting replaned path is depicted in Fig. 13.

Obstacle	Minimum distance to obstacles[m]	Safety distance [m]
1	2.32	2
2	2.89	3
3	6.19	6

Table 7: Replan with static and dynamic obstacles results.

The RPAS successfully avoids the collisions but, as seen in Tab. 7, in the case of the second obstacle, even though no collision occurs, the safety distance is not maintained. When avoiding the static obstacle the path is replanned to maintain minimum deviation from the initial path.

6. Conclusions

This work was developed with the aim of investigating and implement methods that can provide autonomous flight capabilities to RPAS with collision avoidance capabilities.

The task was divided into a global and a local layer. For the global path planning stage the best results were found to be provided by a combination of the two algorithms, using A* and ACO to optimize waypoint order. Regarding the Bezier curves, optimizing the cost function provided the minimum cost paths, but the improvements over a curve calculated to meet only the safety and curvature constraints where not significant when considering the increase in computational time. For the online stage Potential Fields are used to generate a local trajectory when unknown obstacles are detected. The replanning of the path is made considering an uncooperative situation between the vehicles, and a sequential resolution of encounters, prioritized according to time to collision.

The global planner can resolve a series of different scenarios and new optimization criteria can be easily added to expand the range of problems to solve. The Potential Fields method is computationally inexpensive being a feasible solution for real-time implementation. The algorithm was developed to provide a path to a waypoint manager that guides the RPA through the given list, but it can easily be adapted to be integrated with the lower level control modules serving as a navigation system for a reference motion.

To tackle increasingly complex scenarios some issues need to be addressed in the future. A better integration of the vehicle dynamics is important to improve the system reliability and performance. A complete solution should consider a cooperative scenario where the vehicles exchange flight plans among each other. The type of sensors used must

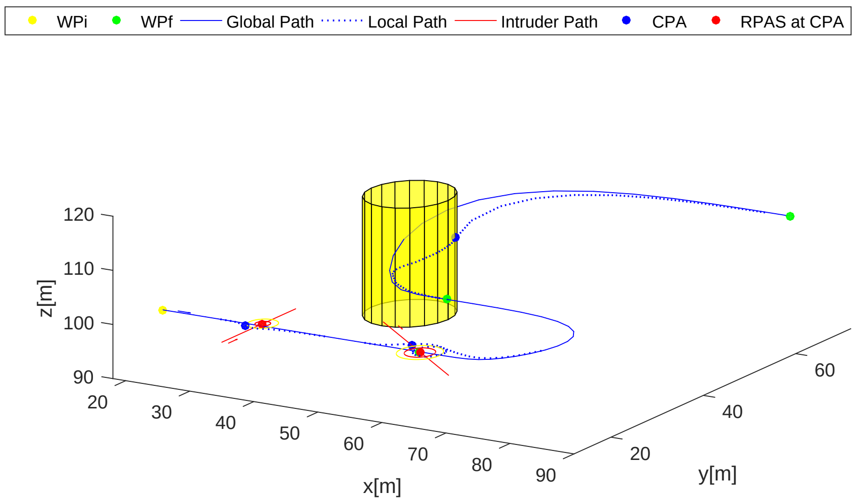


Figure 13: 3D view of path with multiple encounters.

also be taken into account as they are a crucial part of the real-world implementation, that can significantly affect the performance of the system. Different obstacle configurations must be incorporated to encompass the diversity found in the real word.

References

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

[2] Luca Filippis, G. Guglieri, and F. Quagliotti. Path planning strategies for UAVs in 3D environments. *Journal of Intelligent and Robotic Systems*, 65(1):247–264, 2012. doi:10.1007/s10846-011-9568-2.

[3] Yucong Lin and S. Saripalli. Sense and avoid for Unmanned Aerial Vehicles using ADS-B. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6402–6407, May 2015.

[4] Guanjun Ma, Haibin Duan, and Senqi Liu. Improved ant colony algorithm for global optimal trajectory planning of UAV under complex environment. *International Journal of Computer Science and Applications*, 4(3):57–68, 2007.

[5] Chao Zhang, Z. Zhen, D. Wang, and Meng Li. UAV path planning method based on ant colony optimization. In *2010 Chinese Control and Decision Conference*, pages 3790–3792, May 2010.

[6] J. Ruchti et al. RPAS collision avoidance using artificial potential fields. *Journal of Aerospace Information Systems*, 11(3):140–144, 2014.

[7] T. Paul et al. Modelling of RPAS formation flight using 3D potential field. *Simulation Modelling Practice and Theory*, 16(9):1453–1462, 2008.

[8] K. F. Culligan. Three-dimensional velocity obstacle method for UAV uncoordinated maneuvers. Master's thesis, MIT, 2006.

[9] Yazdi I. Jenie et al. Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles. *Jourbnal of Guidance, Control and Dynamics*, 38(6):1140–1146, 2015.

[10] Yazdi I. Jenie et al. Velocity obstacle method for non-cooperative autonomous collision avoidance system for UAVs. *AIAA Guidance, Navigation, and Control Conference*, 2014.

[11] Jur van den Berg et al. Reciprocal collision avoidance with acceleration-velocity obstacles. *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China*, 2011.

[12] Duncan Marsh. *Applied Geometry for Computer Graphics and CAD.* Springer-Verlag London, 2 edition.

[13] ICAO. Rules of the air. Annex 2 to the Convention on International Civil Aviation, 2005.

[14] Diogo Ruivo. Formation of unmanned vehicles with collision avoidance capabilities. Master's thesis, Instituto Superior Técnico, Nov. 2015.

[15] P. Panyakeow and M. Mesbahi. Decentralized deconfliction algorithms for unicycle UAVs. In *Proceedings of the 2010 American Control Conference*, pages 794–799, 2010.