RRT*-Based Algorithm for Trajectory Planning Considering Probabilistic Weather Forecasts

E. Andrés, M. Kamgarpour, M. Soler, M. Sanjurjo-Rivo, and D. González-Arribas

Abstract Convective weather and its inherent uncertainty constitute one of the major challenges in the air traffic management (ATM) system, entailing both safety hazards and economic losses. In the present work, we propose a stochastic algorithm for trajectory planning that ensures feasibility and safety of the path between two points while avoiding unsafe stormy regions. The uncertain zone to be flown is described by an ensemble of equally likely forecasts. We design a scenario-based optimal rapidly exploring random tree (SB-RRT*), and we able to dynamically allocate risk during its expansion so that a safety margin is not violated. The solution is a safe continuous trajectory that minimizes the distance covered. We present preliminary results assuming weather to be the only source of uncertainty. We consider an aircraft point-mass model at constant altitude and airspeed with manoeuvres being limited by a minimum turning radius.

1 Introduction

Weather uncertainties represent a major issue that the air traffic management (ATM) system needs to account for. In particular, areas of convective weather (also known as thunderstorms) constitute a potential safety hazard, being responsible of a quarter

E. Andrés (✉) · M. Soler · M. Sanjurjo-Rivo · D. González-Arribas
Department of Bioengineering and Aerospace Engineering, Universidad Carlos III de Madrid, Leganés, Spain
e-mail: eandres@ing.uc3m.es

M. Soler
e-mail: masolera@ing.uc3m.es

M. Sanjurjo-Rivo
e-mail: msanjurj@ing.uc3m.es

D. González-Arribas
e-mail: dangonza@ing.uc3m.es

M. Kamgarpour
Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland
e-mail: mkamgar@control.ee.ethz.ch

Air Traffic Management and Systems IV, Lecture Notes in Electrical Engineering 731,
https://doi.org/10.1007/978-981-33-4669-7_14

of the en-route delays in Europe [1]. In order to enhance ATM's safety, efficiency and capacity, path planning techniques must take into consideration the inherent stochasticity of these phenomena. The aim of this work is to develop a methodology for safe aircraft trajectory planning considering the intrinsic uncertainties of the stormy environment.

The design of avoidance paths that prevent flying through risky areas is a problem of interest that has been covered in the literature using a wide spectrum of approaches with different applications. A first approach is based on geometric procedures, see e.g., [2]. They benefit from fast computing times at the cost of usually ignoring storm evolution and their uncertainty, aircraft dynamics or trajectory optimality. A second class of methods relies on robust optimal control [3]. The optimization problem considering aircraft dynamics and uncertain thunderstorm development can be solved using nonlinear programming. However, the solution of the robust optimal control problem is quite sensitive to the choice of the required initial guess. To this end, in [3], a randomized initialization is proposed, obtaining a wide range of local optima and identifying the best solution. A third possible approach is the so-called stochastic reach-avoid [4, 5], which is based on dynamic programming. These techniques are able to find optimal trajectories in uncertain and time-varying scenarios. Nonetheless, they are often computationally prohibitive, because there is a need to discretize and explore an entire state space. Therefore, the affordable dimension of the problem is limited due to the so-coined by Bellman "Curse of Dimensionality".

In this work, we opt for an incremental sampling-based algorithm, the rapidly exploring random tree (RRT) [6]. RRT-based algorithms are able to find feasible trajectories in high dimensional problems, including system kinematics, dynamics and constraints [7]. In addition, if the vehicle is moving in a constantly changing environment, they admit online planning, meaning that once a vehicle is following the planned trajectory they can incorporate new data and replan the route in almost real-time [8]. In the literature, there are different versions of RRT that have been applied to trajectory planning of autonomous driving cars [9–11] or UAV flights [12–14].

With regard to the stochasticity of the unsafe regions, previous works on RRT were built on chance constraint approaches with both linear [15] and nonlinear [16] dynamics. Chance constraints provide the probability of being above or below a safety margin for a particular state-space configuration [17], hence, these works only checked the safety of discrete points along the trajectory. To the best of authors' knowledge, RRT-based algorithms have not considered the safety of a continuous path. Moreover, RRT techniques have not been used for aircraft flight planning in areas of uncertain weather.

To assess safety during the flight, there is a need to develop models of thunderstorm evolution. Modelling thunderstorms for flight planning purposes are a challenging task, because it is difficult to forecast their birth and evolution in timescales close to flight departure. The main reason is that the atmospheric evolution is chaotic and extremely sensitive to perturbations, so any change might lead to huge errors in a prediction. In the numerical weather prediction (NWP) framework, the main trend is to characterize weather uncertainties with an ensemble forecast [18]. An ensemble

provides a number of realizations, typically between 10 and 50, that represent the atmosphere assuming slight variations during its evolution. By analysing the ratio of realizations that predict a storm, we can estimate how risky an area is.

Our main contribution and the purpose of this work are the design of a RRT-based algorithm for aircraft flight planning in stochastic weather regions ensuring the safety of a continuous trajectory. We propose a scenario-based rapidly exploring random tree (SB-RRT*) for general nonlinear systems able to grow in an uncertain environment which is described by an ensemble of discrete realizations. Unlike previous techniques that only considered the safety of a discrete set of points, our algorithm ensures that a whole continuous trajectory is safe. As the tree is expanding, it is able to dynamically allocate risk wherever is needed in order to never exceed a global safety margin.

The paper is structured as follows: we introduce RRT algorithms in Sect. 2. In Sect. 3, we propose the SB-RRT*, an optimal RRT able to expand in stochastic and hazardous regions. A case study and simulation results are included in Sect. 4. Finally, some conclusions are drawn in Sect. 5.

2 RRT: Rapidly Exploring Random Trees

RRT algorithms are path planning tools that look for a feasible trajectory between an initial and a final state configuration. From an initial state, RRTs are expanded, iteration by iteration, driving the system towards randomly selected targets. RRTs can deal with a static or dynamic environment made up of unsafe zones to be avoided. RRT planners are able to handle several degrees of freedom with constraints [6].

2.1 RRT Algorithm

Let $X \subset \mathbb{R}^{d_x}$ be the planning space in which the aircraft will manoeuvre. The constant d_x represents the dimension of the coordinate system, generally equal to 2 or 3. Let $X_{\text{storm}} \subset X$ be the unsafe zones that must be avoided, which in this case represent thunderstorms. The complementary set $X_{\text{safe}} = X \setminus X_{\text{storm}}$ represents the safe areas.

The RRT algorithm defines an iterative process that grows a tree $T = (A, E)$, where A is the set of randomly selected nodes (also known as vertices) and E represents the collection of edges connecting pairs of nodes in A. Individual nodes and edges are denoted by a_i and e_i, respectively, with the subscript i bounded by the maximum number of iterations. A fixed position $\mathbf{x}_i \in X$ obtained randomly defines a node a_i. An edge e_i represents a continuous trajectory between two nodes a_i and a_i'. Note that not every node or edge created by the algorithm is included in A or E, some of them are rejected if there is an intersection with X_{storm}. Let $\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}} \in X$ be the initial and goal state configurations, the nodes a_{start} and a_{goal} represent these positions respectively in T.

Let $S \subset \mathbb{R}^{d_s}$, with $d_s \geq d_x$, be the state space and let $U \subset \mathbb{R}^{d_u}$ be the control space. In general, the dynamics of our system will be represented by a state vector $\mathbf{s} \in S$ that evolves according to a transition equation,

$$\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{u}),$$

with $\mathbf{u} \in U$ the control input. Assuming the described setting, the RRT grows by following the set of procedures detailed hereafter:

- *RandomSample*: this function takes a random sample $\mathbf{x}_{rnd} \in X$ from the planning domain and creates its associated node a_{rnd}.
- *NearestNode*: it returns the closest node to a_{rnd}, $a_{nearest}$, according to a predefined metric, e.g. Euclidean distance, Dubins path length.
- *Steer*: this function drives the system from $a_{nearest}$ to a_{rnd} minimizing the distance covered or any other cost function, i.e. time and fuel consumption. The trajectory between both nodes is represented by an edge e_i.
- *Safe*: it checks if an edge goes through not allowed areas.
- *AddNode*, *AddEdge*: these functions include a node a_i or an edge e_i in the sets A and E, respectively.

By using the procedures here listed, the RRT pseudocode is summarized in Algorithm 1.

Algorithm 1 $T = (A, E) \leftarrow \text{RRT}(a_{start})$

```
T ← InitializeRRT();
T ← AddNode(a_start, T);
while i < MaxIter do
    a_rnd ← RandomSample();
    a_nearest ← NearestNode(a_rnd, T);
    e_i ← Steer(a_nearest, a_rnd);
    if Safe(e_i) then
        T ← AddNode(a_rnd);
        T ← AddEdge(e_i);
    end if
end while
return T
```

In order to grow a tree T, the RRT algorithm is initialized with the initial node a_{start}. For a predefined maximum number of iterations $MaxIter$, a random sample a_{rnd} is taken from X and there is an attempt to grow an edge and connect it to the closest node a_{nearest} in A. If the connection is successful and the edge does not go through X_{storm}, the sample and the edge are included in A and E, respectively. An example of RRT expansion is shown in Fig. 1. If the number of iterations is sufficiently large, almost every region in X can be connected with the initial position by a sequence of tree edges [19]. In trajectory planning problems, the main goal is to add the node a_{goal} to A so that it is connected to a_{start}.

Fig. 1 Example of RRT expansion with 200, 500 and 1000 iterations (from left to right)

2.2 *RRT* Algorithm*

Despite being able to obtain a feasible trajectory between two state configurations, the RRT does not ensure trajectory optimality. That means the solution will cover almost surely more distance than required, leading to a higher and unnecessary cost. An update for the RRT was introduced in [20], the optimal RRT, denoted RRT*. The RRT* is a path planning technique that ensures feasibility and asymptotic optimality of a trajectory between two given state configurations. The RRT* algorithm inherits the main core from the RRT including three additional features:

- *Near*: it obtains the set of nodes $A_{\text{near}} \subseteq A$ within a ball $B(a_{\text{rnd}}; k\frac{\log n}{n})$, with k a constant that depends on the RRT and n the number of nodes in T.
- *Parent*: among the set A_{near}, this function finds the node a_{min} that involves the smallest cost from a_{start} to a_{rnd} passing by a_{min}. The node a_{min} is selected as the parent of a_{rnd}.
- *Rewire*: the rewiring process checks if the cost from the initial node to each of the elements in A_{near} can be reduced going through a_{rnd}. It changes parent–child relations, establishing new edges.

These functions are included after line 7 in Algorithm 1. Once the safety of a possible edge has been ensured, the set of nodes A_{near} in the vicinity of a_{rnd} is obtained using the *Near* function. Given $a_{\text{near},i} \in A_{\text{near}}$, the cost c_i of the trajectory between a_{start} and a_{rnd} through $a_{\text{near},i}$ is calculated for each i. The node with the smallest value of c_i is chosen as the parent of a_{rnd} and denoted by a_{min}. The rewiring process takes place afterwards, removing non-optimal connections, so that each node is connected to the root of the tree with the smallest possible cost.

Parent–child relations are an important part of the algorithm, as one node can have many children, but each node only has one parent. Consequently, if we select an arbitrary node in A, we will be able to reach a_{start} just by following the sequence of parents. The aim of the RRT is then to find a parent for a_{goal} so that we can connect it to a_{start}. The RRT*, which is based on the RRT, goes further and removes redundant relations that involve longer trajectories between nodes.

3 Scenario Based RRT*

SB-RRT* is an extension of the RRT* algorithm in which the uncertain unsafe areas are provided as a set of possible scenarios. The tree growth is constrained in such a way that the solution is a safe continuous curve.

3.1 SB-RRT Expansion*

The SB-RRT* presented in this work is an update of the RRT* considering that X_{storm} is uncertain and described by a finite number of possible scenarios, all of which can be treated as deterministic. We assume, without loss of generality, that all the scenarios are equally weighted. However, the formulation can be extended to forecast members of different weights, if such information is available (see, e.g. [21]). In addition, we assume that, in each scenario, thunderstorms are objects described by closed curves.

Let the unsafe set X_{storm} be uncertain. In [15], the authors proposed a chance constrained RRT (CC-RRT), in which the probability of being inside X_{storm} was determined for each state configuration. In contrast to CC-RRT, we choose a scenario-based approach in which the environment is characterized by an ensemble forecast. The ensemble consists of different realizations, or scenarios, all of them possible. The set X_{storm} is composed of N_o different thunderstorms:

$$X_{\text{storm}} = \{C_1(\mathbf{p}_1), \ldots, C_{N_o}(\mathbf{p}_{N_o})\}, \tag{1}$$

$(\mathbf{p}_j)$ denotes a closed curve that describes the j-th thunderstorm and depends on an uncertain vector of parameters $\mathbf{p}_j \subset \mathbb{R}^{d_{p_j}}$, with $d_{p_j} \geq 1$. We assume that N_{sc} discrete realizations of X_{storm} are available, being all of them equally likely:

$$X^l_{\text{storm}} = \{C_1(\mathbf{p}^l_1), \ldots, C^l_{N_o}(\mathbf{p}^l_{N_o})\}, \quad \textit{with } l = 1, \ldots, N_{sc}, \tag{2}$$

where $C^l_j(\mathbf{p}^l_j)$ is a realization of the j-th thunderstorm obtained by sampling the uncertain vector $\mathbf{p}_j$. As each curve C^l_j can be treated as deterministic, we can determine whether a point lies inside it or a curve goes through it. In consequence, the SB-RRT* is able to work with a trajectory defined by a sequence of continuous curves (or edges) or by a sequence of discrete states (or nodes).

Once the environment is defined, the SB-RRT* expands according to the pseudocode in Algorithm 2. The proposed algorithm is essentially a RRT* as described in Sect. 2.2 that computes the safety of a sequence of edges and the following procedures that will be covered hereafter in Sect. ssec:safety (Fig. 2).

Algorithm 2 $T = (A, E) \leftarrow$ SB-RRT* (a_{start})

$T \leftarrow InitializeRRT()$;
$T \leftarrow AddNode(a_{start}, T)$;
while $i < MaxIter$ **do**
  $a_{rnd} \leftarrow RandomSample()$;
  $a_{nearest} \leftarrow NearestNode(a_{rnd}, T)$;
  $e_i \leftarrow Steer(a_{nearest}, a_{rnd})$;
  if $Safe(a_{rnd})$ **and** $Safe(e_i)$ **then**
    $A_{near} \leftarrow Near(a_{rnd}, A)$;
    $a_{min} \leftarrow Parent(a_{rnd}, a_{nearest}, A_{near})$;
    $T \leftarrow AddNode(a_{rnd})$;
    $T \leftarrow AddEdge(e_i)$;
    $T \leftarrow Rewire(a_{rnd}, a_{min}, A_{near})$;
  end if
end while
return T

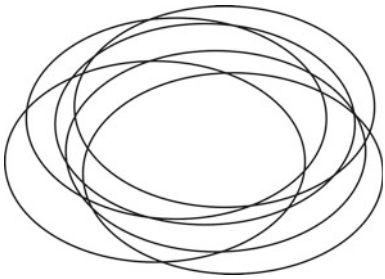


Fig. 2 Schematic representation of the different possible realizations of an unsafe region

3.2 Safety of a Trajectory

Algorithm 2 includes the function *Safe* that determines the safety of either nodes or edges. One of the key contributions of this work is the way the safety is computed so that the solution of the SB-RRT* can be considered safe as a whole continuous trajectory. In a planning in which the unsafe objects are described by deterministic parameters, a node is safe if it lies outside X_{storm} and an edge is safe if there is no intersection with any thunderstorm. If the environment is uncertain, we can only know that a node or an edge is safe up to a certain probability. In the present paper, given an event Z, the probability of Z being safe or not safe is represented by $(Z)^s$ and $(Z)^{ns}$, respectively. If $N_o > 1$, $(Z)^{ns,j}$ denotes the event of Z being not safe in the presence of the j-th thunderstorm. The different definitions of safety that are used in our work are listed down below.

Safety of a node: We say that a node a_i is safe if it is outside all the thunderstorms in X_{storm} with a probability of at least $1 - \epsilon_a$. In other words, it can only be inside at most $\lfloor \epsilon_a N_{sc} \rfloor$ deterministic realizations of unsafe object C_j^l. That is,

$$\Pr((a_i)^{ns}) = \Pr\left(\bigvee_{j=1}^{N_o}(a_i)^{ns,j}\right) \leq \epsilon_a \tag{3}$$

Safety of an edge: In a similar way, we say that an edge e_i is safe if the probability that it intersects with any of the thunderstorms in X_{storm} is less than a safety margin ϵ_e. It means that e_i only can interact with at most $\lfloor \epsilon_e N_{sc} \rfloor$ realizations of unsafe object C_j^l. That is,

$$\Pr((e_i)^{ns}) = \Pr\left(\bigvee_{j=1}^{N_o}(e_i)^{ns,j}\right) \leq \epsilon_e \tag{4}$$

Safety of a trajectory: Let $E^* = \{e_1^*, \ldots, e_{N^*}^*\}$ be the solution of the SB-RRT*, formed as a concatenation of safe edges e_i^*. Let N^* be the number of edges in E^*. The solution of the SB-RRT* will be safe, with a safety margin ϵ, if all the edges from E^* are safe at the same time [22]. That is,

$$\Pr((E^*)^s) = \Pr\left(\bigwedge_{i=1}^{N^*}(e_i^*)^s\right) \geq 1 - \epsilon. \tag{5}$$

By using De Morgan's law, which states that the negation of a conjunction is equal to the disjunction of negations, (5) can be rewritten as,

$$\Pr((E^*)^{ns}) = \Pr\left(\bigvee_{i=1}^{N^*}(e_i^*)^{ns}\right) \leq \epsilon. \tag{6}$$

With Boole's inequality, which states that for a finite number of events Z_i, we have $\Pr(\bigvee_i Z_i) \leq \sum_i \Pr(Z_i)$, and we can conservatively satisfy (6),

$$\Pr((E^*)^{ns}) \leq \sum_{i=1}^{N^*}\Pr((e_i^*)^{ns}) \leq \epsilon. \tag{7}$$

Using (4) and Boole's inequality, (7) is replaced by,

$$\Pr((E^*)^{ns}) \leq \sum_{i=1}^{N^*}\sum_{j=1}^{N_o}\Pr((e_i^*)^{ns,j}) \leq \epsilon. \tag{8}$$

Combining equations (5) and (8), the safety constraint for the solution E^* is,

$$\Pr((e_i^*)^{ns,j}) \leq \epsilon_{ij}, \tag{9}$$

which must be satisfied for $i = 1, \ldots, N^*$ and $j = 1, \ldots, N_o$. The elements ϵ_{ij} are individual safety margins for the i-th edge in the presence of the j-th thunderstorm.

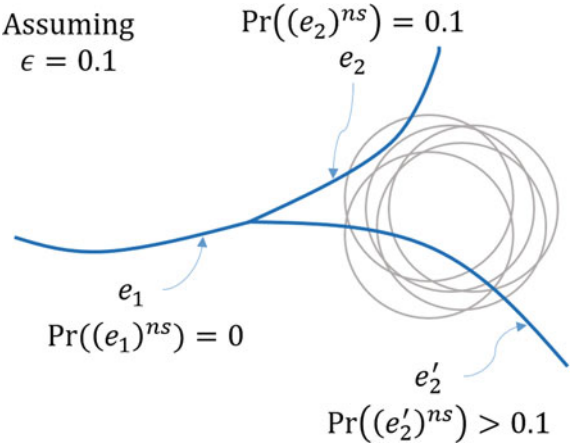


Fig. 3 Example of dynamic risk allocation

They must verify $0 \leq \epsilon_{ij} \leq 1$ and $\sum_{i=1}^{N^*} \sum_{j=1}^{N_o} \epsilon_{ij} \leq \epsilon$. In addition, the safety margin for the i-th edge considering the N_o thunderstorms is denoted by ϵ_i and verifies $\sum_{j=1}^{N_o} \epsilon_{ij} \leq \epsilon_i$. In order to allocate the risks ϵ_{ij}, it can be done in a uniform manner, so that $\epsilon_{ij} = \epsilon/(N^* N_o)$. This kind of allocation is overly conservative. Moreover, the value of N^* in any RRT is not known a priori as it is part of the solution and should be estimated. In reality, some parts of the trajectory will go through areas of no risk and as we get closer to the unsafe set the actual risk will increase.

This work proposes a *dynamic risk allocation* in which the risk is non-uniformly assigned to the different edges as the SB-RRT* is growing. This non-uniform risk allocation leads to a less conservative and more optimal solution in terms of distance. During the expansion, the probability of interaction with X_{storm} of any trajectory starting at $a_{start} \in A$ can be of, at most, ϵ. An example of the risk allocation with $\epsilon = 0.1$ is illustrated in Fig. 3. An arbitrary trajectory starts with an edge e_1, which is unsafe with probability 0. Then, the tree is able to grow in directions in which, according to (7), the total sum of probabilities of being unsafe is bounded by 0.1. Consequently, the edge e_2, which is unsafe with probability 0.1 is accepted:

$$\Pr\left(\bigvee_{i=1}^{2} (e_i)^{ns}\right) \leq \Pr((e_1)^{ns}) + \Pr((e_2)^{ns}) \leq 0.1.$$

On the contrary, e_2' is rejected, as the resulting sum of probabilities is higher than 0.1:

$$\Pr\left(\bigvee_{i=1}^{2} (e_i)^{ns}\right) \leq \Pr((e_1)^{ns}) + \Pr((e_2')^{ns}) > 0.1.$$

The tree could continue growing from e_2 provided that (7) is verified. The definitive mathematical formulation of the dynamic risk allocation is in progress and will be presented in future work

4 Case Study

In this section, the SB-RRT* is tested considering a simplified point-mass model of aircraft. We solve the problem in a simple environment formed by circular uncertain unsafe areas that represent possible stormy regions.

4.1 Aircraft Dynamics

We assume that it is flying at constant velocity V and constant altitude. Let $\mathbf{s} = (x, y, \lambda)$ be the state vector. The vector $\mathbf{x} = (x, y) \subset \mathbb{R}^2$ represents the aircraft position and $\lambda \in [-\pi, \pi]$ its heading angle. The manoeuvres are limited by the aircraft minimum turning radius $R_{\min}$ or equivalently, its maximum yaw rate $u_{\max} = V/R_{\min}$. For simplicity, only three control inputs are considered: no turn, right turn or left turn (with $u_{\max}$ or $R_{\min}$). The system dynamics is given by,

$$\dot{\mathbf{s}} = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\lambda} \end{Bmatrix} = \begin{Bmatrix} V\cos\lambda \\ V\sin\lambda \\ u \end{Bmatrix}, \tag{10}$$

with controls, $u \in \{-u_{\max}, 0, u_{\max}\}$. Each time we want to expand the SB-RRT* between two nodes and we must solve the system in (10) minimizing the distance covered. Given two states $\mathbf{s}_0 = (x_0, y_0, \lambda_0)$ and $\mathbf{s}_f = (x_f, y_f, \lambda_f)$, there exist an analytical solution for this optimization problem, the Dubins path [23]. Dubins paths are continuous and differentiable curves formed by one of the following:

- Three arcs of circle of radius $R_{\min}$.
- Two arcs of circle of radius $R_{\min}$ with one straight line in between.

Dubins paths are included in two functions from the SB-RRT* algorithm:

- *NearestNode*: they are used as the metric. When looking for the closest node to a_{rnd} from the tree T, it is appropriate to use the shortest Dubins path, as it considers the heading of the nodes and the minimum turning radius R_{min}. Using another metric, such as the Euclidean distance, could lead to manoeuvres that violate the turning constraint [10].
- *Steer*: this function drives the system from $a_{nearest}$ to a_{rnd} with a Dubins path.

4.2 Problem Setting

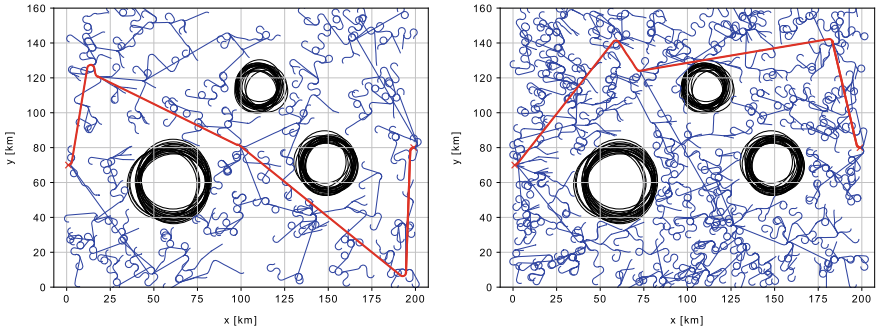
As a case study, the SB-RRT* is tested in a domain $X = [0, 200] \times [0, 160]$ (in km), with an aircraft flying between (0, 70) and (200, 80), and manoeuvering with $R_{\min} =$ 2 km. The unsafe set X_{storm} is formed by $N_o = 3$ circular and static thunderstorms

of known radii and uncertain centre positions sampled from a Gaussian distribution. Mean positions of the centres are (60, 60), (150, 70) and (110, 115), with radii of 20, 15 and 12 km, respectively. In this example, circles are used because they admit an analytical intersection with Dubins paths. The number of scenarios considered is $N_{sc} = 20$. A maximum risk of 10%, or $\epsilon = 0.1$, is allowed. In consequence, only $\lfloor \epsilon N_{sc} \rfloor = 2$ intersections with the unsafe set are permitted. They can occur with the same thunderstorm or be distributed between two. The algorithm is implemented in Python (basic Python and Numpy library). The computations are performed in a workstation equipped with an Intel Core i7-8550U CPU running at 1.8 GHz.

4.3 Results and Discussion

Figure 4 displays the SB-RRT* evolution and solution in two stages of its expansion, at the 500th and 1000th iterations. It can be seen that a higher number of iterations involve less distance covered by the solution trajectory. This fact results from the appropriate parent choice and the rewiring process, both of which keep optimizing the tree structure as it grows, removing redundant connections and ensuring that each point is connected to the root with the shortest possible trajectory. In addition, it can be observed that the tree is successfully avoiding the discrete realizations of uncertain thunderstorms. Each node, including the target, is connected to the starting point with a trajectory that involves at most two interactions with the unsafe regions. This fact guarantees that the flight would be safe in a 90% of the possible scenarios.

Figure 5 shows the solution after the 2000th iteration. As the number of iterations increase, the algorithm is able to find solutions through the corridor between the thunderstorms reducing the total distance that is required. In Table 1, the reduction in the distance covered with the number of iterations is shown. As a lower bound, the straight trajectory connecting the initial and target positions involved 200 km, but it is not valid as it assumes a high risk. With 2000 iterations, a 207 km trajectory



(a) 500 iterations (b) 1000 iterations

Fig. 4 SB-RRT* expansion and solution (in red) for different maximum number of iterations

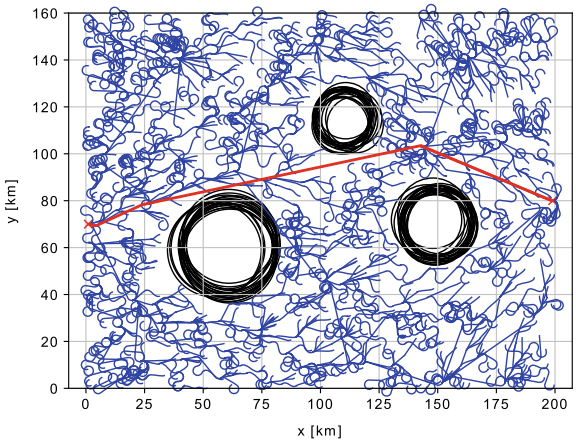


Fig. 5 SB-RRT* solution after 2000 iterations

Table 1 Length of the solution trajectory and execution time for the SB-RRT* with the number of iterations

Iterations	Solution length (km)	Execution time (s)
500	355	76
1000	291	283
2000	207	1160

was obtained, meeting the safety requirement. That is, a 3.5% increase in the total distance meant a 90% rise in the safety of the solution.

However, the main drawback of the SB-RRT* is the asymptotic convergence. That means there is no guarantee of optimality unless the number of iterations tends to infinity. In this example, increasing the number of iterations above 2000 was of little benefit, as it required an excessive computational time with no important shortening of the solution trajectory. As can be seen in Table 1, doubling the number of iterations means that the execution time grows by a factor of approximately 4. The main bottleneck in the SB-RRT* algorithm is *Safe* function, which checks if an edge is safe before being added to the tree or during the rewiring step. It must check if there is an interaction with any object in any scenario. The function is called in line 7 from Algorithm 2 and recursively inside *Parent* and *Rewire*. In particular, the safety checks from these two functions are what cause the exponential increase in CPU time. Both evaluate multiple connections during each iteration, one per node in the set A_{near}. As the iterations increase, the total number of nodes in A grows, and so does the possible nodes in A_{near}. Adding, for the total number of iterations, the CPU time associated with *Safe* function times the number of calls of the function leads to an exponential trend. Further research will be required to reduce the execution time of the algorithm, being able to work in timescales compatible with near-real-time modification of trajectories.

5 Conclusions and Future Work

We presented an incremental sampling-based algorithm for flight planning, the SB-RRT* able to obtain safe trajectories in an environment formed by stochastic unsafe regions. Provided that the uncertainties are characterized with an ensemble of discrete realizations, the algorithm ensures that the trajectory between two state configurations and never violates a predefined safety margin. The ability of the tree to grow, constrained by a maximum number of interactions with the unsafe set, is demonstrated. A RRT-based algorithm was chosen due to its versatility, and it can be easily modified and updated. For the moment, the algorithm has been tested assuming constant flight level, however, the extension to variable altitude considering 3D Dubins paths is immediate, e.g. in [24]. Moreover, no operational constraints have been considered, but RRT algorithms are compatible with speed or spatial limitations (see kinodynamic RRT algorithms, e.g. [25]). The main disadvantage of a RRT* is that theoretically, an infinite number of iterations are required for optimality. Nonetheless, the RRT*-smart extension presented in [26] can be incorporated, leading to an increase in the rate of convergence and significantly reducing the number of iterations required to approach the optimal trajectory. In future, data from real weather forecasts must be integrated in the algorithm. The thunderstorms to be avoided will not be described by analytical expressions, requiring a strategy to obtain intersections between tree edges and more general curves.

References

1. Eurocontrol, in *Performance Review Report. An Assessment of Air Traffic Management in Europe During the Calendar Year 2017*. Technical Report (2017)
2. H. Erzberger, T. Nikoleris, R.A. Paielli, Y.C. Chu, Algorithms for control of arrival and departure traffic in terminal airspace. Proc. Inst. Mech. Eng. Part G: J. Aerospace Eng. **230**(9), 1762–1779 (2016)
3. D. González-Arribas, M. Soler, M. Sanjurjo-Rivo, M-Kamgarpour, J. Simarro, Robust aircraft trajectory planning under uncertain convective environments with optimal control and rapidly developing thunderstorms. Aerospace Sci. Technol. **89**, 445–459 (2019)
4. S. Summers, M. Kamgarpour, J. Lygeros, C. Tomlin, A stochastic reach-avoid problem with random obstacles, in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control (HSCC'11)* (ACM, New York, NY, USA, 2011), pp. 251–260
5. D. Hentzen, M. Kamgarpour, M. Soler, D. González-Arribas, On maximizing safety in stochastic aircraft trajectory planning with uncertain thunderstorm development. Aersopace Sci. Technol. **79**, 543–553 (2018)
6. S.M. LaValle, *Rapidly-Exploring Random Trees: A New Tool for Path Planning* (Iowa State University, Tech. Rep., 1998)
7. S.M. LaValle, J.J. Kuffner, Randomized kinodynamic planning, in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol 1 (1999), pp. 473–479
8. S.R. Martin, S.E. Wright, J.W. Sheppard, Offline and online evolutionary bi-directional RRT algorithms for efficient re-planning in dynamic environments, in *IEEE International Conference on Automation Science and Engineering* (2007), pp. 1131–1136
9. P. Cheng, Z. Shen, S.M. LaValle, RRT-based trajectory design for autonomous automobiles and spacecraft. Arch. Control Sci. **11**(4), 167–194 (2001)

10. Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, J.P. How, Real-time motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. **17**(5), 1105–1118 (2009)
11. C.E. Tuncali, G. Fainekos, *Rapidly-Exploring Random Trees-Based Test Generation for Autonomous Vehicles*. Arizona State University, Technical Report (2019)
12. J. Kim, J.P. Ostrowski, Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints, in *IEEE International Conference on Robotics and Automation*, vol 2 (2003), pp. 2200–2205
13. K. Yang, S. Sukkarieh, 3D smooth path planning for a UAV in cluttered natural environments, in *IEEE/RSJ. International Conference on Intelligent Robots and Systems* (2008), pp. 794–800 (2008)
14. Y. Bouzid, Y. Bestaoui, H. Siguerdidjane, Quadrotor-UAV optimal coverage path planning in cluttered environment with a limited onboard energy, in *IEEE 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017)
15. B.D. Luders, M. Kothari, J.P. How, Chance constrained RRT for probabilistic robustness to environmental uncertainty, in *AIAA Guidance, Navigation and Control Conference* (2010)
16. B.D. Luders, J.P. How, Probabilistic feasibility for nonlinear systems with non-gaussian uncertainty using RRT, in *AIAA Infotech@Aerospace Conference*, St. Louis, MO (2011)
17. L. Blackmore, A probabilistic particle control approach to optimal, robust predictive control, in *AIAA Guidance, Navigation, and Control Conference and Exhibit* (2006)
18. *Guidelines on Ensemble Prediction Systems and Forecasting*. World Meteorological Organization (2012)
19. S.M. LaValle (ed.), *Planning Algorithms* (Cambridge University Press, New York, NY, USA, 2006)
20. S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**, 846–894 (2011)
21. Y. Matsuno, R. Kikuchi, N. Matayoshi, Robust optimal guidance algorithm for required time of arrival operations using probabilistic weather forecasts, in *AIAA SciTech Forum* (2019)
22. V. Lefkopoulos, M. Kamgarpour, *Using Uncertainty Data in Chance-Constrained Trajectory Planning* (2019). [Online]. Available: http://arxiv.org/abs/1904.12825
23. A.M. Shkel, V. Lumelsky, Classification of the Dubins set. Robot. Autonom. Syst. **34**, 179–202 (2001)
24. P. Pharpatara, B. Hérissé, Y. Bestaoui, 3-D trajectory planning of aerial vehicles using RRT*. IEEE Trans. Control Syst. Technol. **25**, 1116–1123 (2017)
25. S. Karaman, E. Frazzoli, Optimal kinodynamic motion planning using incremental sampling-based methods, in *49th IEEE Conference on Decision and Control (CDC)* (2010)
26. F. Islam, J. Nasir, U. Malik, Y. Ayaz, O. Hasan, "RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution," in. IEEE International Conference on Mechatronics and Automation **2012**, 1651–1656 (2012)