Adaptive Dynamic Path Re-Planning RRT Algorithms with Game Theory for UAVs

A.Bertola[1], L.F.Gonzalez[2]
[1]*Politecnico di Torino – 24 C.so Duca degli Abruzzi , 10129 Torino Italy*
[2]*Queensland University of Technology (QUT) , Australian Research Centre for Aerospace Automation (ARCAA), 22 Boronia Rd. , 4007 Eagle Farm, Brisbane QLD, Australia*

Summary : The main aim of this paper is to describe an adaptive re-planning algorithm based on a RRT and Game Theory to produce an efficient collision free obstacle adaptive Mission Path Planner for Search and Rescue *(SAR)* missions. This will provide UAV autopilots and flight computers with the capability to autonomously avoid static obstacles and No Fly Zones (*NFZ*s) through dynamic adaptive path replanning. The methods and algorithms produce optimal collision free paths and can be integrated on a decision aid tool and UAV autopilots.

Keywords: Unmanned Aerial Vehicles (UAV), Search and Rescue mission (SAR), dynamic path planning,Rapidly Exploring Random Trees, Google Earth, Matlab, Aerosim/Simulink, Game Theory, Nash Equilibrium.

1. Introduction

Interest and research in civilian Unmanned Aerial Vehicles (UAVs) has been increasingly growing, especially due to the decreasing cost, weight, size and performance of sensors and processors. Researchers at the Australian Research Center for Aerospace Automation (ARCAA) are developing new software and hardware which allows UAVs to become a more suitable platform for conducting Search and Rescue (SAR) missions.SAR operations may take place in areas and times where hazardous operational conditions might impact the completion of the mission. An automatic dynamic path re-planning algorithm embedded within the Guidance System of the aircraft's autopilot represents an enabling technology for these kind of applications. During the generation of the avoiding path, the vehicle's constraints must be considered (i.e. non-holomic constraints).When a collision is predicted, a number of strategies to update the current flight plan and obtain an alternative path between two different position scan be found in the literature[3][4][5].Common strategies are the visibility graph, cell decomposition and potential field planners [3],[21],[22]. In order to assist in optimal *SAR* missions we assume that the UAV scans the area with the use of a fixed, downward-facing camera, any roll, pitch or yaw which may move the asset outside the camera Field Of View (FOV) is undesirable. Therefore it is important to develop robust software which determines a suitable flight plan according to the vehicle's flight envelope and payload requirements.

In this work a modified version of a sample–based algorithm inspired by Rapid Random Exploring Tree (*RRT*) method [1],[21] is developed. This method exploits the speed of *RRT* in exploring the space and generating collision-free paths to update the current flight plan.This ensures, within certain limits, that the aircraft can fly the desired path with a fixed relative position and relative body attitude with respect to the track under inspection.The interest shifts now to the autonomous navigation system, in order to avoid areas such as user's defined No Fly Zones (NFZ) that dynamically appear inside the mission scenario interfering with the desired path. Once an obstacle is detected, an autonomous re-plan capability is important in conducting air search missions because the environmental conditions may evolve rapidly (weather, ice, turbulence, gusts, ash clouds, etc.).One other consideration is multiple criteria for a path risk, distance therefore the developed re-planning algorithm also incorporates an optimization process based on Game-Theory to identify the Nash Equilibrium strategy as the best re-planned path possible among those considered.

The rest of the paper is organised as follows: Section 2 the background on air search, techniques for remote sensing and trajectory optimization with Game Theory. Section 3 discusses the use of *RRT* algorithms and the implementation into the UAV autopilot. Section 4 discusses the algorithm robustness and Section 5 discusses the results of test cases made in a Google Earth scenario.

2. Background

Information related to SAR planning can be obtained from different sources of documentation. We take into account the air searching patterns reported in the En Route Supplement Australia (ERSA) Manual [9]. It provides information about the planning of air search missions, the aerodrome physical characteristics, hours of operation, visual ground aids, air track services, naval aids, lighting, CTAF frequency, aerodrome operators' details and any changes applicable. The document is amended every 12 weeks. The Flamingo UAV was previously modeled in AeroSim/Simulink [14] to facilitate the development of path planning algorithms [2].The FMS consists of a PID controller where the vehicle states are evaluated to select controls. This simulator demonstrated high fidelity level and good navigation performance in previous works [2],[14]. The simulator can also be used to simulate different airframes if the values of their aerodynamic coefficients are known.

The size and logistics associated with the sensor payloads, which often involve either an optical or Airborne Laser Scanning (ALS) system, dictate that larger manned aircrafts are used for *SAR* missions [6],[7]. With the advent of larger and more stable UAV platforms and a decrease in LiDAR weight an uptake of UAV usage in the field has begun and can be expected to continue [8],[10]. Here we refer to an existing visual sensor able to couple with the actual Flight Guidance and Control System in use, such as the MP-DayView camera [15]. This is a light weight, high performance stabilized Pan Tilt Zoom Camera System. Hence, rather than autonomously scanning the ground beneath it, as an Airborne Optical Scanner does, its relative attitude to the aircraft can be remotely controlled from the HORIZON Ground Control Software or autonomously by the MP2028/2128 autopilot, according to the aircraft's current position and attitude.

In this work we make the assumption that the camera is facing downward towards the ground with a fixed attitude. Assuming the aircraft flying at constant airspeed V and altitude h, the problem reduces itself to surveying an area using a fixed sensor footprint. The sensor's Swath Width (W) is therefore given by the camera maximum Angle of View (*AOV)* according to Eqn 1 :

$$W = 2\sqrt{\frac{(h^2 + d^2)\, sin^2\left(\frac{AOV}{2}\right)}{\left(1 - 4\, sin^2\left(\frac{AOV}{2}\right)\right)}} \tag{1}$$

The Flamingo UAV has a constant cruise speed of $V = 23\ [m/s]$, hence its performance in executing a coordinate turn are limited by the minimum turning radius ρ_m given by Eqn 2:

$$\rho_m = \frac{V^2}{g tan\Phi} \tag{2}$$

where Φ is the maximum bank angle considered to enter the turn maneuver and g is the local value of gravity acceleration.

2.1 Trajectory Optimization Trade-Off Process

There are many way to optimize the re-planning process. Here we explore the possibility of evaluating the global performance of a number of generated paths and choose the best one using a trade-off process based on the Game-Theory; a well-known approach to find trade-off solutions. When the re-planning algorithm is executed, the *RRT-DR* provides a number of solutions to perform an evasive path. Through the Trade-Off process, these solutions can be ranked according to some metrics and therefore the one that better approximates the Nash Equilibrium is chosen to update the current flight plan.

2.2 Game Theory

In the approach developed in this work the RRT-based algorithm is executed a number of times, and the best path in the generated set is selected using the Nash Equilibrium solution. In game theory, *Nash equilibrium* is a solution concept of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his own strategy unilaterally [20]. If each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep their unchanged, then the current set of strategy choices and the corresponding payoffs constitute Nash equilibrium. However, Nash equilibrium does not necessarily mean the best payoff for all the players involved; in many cases, all the players might improve their payoffs if they could somehow agree on strategies different from the Nash Equilibrium. A pure Nash Equilibrium solution between a set of strategies is therefore verified if no player can do better by unilaterally changing his or her strategy. Further details on the Game Theory implementation are presented in Section 3.6.

3. Automatic Dynamic Re-planning (RRT-DR)

A modified version of the canonical *RRT* path planning algorithm is proposed and labeled as *RRT-DR*. The algorithm exploiting the speed of *RRTs* in exploring a state space and computes an optimal path to overcome obstacles for an aircraft survey track or SAR mission. In a *RRT* based planner, after the state space has been explored, the best path is chosen selecting only the full set of edges among all those in the Swath *S* of the tree that lead to the desired state x_{goal}. It may be difficult to generate the complete dense set of all samples in the state space to expand the tree. A pseudo-code for the proposed *RRT-DR* algorithm is presented in Fig. 3.The idea is to consider X_{free} only as a local set of feasible states X_{front} according to the aircraft's kinematic constraints. The nearest vertex X_{near} to grow the tree towards x_{goal} is chosen within a subset of randomized samples $X_{rand} \in X_{front}$ evaluating their metric performance in a Cost Algorithm (D).

BUILD_RRT-DR ($\boldsymbol{x}_{base}, x_{goal}, X_{obs}$)

$T.init(x_{base})$;
while $k \leq K$
T.eEvaluateFront($x_{init}(k), \rho(k)$)
$X_{rand}(k,i)$ =RANDOM_STATE$\left(X_{front}(k), R, \rho(k)\right)$
for*i = 1* **to** *R*
$X_{rand,Good}(k,j)$ =COLLISION($x_{rand}(k,i)$, X_{obs})
$X_{near}(k)$ = CONNECT$\left(X_{goal}, X_{rand,Good}(k,j)\right)$
end
T.addvertex$\left(X_{near}(k)\right)$
end
return*T*

CONNECT (x_{goal} , *T*)

$\boldsymbol{x_{near}(k)}$ =*NEAREST(x,T)*
$T.addvetex\left(X_{near}(k)\right)$

COLLISION(x_{rand}, X_{obs})

d2n = GEO_DISTANCE(x,y)
If*d2n* $\leq r$
Collision = 1;
else
Collision = 0;
end if
return *collision*

Fig. 3: RRT-DR pseudo-code.

Samples are randomized in order to promote the exploring capability of the original *RRT* avoiding it to be trapped in local minimum. The process is iterated until X_{goal} (or some other stopping condition) is reached and the set of edges connecting the chosen states X_{near} at each iteration step represents the avoiding path. These modifications are consistent with a planning problem where a flying vehicle such as a UAV is used and the shape and location of X_{obs} is not known. We implicitly assume here that the UAV is able to detect a collision with X_{obs} if it occurs within a given "look-ahead distance"(d), i.e. the sensing ability of the UAV. This parameter is very important and has to be large enough to allow a minimum turn radius maneuver able to avoid the obstacles without any collision.

3.1 Generating the offspring

We define the kinematic constraints of the vehicle by identifying a local set X_{front} of feasible new states in the space from x_{base}. Two constraints are of primary importance: the first is that the velocity is always greater than zero and the second that the minimum turning radius of the vehicle, which is algorithm of the instant airspeed, is limited in a coordinated turn:

$$\rho_{min} = \frac{V_{min}}{\omega} \tag{3}$$

Where V_{min} is the minimum airspeed at CL_{max} and ρ_{min} is the minimum turning radius for the chosen aircraft. When iterating through the process, at each k^{th}stepthis set of random samples is dense enough to preserve the *RRT* general properties, and to guide the growth of the *S* towards unexplored areas of the state space. Letting $X_{front}(k) \subset X_{free}$ be this local set of feasible states at each k^{th} step of the iteration process, the RANDOM_STATE algorithm randomly chooses an offspring of i^{th} states from $X_{front}(k)$. This is the set $X_{rand}(k,i)$ of candidate states that can be reached by the UAV. Even if the whole state space is not sampled, an offspring of randomly generated states with a algorithm of uniform probability distribution in this local boundary, the samples from x_{init} are still dense with probability equal to one. Hence, each generated x_{rand} have the same structure of the aircraft's current state, for instance see Eqn 4:

$$x_{rand}(k,i) \simeq x(t) = [lat, lon, alt, \psi] \tag{4}$$

Where lat, lon and alt are respectively the aircraft's latitude and longitude expressed in radians and alt is the altitude. $\psi$is the associated heading to each states in radians. As it is shown in Fig. 5(a), the region where the proposed new states are randomized can reasonably be reduced to an π wide arc since the heading angles related to each $X_{rand}(k,i)$are already included in their definitions; hence, having a complete 2π wide circumference for the $X_{front}(k)$ is redundant. The number $R(k)$ of randomized states at each k^{th} offspring has to be defined in advance and together with the size of $X_{front}(k,t)$ affects the performance of the search. The problem shifts then to determine which, amongst $x_{rand}(k,i)$, should be considered as the "*nearest*" state $x_{near}(k)$, and since it is desirable to bias the selection of the states pushing the search towards some predictedx_{goal}. This determines how efficient and effective is the design of waypoints in the avoidance path; preventing the vehicle to loiter and minimise the Cross Track Error (*XTE*).

3.2 Avoiding obstacles with *RRT-DR*

The *RRT-DR* re-planning algorithm designed designs a new flight plan by means of a set of waypoints once the location and the shape of X_{obs} are known. Hence, in generating the offspring of candidate states, those that are in the X_{obs} region do not need to be evaluated because they are not feasible in the space. As illustrated in Fig. 4b, at each iteration step a Collision Detection method discards those configurations that violate the global constraints in advance. The COLLISION algorithm is used to detect and predict collisions

with X_{obs} and filter the generated offspring at a higher level of control, progressively filling two matrices at each k^{th}iteration steps. Letting j be the number of feasible configurations and (i) the indices for the R-elements in $X_{rand}(k,i)$ it is possible to write Eqn 5 :

$$X_{rand,Good}(k,j) \cup X_{rand,Bad}(k,(i-j)) = X_{rand}(k,i) \quad (5)$$

The CONNECT algorithm shown in Fig. 5 replaces the EXTEND algorithm in Fig. 4 to embed this Collision Detection method before the NEAREST algorithm as it is used to evaluate the performance of $X_{rand,Good}(k,j)$. The meanings of the terms are better explained graphically in Fig. 4.

The number of randomized points (R)in the offspring $X_{front}(k)$ has to be chosen accurately. This needs to be large enough to achieve both a rapid convergence and high efficiency in explore new regions.

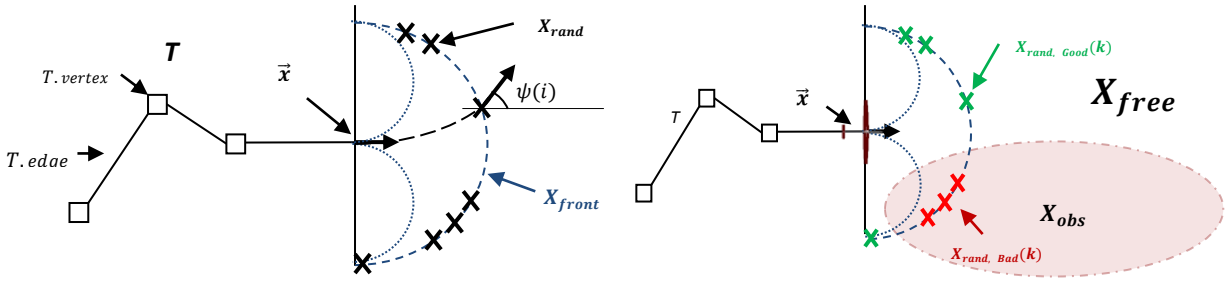


Fig. 4 : (a) First evaluation step in RRT-DR for x_{near}, X_{rand} ,(b)separation between feasible and unfeasible candidates.

3.3 Biasing the search towards the goal

At the same time, if too many of these possible configurations are generated in each offspring, then the planner may start to behave like a randomized potential field planner, trapped in a local minimum [23]. The random condition is implemented as the RANDOM_STATE algorithm. This task is undertaken by making use of a random number generator algorithm as a sampling scheme that draws *R* -states from a non-uniform probability density algorithm. The modifications to the basic *RRT* algorithm require to constantly biasing the search of new vertexes toward the desired end configurationx_{goal}. The CONNECT algorithm also embeds the bias technique allowing a faster convergence, but the algorithm's "aggressivety" is increased, leading in its limit to a straight line path between X_{init} and X_{goal} , which is a local minimum. It is therefore more appropriate to evaluate the performance of each $x_{rand}(k,i)$ and sort them using a cost evaluation method that allows us to avoid this unwanted behaviour. Once a hierarchy is established, only the best one is select as the $x_{new}(k)$ to become a new vertex in the tree. The weighs for the performance parameters that characterize x_{rand} are therefore of primary importance in determining the behaviour of the searching algorithm. The process should explore new state space regions to avoid the obstacle, but also recover the original path once X_{obs} has been avoided by trying to minimize the Cross Track Error (*XTE*) at each step. A *Cost function*$D(i,k)$ defined in Eqn 6 is used to identify and discard unfeasible states. This is a weighted metric distance between each random state and the desired end configuration x_{goal}:

$$D(k,j) = \boldsymbol{K_1} XTE(k,j) + \boldsymbol{K_2} D2T(k,j) \quad (6)$$

Where j indicates the random states in the k^{th} offspring, *XTE(k,j)* is the contribution to the current Cross track Error *XTE* and *D2T* is the distance from each x_{rand} to the desired goal state. The meanings of the terms are better explained graphically in Fig. 5.

The Cost algorithm(D) allows us to set penalties for the proposed states through the weighting factors $K_1$and K_2. Increasing the probabilities for these two values leads to both recover the original track and push the search towards the desired state after the obstacle has been avoided. This particular behavior can be

better achieved by introducing a third performance term *ANG(k,j)* defined as the angular difference between the proposed bearings $(\psi_{rand}(k,j))$ and the desired heading to the current destination WP_{cur}. Therefore, Eqn 6 explicits the cost algorithm:

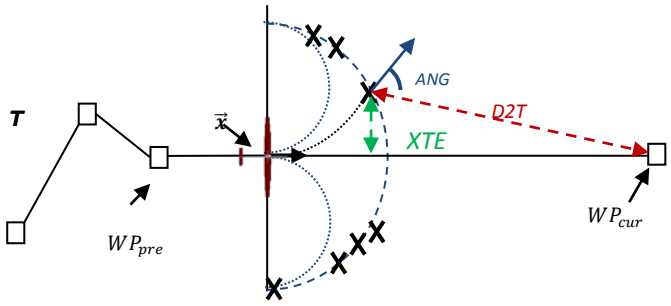


Fig. 5 : XTE, D2T and ANG displayed to evaluate the performance of $X_{rand}(k,j)$.

$$D(k,j) = \boldsymbol{K_1}\, XTE(k,j) + \boldsymbol{K_2} D2T(k,j) + \boldsymbol{K_3} ANG(k,j) \tag{7}$$

Minimising D yields the nearest state $x_{near}(k)$ to $x_{init}(k)$ at the k^{th} iteration and the nearest state is set as the new vertex, i.e. starting point to grow the tree in the next iteration:

$$x_{init}(k+1) = x_{near}(k) \tag{8}$$

The optimal settings for the weighting factors K_1, K_2 and K_3 are discussed in Section 4 where the results achieved from an extensive non-dimensional analysis over a wide variety of cases is presented. Generally speaking setting $K_2 \gg K_1$ by at least of one order of magnitude, pushes the growth towards the destination and also forces the searching process to regions that allow the obstacle to be avoided. The weight $K_3$is important for a faster recovery to the original path once the obstacle has been avoided and should be set to the same order of magnitude as XTE. This value affects the performance in more difficult scenarios were a number of obstacles located in different region are considered (see section 4). The exploration performance of the algorithm is hence determined by these three parameters which influence the speed of convergence to x_{goal} as well as the efficiency and the effectiveness of the final path.

3.4 Adaptive turn radius

The amount of time and space available to avoid the obstacle and obtain a smoother path can be better exploited by having the system able to dynamically adapt the turn radius ρ according to the vehicle's current position relative to WP_{cur} and X_{obs}. We set the value of ρ as a variable in the interval: $\rho = [\rho_m \ \infty]$: where ρ_m it is the minimum maneuver radius at V_{cruise}.If the aircraft and X_{obs} are far from colliding then a larger step can be taken, which will ensure that the fittest configuration will be the closest tox_{goal} inside$X_{front}(k)$, which leads to a straight line trajectory segment. Conversely, ρ has to become smaller as the obstacle is closer, getting close to the minimum turning radius ρ_m. Moreover, ρ needs to be proportional to the goal proximity to gain accuracy in manoeuvres when approaching WP_{cur}.Therefore the aircraft will maintain the same heading if the path is safe, otherwise a new heading is selected to overcome the obstacle with minimum off-track distance. A number of approaches and test cases are analysed in Section 5. Results demonstrate that this caution reduces the number of RRT-DR's vertexes required and that the original path can be better recovered the most of the time. Fig. 6 shows an example from the best method found in which ρ expands proportionally to the percentage in proximity from the obstacle.

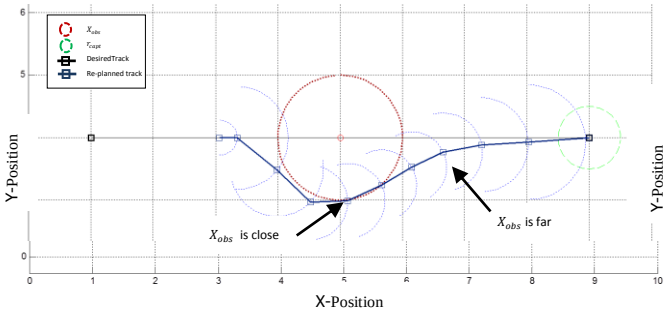


Fig. 6 : Examples of path re-planned with dynamic adaptive turning radius based on distance from X_{obs}. Notice that the turning radius gets smaller when the vehicle approaches the obstacle.

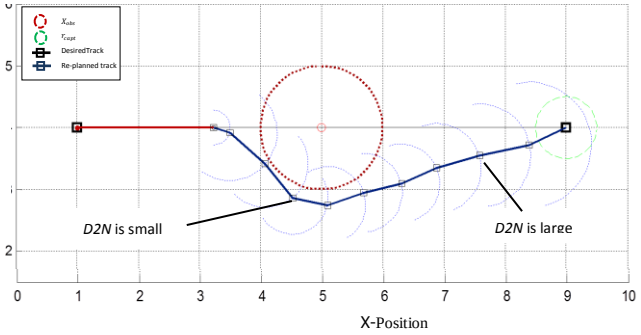


Fig. 7 : The UAV turn radius is progressively increased as the distance from the NFZ arises.

Assuming the vehicle to be equipped with a sensor for obstacle detection, where d is the device's detection range, $D2N(k)$ the proximity to X_{obs} and r the minimum clearance to be kept, we can set a linear proportional law to adapt the turning radius. This yields a much more straight line as the distance from the obstacle goes to infinite.

$$\rho = \rho_{min} + \left(\frac{D2N(k)}{r}\right)\rho_{min} \tag{9}$$

This version of the algorithm is labelled as *RRT-Dyn* and Fig. 7 shows an example of its application. However, even if the quality of the path is enhanced, without any proper setting of the weights (K_1, K_2, K_3) to penalize the choice of $x_{rand}(k)$, the ideal path will not be recovered efficiently. After extensive tests [24], a good combination for these weighting factors to evaluate the cost algorithm assigned to the *RRT-Wgh* version is:*K1* = 7, *K2* = 10 and *K3* = 4. The results of 1000 consecutive executions for each algorithm's version are evaluated in several non-dimensional scenarios to achieve a statistical value and predict their behaviour. In the presence of convex shaped obstacles, the algorithm may get trapped because X_{front} is limited in the interval $[0\ \pi]$ and the obstacle proximity to the aircraft is evaluated from one single point at time.

3.5 Dynamic Replanning in 3D

The same procedure to sample the state space looking for new feasible waypoints used in the planar version of the *RRT-DR* can be naturally extended to a 3 dimensional environment with the *RRT-3D*. In this version of the algorithm a second set of pi-wide arcs can be generated centred in the aircraft current position and intersecting $X_{front}(k)$ at each $x_{rand}(k,i)$ as it is shown in Fig.8. The same RANDOM function is used to generate the new sets of feasible waypoints on this second arc and considering a variation in height for the flight path.

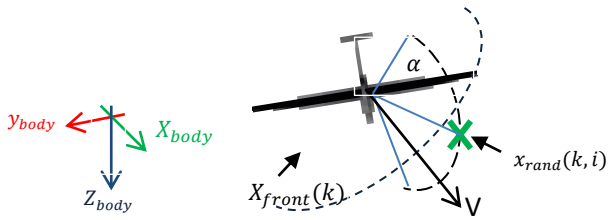


Fig 8 : Scheme of the RRT-3D principle for dynamic replanning in 3D scenarios.

The throttle limitation, the available power at the given altitude, and many other constraints are important to achieve a consistent 3D re-planning method. In this work the bounds on the maximum and minimum flight path are considered for the aircraft flying at its cruise speed are bounded by maximum and a minimum pitch angle value : $-4\ [deg] < \alpha < +20\ [deg]$.

3.6 Application of the Game Theory to Dynamic re-planning

In Game Theory the problem of reaching the next waypoint (or close enough to it) while avoiding obstacles can be formulated as a symmetric N-Player non-cooperative game. Each player can choose its own strategy to play the game in a variable dimension strategy set: $S_i = (s_{i,1}, s_{i,2}, \dots, s_{i,M})$. When a strategy is chosen by a player, no other player can chose it and each player is therefore characterized by the strategy it chooses. A payoff algorithm f_i shown in Eqn 10 determines the score for each player according to the performance parameters listed in Table 1.

$$f_i = rank(Player(q)) = \begin{cases} \max(performance(i)) \rightarrow min_{rank} \\ \min(performance(i)) \rightarrow max_{rank} \end{cases} \quad (10)$$

Where: $Player(q).Performance(i) = [L, R, C]$

Track Length (L) : $\mathbf{L} = \sum_{\mathbf{i}}^{\mathbf{N}} \mathbf{l_i}$	Sum of all the edges's lengths l_i from x_{base} to x_{goal}.
Track Risk (R) : $\boldsymbol{R} = \lvert\overline{\boldsymbol{XTE}}\rvert$	Mean value of the Cross Track Error at each step
Track Curvature (C) : $\boldsymbol{C} = \boldsymbol{mean}\left(\frac{y''}{(1+y'^2)^{\frac{3}{2}}}\right)$	Mean value of the interpolated track curvature.

Table 1: Meanings of payoff parameters used.

The Track Length ($\mathbf{L}$) and the Track Risk ($\mathbf{R}$) are bounded together by a relationship of inverse proportionality, while Track Curvature ($\mathbf{C}$) is completely independent. Ideally, this will lead to the Nash Equilibrium solution because the payoff assigned to each strategy is based on the performances achieved by the others. The Nash Equilibrium solutionis identified as the most balanced path between the performance parameters, i.e. the player gets the same ranking for at least two of L, R or C. If this is the case, then the players gains one additional point. The highest score reached at the end of the process identifies the winner.

3.7 Flight Director with *RRT-DR* implementation

The *FlightDirecto*r has the Dynamic Replanning Algorithm (RRT-DR) embedded in order to enable both collision detection and path re-planning. Other than navigate through waypoints by adjusting the aircraft's heading, this new block allows to update the current flight plan if needed, but the Guidance law to navigate through the waypoints have not been changed and are still based on the Cross Track Error value. While the simulation of a given mission is running, the users can dynamically define new No Fly Zones *(NFZs)* within the current scenario through a Google Earth GUI. As soon as their relative file containing their coordinates (*.kml* file) is saved, the *Collision Detection* algorithm inside the *FlightDirector* automatically acknowledge its presence and measures the aircraft's proximity ($D2T$) to it, increased by a certain *"look-ahead distance"* (d). Fig. 10 illustrates the overall concept where: $u = [lat, lon, alt, \psi]$ is the aircraft state vector. If (d) becomes smaller than a chosen thresholdr_c, defined as the desired clearance from the center of X_{obs}, the *FlightDirector* algorithm automatically runs the re-planning algorithm to build a collision free path. Considering that obstacles are dynamically evolving within the scenario, the look-ahead distance *(d)* can also be seen as a limitation to the minimum notice needed about the obstacle's location and ensure the effectiveness of the evasive action.

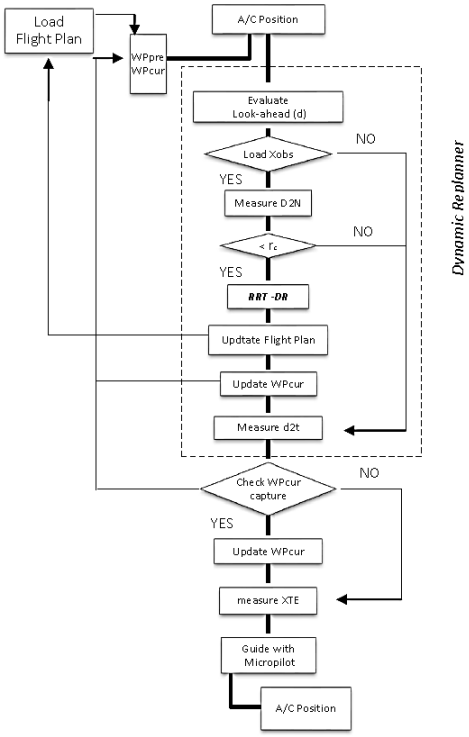


Fig. 9 : The FlightDirector flow chart

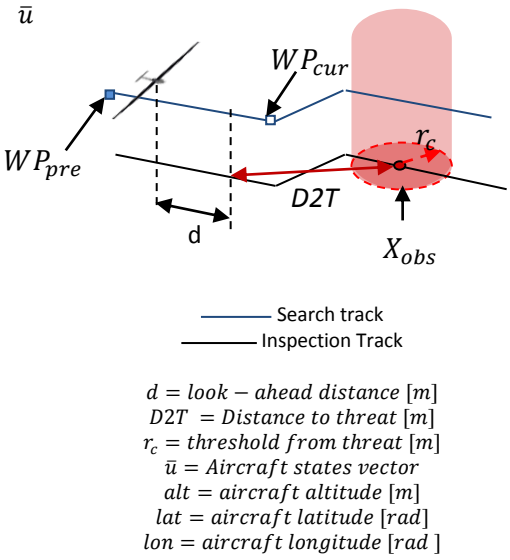


Fig. 10: Parameters considered in the dynamic re-planning algorithm RRT-DR at time (t).

4. Model Test Cases: Algorithm Robustness

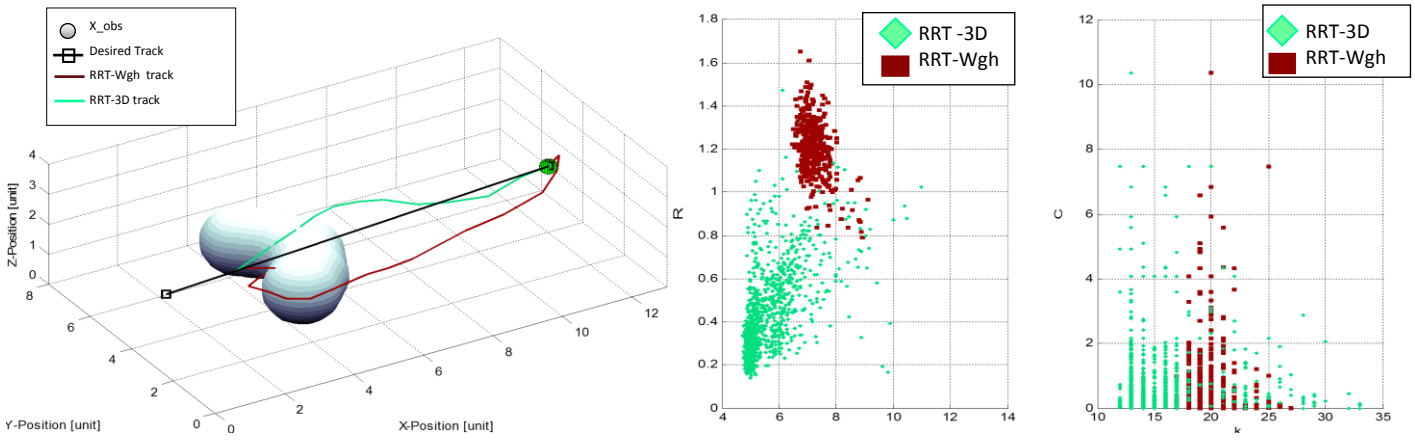
4.1 Metrics

XTE, *D2N* and *D2T* parameters are used to identify the trends of the best state chosen by each offspring $\boldsymbol{X}_{rand}$ as described in Section 3.3. The *XTE* measures how far the vehicle is from the desired track. The objective is to minimize *XTE* as *D2N* reach its minimum and force it to be as small as possible as *D2N* progressively increases. If no weights are adopted in the cost function, and a constant value of turning radius is used and the control on the heading angle is neglected, the *RRT* is indeed able to rapidly explore the state space evaluating a path to effectively avoid a 3D obstacle, but despite that it is also remarkably inefficient for the purpose maximizing track coverage. Several test cases were performed [24] but only an evasive maze test in 2D and wall test in 3D are described here.

4.2 Dynamic re-planning in 3 Dimensions (*RRT-3D*)

The developed algorithm was compared to the existing *RRT* algorithm during the first concept phase of the project to understand how it could be modified to fit a UAV path planning problem and achieve the particular behavior described in the previous sections. The test cases presented in this work compare different versions of the dynamic *RRT* algorithm. There are some situations in which, due to the particular shape of the obstacle, a planar path might require the aircraft to fly very far away from the desired inspection track in order to be able to avoid the obstacle. Aiming to maximize the coverage, one solution is to allow the re-planning to evaluate configurations in a 3D state space. In the *RRT-3D* version of the algorithm, the *RANDOM* function is used to generate random new states offspring in a 3D space. However, as it was discussed in Section 3,we must consider the limitations given by the aerodynamic constraints of the vehicle. An example of the path evaluated for a convex wall by the *RRT-3D* compared to the *RRT-Wgh* is shown in Fig. 11 (a), whilst the diagrams in Fig. 11 (b) and (c) compare the values of XTE, L, C and k parameters for 100 different paths generated with both version of the algorithm.

It is seen that a non- planar avoiding path effectively improve the efficacy of the evasive manoeuvre because paths are shorter and closer to the original track than those generated by the *RRT-Wgh*. The aircraft avoids the obstacle by flying over it. In this work we chose use the *RRT-3D* version of the algorithm and the Game theory to optimize the re-planning process for more realistic SAR scenarios.

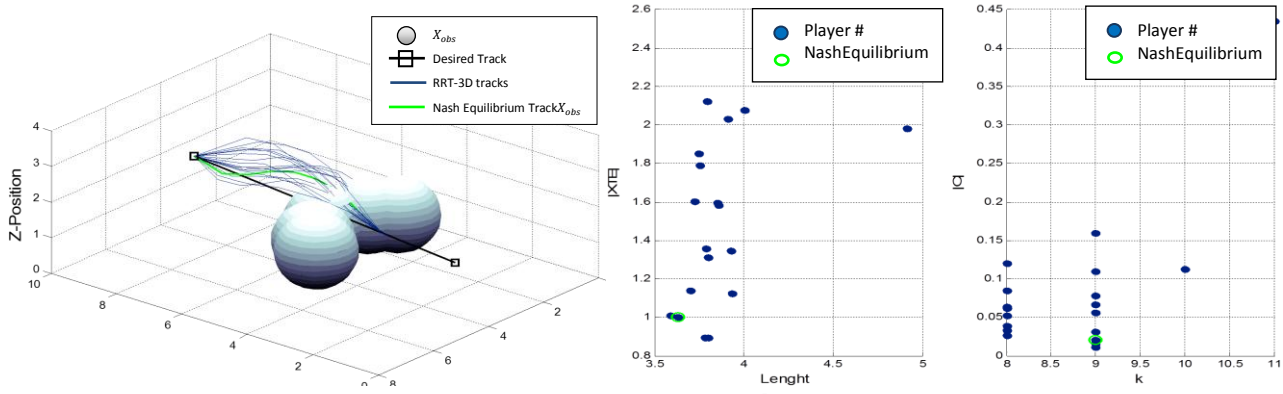


(a) *(b)* *(c)*

Fig. 11 : Comparison on planar and non-planar algorithms.
(a) Example of well re-planned flight plan. - (b) L-R scatter plot - (c) $|\bar{C}|$ VS (k) diagram.

4.3 Game Theory implementation and model test cases

It was seen in the previous sections that the strategy to be adopted to efficiently and effectively compute a path that will not lead to collisions, depends on the particular shape of the obstacle ($\boldsymbol{X_{obs}}$) and cannot be determined a priori. Thus a trade-off process based on Game Theory can be exploited to select a path amongst those obtained by each algorithm. This identifies the path in the Nash equilibrium or produces the best trade-off solution defined by performance metrics. The paths evaluated by 20 Players trying to avoid a Convex Wall are shown in Fig. 12 (a). The Nash Equilibrium solution is highlighted in bright-green colour. Fig. 12 (b) and (c) compares these paths and indicates the Nash Equilibrium solution, which is the most balanced one in relationship to the considered parameters. Table 2 is used to sort the paths according to their payoffs and identify the Nash Equilibrium solution.



(a) *(b)* *(c)*

Fig. 12 : 20 Players game for C-Wall scenario with RRT-3D.(a) Players paths to avoid the convex wall, Nash Equilibrium is in bright green. (b) L-R scatterplot. (c) k-|C| diagram.

Player #	L	R	C	Payoff			Score
1	3,9287404	1,3457128	0,1203771	5	12	3	20
2	3,7804654	0,8946101	0,078087	13	18	7	38
3	3,933	1,1241509	0,1594442	4	15	2	21
4	3,9128494	2,0325557	0,0328659	6	4	14	24
5	3,7016547	1,1381463	0,1094237	17	14	5	36
6	3,8616686	1,5832362	0,0517568	7	10	12	29
7	3,798716	1,3125462	0,0162486	10	13	18	41
8	3,7967349	2,1213967	0,0847605	11	2	6	19
9	3,7246461	1,6022619	0,0631998	16	8	9	33
10	3,8528162	1,596012	0,112197	8	9	4	21
11	3,7562227	1,7898591	0,0387852	14	7	13	34
12	3,5879475	1,0068657	0,0559485	19	16	11	46
13	4,0027078	2,0772213	0,0263949	3	3	16	23
14	4,8205035	2,5367813	0,4347341	2	1	1	5
15	3,7500504	1,8529897	0,061758	15	6	10	31
16	3,802683	0,8920578	0,0306695	9	19	15	43
17	**3,6319882**	**1,00001**	**0,0210474**	**18**	**17**	**17**	**53**
18	4,9154162	1,9833427	0,0114129	1	5	19	25
19	3,78916	1,3558729	0,0665448	12	11	8	31

Table 2 : 20 Players game payoff matrix with RRT-3D, Nash Equilibrium in bold font.

5. Test Cases

5.1 Test Case 1: Multiple static obstacles – 3D

In this test we refer to a mission scenario where a UAV is used to investigate a long track in the coastal area near Hervey Bay, QLD, Australia. We assume that at time $t^* = 50\ s$ the ATC communicates that more than just one other vehicle is operating in the area. Multiple No Fly Zones interfering with the desired track are therefore defined dynamically during the simulation as squared-based polygons in Google Earth. Fig. 18 shows a screenshot of the desired track intersecting where: NFZ-1 and NFZ-2 intersect the southern and the northern leg respectively and the distance between their boundaries is about $500\ m$. NFZ-3 instead intersects both legs. When the first dynamic replan occurs at time $t_1 = 94.82s$, the algorithm demonstrates the capability to find a safe path passing in the narrow passage between NFZ-1 and NFZ-2.

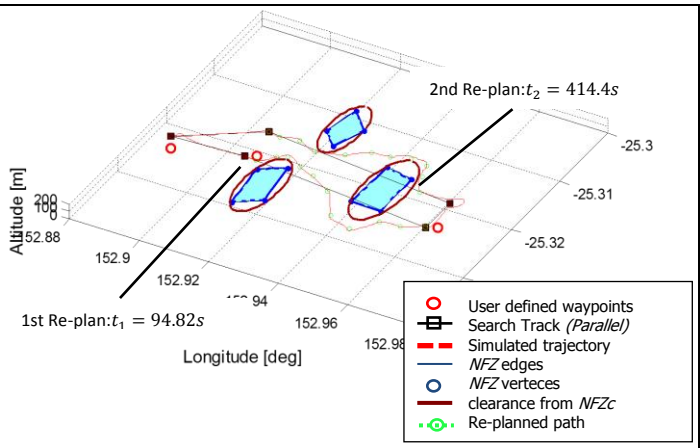


Fig. 13 : Searching Track updated after two re-planning, the UAV Trajectory is displayed.

Fig. 14 : Screenshot from Google Earth GUI.

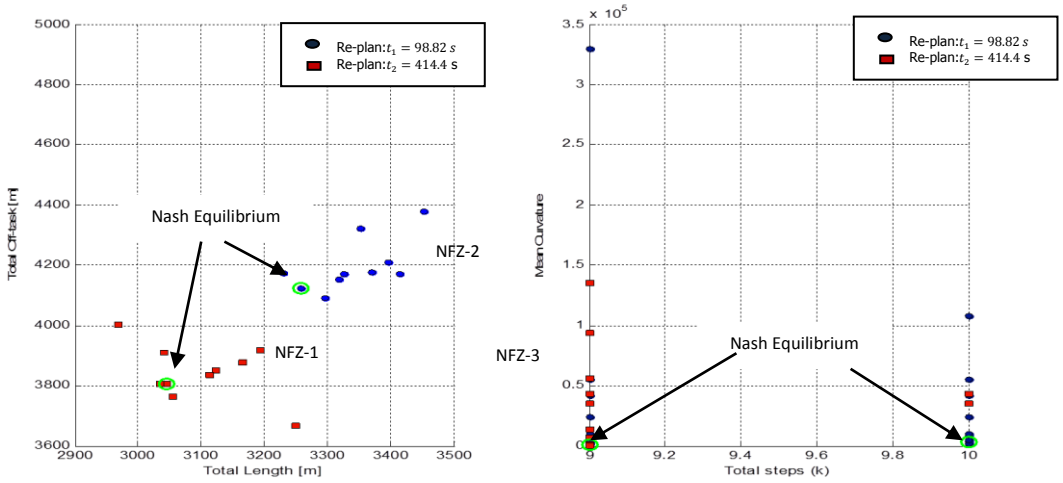
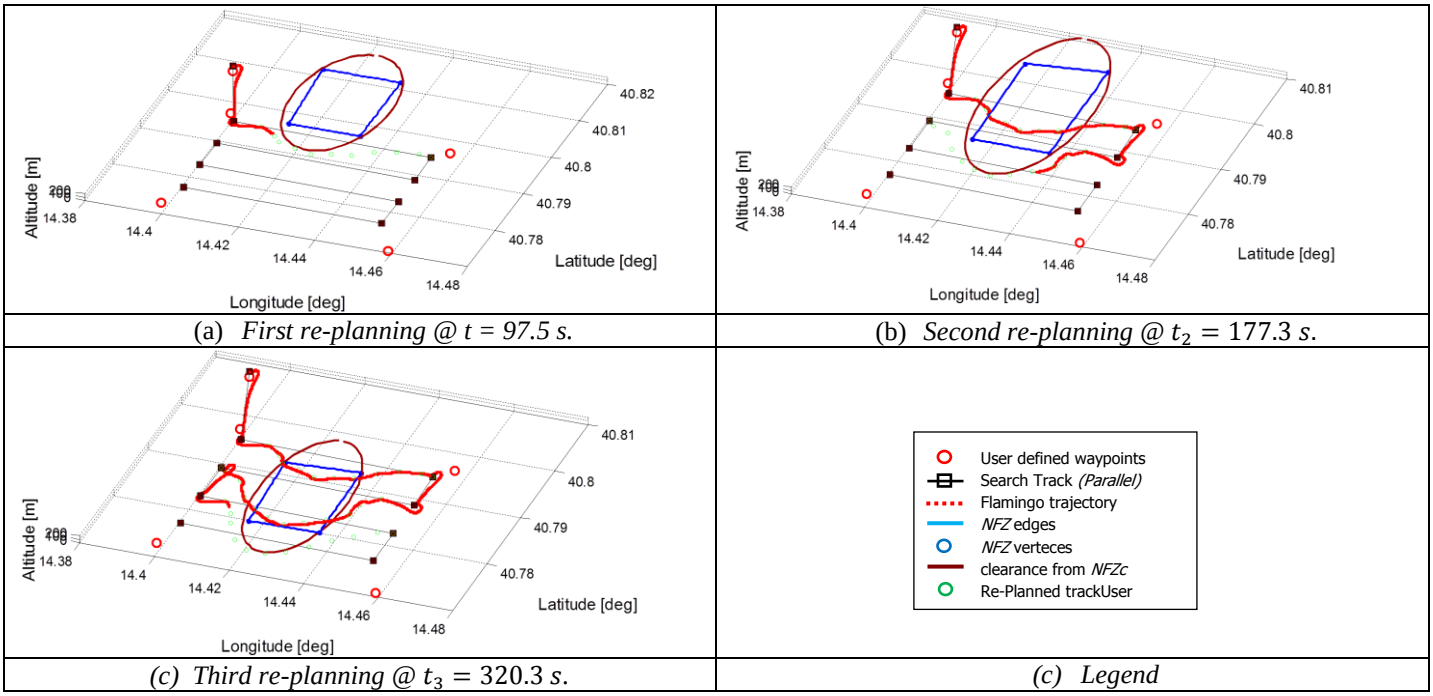


Fig. 15: Scatter plots from the Game Theory Process.

Moreover, it autonomously switches between the No Fly Zones considering only the closest to the aircraft position. The values assigned to the weighting factors in the cost algorithm D are: $K_1 = 8.48, K_2 = 8.29, K_3 = 4.2, K_4 = 1.32$: they determine the quality of the avoiding path and its capability to recover the original flight plan. The algorithm can replan the trajectory more than once during the mission, allowing new NFZ to be dynamically updated on the map. The set of alternative paths generated is filtered by the Game Theory optimization process: the scatterplots shown in Fig. 14 result to be sparse because of the small number of candidates set for each offspring. The Nash Equilibrium solution, represented with a green circle, is the best trade-off among all paths simultaneously generated and corresponds to the minimum path curvature as well.

5.2 Test Case 2: Single dynamic NFZ

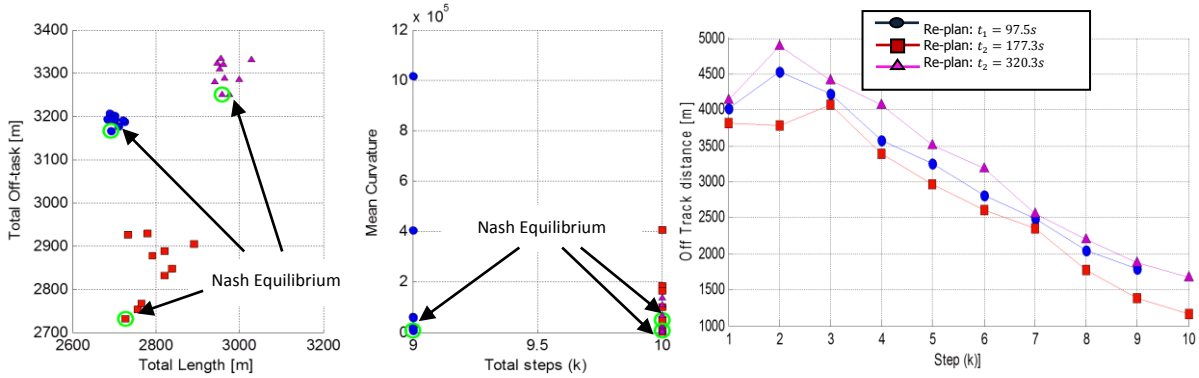
In this case we assume to face a distress situation in the surrounding area of *Mount Vesuvius*, an active volcano located about 5.5$[km]$ East of Torre del Greco in the province of Naples, Italy. A The SAR expert selects a Parallel track approach to cover this $4.5x3\ km$ wide area with a Flamingo UAV at the altitude of $h = 152.4\ m$. We assume that, a thick ash cloud of 2.3 km in diameter and 637 m in height is moving towards the desired search area, exposing the aircraft to danger. After the mission has started, the weather expert updates the position and the extension of this No Fly Zone by modeling it in Google Earth. In Fig. 16 (a),(b) and (c), we can see how the scenario is dynamically updated during the simulation and how the re-planning is executed. In Fig. 16(a) the ash cloud is uploaded at time $t^* = 50s$ before WP-2 is reached. This causes the first re-planning to be executed at $t_1 = 97.5\ s$. In Fig.16(b) the current flight plan is updated a second time after the ash cloud has moved by 1 km, the second re-planning is executed at $t_2 = 177\ s$ and a third one is executed at $t_3 = 320\ s$.
The Nash Equilibrium Path is also computed and displayed in these Figures as a green-dotted line. The RRT-3D version of the algorithm was chosen to perform the re-planning in 3D. In this test, an offspring composed of R=50 samples is considered. This ensures a more even behavior between simultaneously evaluated solutions with the same strategy, and the number of players has been reduced to 10. Therefore, the dispersion of the points in the scatterplots of Fig. 17(a), (b) result to be smaller than the previous cases. The Nash Equilibrium solution resulting after each re-planning has been highlighted with a green circle. Fig. 17(c) shows the Cross-Track error after each re-plan.



(a) *First re-planning @ t = 97.5 s.* (b) *Second re-planning @ $t_2 = 177.3$ s.*

(c) *Third re-planning @ $t_3 = 320.3$ s.* (c) *Legend*

Fig. 16 : Dynamic evolution of the scenario while replanning the current flight plan.



(a) (b) (c)

Fig. 17: Scatter plots from the Game Theory Process, (b)- XTE values of added waypoints.

6. Conclusions

A Mission Path Planning tool coupled to a modified *RRT-2D* and *RRT-3D* with Game theory Nash Equilibrium has been developed providing a rapid and effective method to generate flight plans for SAR missions. The prototype version presented in this work can be used a number of experts working together: a SAR expert chooses the most suitable path to cover a desired area, a UAV expert then monitor the flight while the ATC and weather experts interact dynamically by defining obstacles and No Fly Zones. The proposed method, based on RRT algorithms for path planning, was obtained considering a geometric constrain related to the UAV kinematics. The *RRT-3D* rapidly introduces a set of intermediate waypoints into the current flight plan to avoid obstacles and NFZs. A cost algorithm is used to effectively push the growth of the tree towards the desired goal in which the values of the weighting factors considered have been adjusted through a non-dimensional analysis. The algorithm also selects the turning radius according to the obstacle's proximity. The path optimization process is based on Game Theory which allows to better recover the original flight plan once the obstacle has been avoided and reduce the amount of track which is not covered due to the evasive action. The guidance analysis demonstrates the capability of the model to simulate the flight through the prescribed waypoints. The RRT algorithms developed for 2D and 3D

scenarios enable to consider many constraints at the same time. Their application in path planning is suitable also in dynamic collision avoidance problems which includes multi-objective optimization problems.

References

1. LaValle, S.M. and J.J. Kuffner, Jr. *Randomized kinodynamic planning. in Robotics and Automation.* Proceedings on IEEE International Conference, 1999.
2. Torta, P., *Development of a simulation model for an Air search/Air sampling UAS,* MD in Aerospace Engineering , Politecnico di Torino 2010, Torino.
3. Kavraki, L.E., *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* Transactions on Robotics and Automation, IEEE, 1996. 12(4): p. 566-580.
4. Macharet, D.G., A.A. Neto, and M. Campos. *On the generation of feasible paths for aerial robots in environments with obstacles*. IROS 2009. Proceedings on IEEE/RSJ International Conference on Intelligent Robots and Systems.
5. Wzorek, M., J. Kvarnström, and P. Doherty, *Choosing Path Replanning Strategies for Unmanned Aircraft Systems.* ICAPS , AIAA 2010, pp. 193-200.
6. Authority, A.M.S., *Air Search Observer*, Lerner Guide, A.M.S. Authority, Editor. 2004, Australian Government: Australia. p. 44.
7. Westall, P., *Evaluation of Maritime Vision Techniques for Aerial Search of Humans in Maritime Environments in Computing: Techniques and Application.* DICTA 2008.*Digital Image.* 2008.
8. *Flight Assist System. 2011;* Ergon Energy ROAMES project. Available from: http://www.arcaa.aero/research/flight-assist-system/
9. Australia, A.S., *Aeronautical Information Package (AIP)*, in *En Route Supplement Australia (ERSA).* 2011.
10. http://www.close-search-project.eu/index.php/onsarmissionswithuavs.
11. Young, B. *Silvertone Flamingo UAV - The UAV in detail 2011 26/5/2011*; Available from: http://www.flamingouav.com/flamingo.php.
12. http://www.arcaa.aero/facilities/fixed-wing-uas/.
13. http://www.arcaa.aero/research/flight-assist-system/
14. Molloy, T., *Development of a 6-degree of freedom non-linear dynamic model for guidance, control and simulation of a Remote Sensing UAV*. 2010, QUT & ARCAA: Brisbane p. 103.
15. MicroPilot, ed. *MP Camera Software User Guide*. 2009, Micropilot Support: Sturgeon Rd., Stony mountain, Canada.
16. Phillips, W.F., *Mechanics of Flight* 2004: Jhon Wiley & Sons Inc.
17. Aumann, R.J., *Game theory, in The New Palgrave Dictionary of Economics*, S.N. Durlauf and L.E. Blume, Editors. 2008, Palgrave Macmillan: Basingstoke.
18. B.Myerson, R., *Game Theory: Analysis of Conflict*. 1991: Harvard University press.
19. Osborne, M.J. and A. Rubinstein, *A course in game theory*. 1994: MIT Press.
20. Nash, J.F., *Equilibrium Points in n-Person Games*. Proceedings of the National Academy of Sciences of the United States of America, 1950. 36(1): p. 48-49.
21. P. Cheng, Z.S.S.M.L., *RRT-based trajectory design for autonomous automobiles and spacecraft*. 2001, Archives of Control Sciences. p. 11(3-4) 197-194.
22. LaValle, S.M., *Sampling-Based Motion Planning, in Planning Algorithms*. 2006, Cambridge University Press: Cambridge,U.K.
23. Jatmiko,W., K..Sekiyama, and T.Fukuda, *A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement.* Computational Intelligence Magazine, IEEE, 2007. 2(2): p. 37-51.
24. Bertola, A., *Dynamic Adaptive path Re-Planning for UAV: Theory, Algorithm and Applications.* ,MD in Aerospace Engineering. , Politecnico di Torino 2010, Torino.