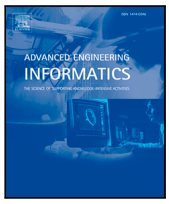
Advanced Engineering Informatics 64 (2025) 102975

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



A flight-procedure generation framework based on an RRT* path planning algorithm

Raúl Sáez [a,*], Daichi Toratani [a], Ryota Mori [b], Xavier Prats [c]

[a] Air Traffic Management Department, Electronic Navigation Research Institute (ENRI), Chofu, Tokyo, Japan
[b] Graduate School of Maritime Sciences, Kobe University, Kobe, Hyogo, Japan
[c] Department of Physics-Aerospace division, Technical University of Catalonia (UPC), Castelldefels, Barcelona, Spain

ARTICLE INFO

Keywords:
Path planning
Required navigation performance
Performance based navigation
Flight procedure design
Rapidly-exploring random tree

ABSTRACT

We introduce a flight-procedure generation framework that relies on the optimal version of the rapidly-exploring random tree (RRT*) algorithm, where nodes are connected using Dubins paths. This framework is employed for the generation of required navigation performance authorization required approach (RNP AR APCH) procedures, taking into consideration design constraints established by the international civil aviation organization (ICAO). To evaluate the framework's viability, we examine a series of challenging scenarios, for which we swiftly generate a series of operationally-compliant flight procedures. Furthermore, we assess how several of the RRT* configuration parameters can affect the generated procedures. The automation capabilities provided by our framework will benefit flight-procedure designers, saving time in the design process and enabling them to propose flight procedures which will potentially lead to more efficient aircraft trajectories.

1. Introduction

Flight procedure design plays a crucial role in civil aviation, impacting various aspects of aircraft operations. It directly influences factors such as flight duration and fuel consumption. Additionally, it has an effect on airport accessibility for flights operating under instrumental flight rules (IFR), ultimately affecting airport capacity. Furthermore, it can be used to reduce (or better distribute) the noise exposure of populated areas in the vicinity of airports.

Traditionally, the design of flight routes (i.e., the horizontal component of the procedure) was significantly constrained by the physical locations of ground-based navigational aids (Fig. 1). As a result, there was limited flexibility when designing a route. However, advancements in navigation systems, particularly the introduction of area navigation (RNAV) and the usage of global navigation satellite systems (GNSS) for civil aviation, have ushered in a new era of more adaptable route designs. Notably, required navigation performance (RNP) approaches have gained prominence.

RNP is a family of navigation specifications under performance-based navigation (PBN) that allows aircraft to follow precise flight paths while maintaining a specific level of accuracy and integrity in determining their positions. Within the realm of RNP procedures, a distinct category exists, known as RNP authorization required approach (RNP AR APCH) procedures. These procedures necessitate a lower lateral total system error compared to standard RNP values during any segment of the approach procedure.

RNP AR APCH procedures are characterized by their ability to define narrow, linear obstacle clearance corridors within the design. This is made possible by the assurance of precise navigation performance, enabled by aircraft on-board position monitoring and alerting systems. Regulatory authorities have labeled these procedures as "AR" due to the mandated monitoring and alerting systems on aircraft, as well as the essential pilot training required for executing these approaches. For operators seeking to employ RNP AR APCH procedures, a formal approval in the form of a letter of authorization or operational specification is needed.

The primary motivation behind the development of RNP AR APCH was to facilitate the implementation of procedures in challenging obstacle-rich environments (e.g., Queenstown Airport, approach RNP Y RWY 05 (AR) [2]) where conventional obstacle protection surfaces often limit feasibility. In addition, these procedures are equally useful in highly congested airspaces (e.g., John F. Kennedy International Airport in New York City, approach RNAV(RNP) Z RWY 04L [3]), where the reduction of obstacle protection surfaces helps mitigate potential conflicts between various traffic flows. Furthermore, RNP AR APCH could enable effective noise-abatement procedures (e.g., San Francisco International Airport, approach RNAV(RNP) Y RWY 28R [4]).

* Corresponding author.
E-mail addresses: raul.saez.garcia@upc.edu (R. Sáez), toratani-d@mpat.go.jp (D. Toratani), r-mori@people.kobe-u.ac.jp (R. Mori), xavier.prats@upc.edu (X. Prats).

https://doi.org/10.1016/j.aei.2024.102975
Received 16 April 2024; Received in revised form 29 October 2024; Accepted 25 November 2024
Available online 18 December 2024
1474-0346/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

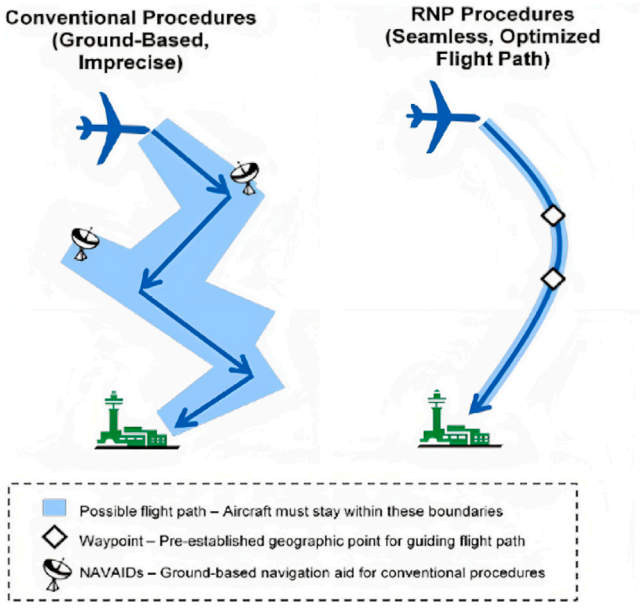


Fig. 1. Conventional and RNP navigation methods [1].

Other benefits of RNP AR APCH procedures include an increase of airspace efficiency through reduced separation; a better use of multiple airport runway configurations for increased airport capacity; or reduced fuel burn/emissions from shorter flight paths via not being constrained to overfly navigational-aids on the ground.

Depending on the scenario, manually designing an IFR procedure can present challenges. On the one hand, designing a procedure for the arrival of an aircraft at a runway involves complex constraints. For example, in the case of an RNP AR APCH, the procedure may require a combination of straight segments and curved flight paths, referred to as radius-to-fix (RF) legs. Furthermore, akin to conventional flight procedures, the flight path must maintain a specific clearance from the ground surface and obstacles. Moreover, the designed procedure must align with the rules set forth by the International Civil Aviation Organization (ICAO) for procedure design [5,6]. These rules play a vital role in the creation of an optimal aircraft approach procedure. On the other hand, several potential procedures could comply with the constraints set by ICAO, rendering the selection of the optimal one a complex task.

In a prior publication [7], some of the authors of this paper explored the potential of automating the generation of optimal RNP AR APCH procedures using genetic algorithms (GAs). Optimal procedures were successfully derived through a series of case studies across multiple airports in Japan. However, the application of this approach had several limitations.

Firstly, the methodology employed involved several trial-and-error adjustments of various constraints and initial-guess settings to achieve optimal solutions. In addition, the computational times associated with this method were exceptionally long, particularly when evaluating constraints related to ground obstacles. This situation is undesirable from the perspective of procedure design: flight-procedure designers typically prefer a process that yields multiple feasible solutions, each satisfying the required constraints, in a relatively brief time frame, as opposed to obtaining a single optimal solution that requires a substantial amount of time for computation.

Given the aforementioned challenges, our work focuses on proposing and testing a novel path-planning algorithm designed to automatically generate optimal RNP AR APCH procedures. The field of path planning encompasses a broad range of approaches found in literature, each tailored to specific objectives. These objectives can span from establishing a secure path to optimizing different criteria such as distance, fuel consumption, time, and more, depending on the application's requirements.

In this paper, we present a flight-procedure generation framework that uses an adapted version of the RRT* path-planning algorithm to generate RNP AR APCH procedures, in which nodes in the RRT* tree are connected with Dubins paths.

While the RRT* algorithm and its variants are classic algorithms in path planning, the novelty of this paper lies in their specific application to the generation of RNP AR APCH procedures, which has not been extensively explored in existing literature.

This algorithm has a low time complexity, thus, leading to low computational times. It is also asymptotically optimal and it was recently used by some of the authors of this paper in a prior publication [8] to generate aircraft emergency routes.

More recently, in [9], we presented some preliminary work in which we used the flight-procedure generation framework to compute approach procedures for two airports in Japan. In this paper, we extend this previous work in several ways. First, the methodology has been completely revised and updated: the RRT* algorithm has been improved in several ways to yield straighter and, thus, paths more acceptable at an operational level (e.g., higher fuel-efficient paths and a reduction of workload for both ATCOs and pilots). Secondly, the Dubins paths have been integrated in the algorithm, allowing to simplify the previous 3-step process presented in [9] into a single-step process, thus, making the generation of the approach procedure more robust. More details regarding the several algorithms have also been added, as well as a more comprehensive explanation of RNP AR APCH procedures and their benefits. Furthermore, we present an analysis of how some of the RRT* configuration parameters affect the framework output. Finally, we have applied our framework in new airports involving challenging environments, with the aim of assessing the viability of our solution in new scenarios involving different constraints.

The research presented in this paper is anticipated to offer valuable insights to flight-procedure designers. Currently, flight-procedure-design software like PANADES [10] by NTT data or GéoTITAN® by CGX Aero and École Nationale de l'Aviation Civile (ENAC) [11] (Fig. 2) play a crucial role in the creation of flight procedures across all stages of flight. This kind of software adheres to the ICAO PANS-OPS criteria [5] and provides the capability to design procedures, evaluate obstacles, manage aeronautical information publication (AIP) data, and display data with exceptional accuracy and in multiple layers, including elevation information.

It is important to note that our results are not intended to supplant such comprehensive software solutions. However, through discussions with flight-procedure designers, it became evident that there is a considerable value in having an automated framework at their disposal. This framework could generate a set of initial (i.e., nominal) procedures, which designers could subsequently refine using the existing flight-procedure-design software.

In addition to providing an automated way to generate flight procedures, our framework could offer significant time savings for designers, particularly in the early stages when creating a draft flight procedure. Based on discussions with flight-procedure designers, they find the tool useful in reducing the time spent on repetitive tasks. Ultimately, designers still refine the procedures through trial and error, often needing to account for additional factors, such as local government restrictions or environmental considerations (e.g., avoiding routes over sensitive areas). This integration of our framework could simplify the initial drafting process while still enabling necessary adjustments at later stages.

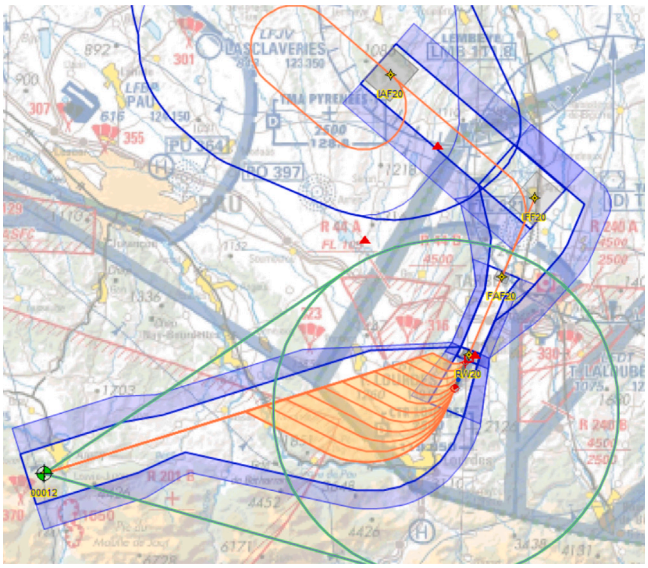


Fig. 2. Drawing a flight procedure in GéoTITAN® [11].

2. Literature review

An example of the manual design of a flight procedure was recently developed by Unkelbach et al. [12], who constructed an RNP AR APCH procedure at Isa Air Base in Bahrain by considering all the constraints imposed by ICAO and assessed the viability of their solution via simulations on a full-flight simulator. However, this study lacked the automation capabilities that a path-planning algorithm could provide.

Some of the most noteworthy path-planning algorithms include optimal control, graph-based methods, and front-propagation techniques, all of which allow to obtain optimal solutions. In recent years, sampling-based path-planning algorithms, initially introduced in [13] and refined for asymptotic optimality in [14], have gained prominence. Notable algorithms in this category include the probabilistic roadmap (PRM), rapidly-exploring random tree (RRT) and its optimal variant (RRT*), and fast marching trees (FMT*) [15]. While initially applied in robotics, these methods have found practical utility in aviation.

Previous studies have explored aircraft route generation through the use of various implementations of the RRT* algorithm. Andrés et al. [16] focused on thunderstorm avoidance by creating paths that navigate around thunderstorms predicted by ensemble weather models. They used a kinematic model for aircraft flying at a constant altitude and airspeed. Their findings provide useful insights for pilots, helping them safely divert from the original flight plan in response to thunderstorms, while also minimizing the cost of the operation. Another application focusing on hazardous weather avoidance was conducted by Qiu et al. [17]. A series of flight-restricted areas were considered in this work, which were subsequently avoided by paths generated by the RRT* algorithm.

In the context of military applications, Wei et al. [18] focused on the generation of aircraft trajectories aimed at penetrating airspaces without being detected by radars. They used a combination of the RRT* algorithm together with Dubins paths. A similar application was developed by Candemir et al. [19], aiming at generating paths which could be used in military missions. In a different military application, Meng et al. [20] used the RRT algorithm to generate a path for safely towing aircraft aboard carrier decks, where points in the RRT tree were connected by Reeds-Shepp curves.

In the field of emergency trajectories, Fallast et al. [21] used the RRT* algorithm and Dubins paths, together with a simplified aircraft dynamics model, to compute trajectories safely leading the aircraft to emergency landing sites. A more comprehensive aircraft dynamics model was used by Pharpatara et al. [22]. The main focus of the work, tackling a missile application, was to avoid obstacles by the computed trajectory.

Further examples of similar applications can be found in [23,24], where the dynamics of robots were taken into account for generating obstacle-avoiding paths using the RRT* algorithm. An advanced version of the RRT algorithm focusing on a robot application was explored in [25], in which Gaussian mixture models were employed in order to reduce the search space of the RRT.

A more recent development is highlighted in [26], where the authors proposed an application involving the use of the RRT algorithm for robot path planning. This application also involved a combination of path pruning and path smoothing using Bézier curves. Finally, an initial approach to incorporating RNP requirements with the use of the RRT algorithm was introduced in [27]. The authors used an A* algorithm for the path-pruning process, while path smoothing was achieved through B-spline curves. However, this approach did not account for aircraft turn dynamics.

Other authors, like Visser et al. [28] and Prats et al. [29] used an optimal control formulation with the main aim of optimizing the nominal flight path. Other related research includes [30,31], which used evolutionary algorithms in order to extend the nominal flight path optimization problem. However, these studies primarily focused on the design of noise abatement procedures, and they considered only the nominal flight path, thus, overlooking minimum obstacle clearance considerations and ICAO-established RNP corridors. In this context, a survey comparing different optimization methods can be found in [32].

Another remarkable approach related to the work presented in this paper was proposed by Hasegawa et al. [33]. The authors used genetic algorithms to design a fuel-efficient RNP AR APCH procedure complying with the ICAO design constraints. However, the authors made strong assumptions regarding the leg types and number of waypoints, making them match the published procedure for the scenario tackled in their work.

More recently, Chevalier [34] combined the simulated annealing method with the well-know Bellman–Ford algorithm to create an automated framework to generate IFR procedures. However, most of the ICAO constraints for flight procedure design were not considered in this work, and only the nominal flight path was generated.

As it has been discussed throughout this section, while several works can be found focusing on robot path planning by using sampled-based algorithms (such as RRT), not a lot of effort has been made in the field of flight procedure design with such kind of algorithms. Most of the works in the aviation field which use such kind of path-planning algorithms focus on the generation of aircraft trajectories, such as emergency trajectories, in which the aim is to avoid static or dynamic obstacles. And those works who do focus on flight procedure design often involve strong assumptions to simplify the problem or disregard most of the ICAO constraints for flight procedure design, yielding procedures which might not be acceptable from an operational point of view.

To the best of the authors' knowledge, no previous work has tackled the specific problem of the design of ICAO-compliant RNP AR APCH procedures [6] using the RRT* algorithm (or any other kind of path-planning algorithm), in which the nodes of the tree are connected by Dubins paths taking into account the achievable radius of turn of the arriving aircraft.

Despite the various applications of the RRT* algorithm in aviation—particularly in emergency trajectories and military contexts—there remains a significant gap in the literature regarding its use in designing ICAO-compliant RNP AR APCH procedures. This work aims to fill that gap by exploring how RRT* can be adapted to generate operational ICAO-compliant procedures, thereby extending the utility of established path-planning algorithms to enhance flight procedure design. Consequently, this approach offers flight procedure designers a practical tool for developing more effective and efficient procedures, ultimately improving safety in aviation.

3. RNP AR APCH design considerations

In this section, we describe some of the essential concepts of RNP AR procedure design, which were considered when applying the methodology used in this work. A series of constraints were taken into account when generating the approach procedures, which follow the guidelines regarding the design of RNP AR procedures defined by ICAO [5,6][1]. These constraints inherently incorporate considerations of aircraft dynamics and performance, ensuring that the generated flight procedures are achievable and operationally feasible. Note that only the most relevant constraints considered are described.

In the broader context of communication, navigation, and surveillance/air traffic management (CNS/ATM)—particularly for operations in oceanic or remote airspace—required communication performance (RCP) and required surveillance performance (RSP) are important considerations. However, these requirements do not apply directly to this study. Our main focus is on the design of RNP AR APCH procedures, where communication and surveillance are typically managed through conventional systems, such as very high frequency (VHF) communication and radar surveillance.

Among RNP procedures, RNP AR procedures are amongst the most modern and precise instrument approach options available nowadays, including unique capabilities that require special aircraft and flight crew authorization. They incorporate additional navigational accuracy, integrity and functional capabilities to permit operations using reduced obstacle clearance tolerances that enable approach and departure procedures to be implemented in circumstances where other types of approach and departure procedures are not operationally possible or satisfactory.

RNP AR procedures require a lateral total system error (TSE) lower than the standard RNP values on any segment of the approach procedure. More specifically, a lateral TSE as low as ±0.1 NM during 95% of the flight time is required on any segment of the approach procedure.

In the following sections, we will describe some of the most relevant constraints for the problem tackled in this paper. It should be noted that most of the information presented herein was already discussed in a prior publication by the same authors of this paper [9]. However, for the sake of coherence and to ensure a comprehensive understanding of the current study, we have opted to include this content here as well. We aim to provide context and clarity to the reader, particularly in relation to subsequent discussions and analyzes presented in this paper.

3.1. Segment and leg types

The RNP AR APCH procedures are composed of 4 segments: the initial, intermediate, final, and missed approach segments. Note, however, that in this work the missed approach segment is not considered.

The ICAO procedure design rules [6] allow the use of only two leg types when flying RNP AR APCH procedures: track-to-fix (TF) and radius-to-fix (RF). TF legs are intercepted and acquired as the flight track to the following waypoint, as illustrated in Fig. 3(a). On the other hand, RF legs are defined as a constant radius circular path around a defined turn center that terminates at a fix, as shown in Fig. 3(b). TF legs are geodesic flight paths between two fixes and are the normal standard legs used in RNP AR procedures. TF legs are linked by fly-by waypoints or RF legs. More information regarding turns is given in Section 3.2.

3.2. Turns

The turn constrains considered in RNP AR procedures depend on the leg type (i.e., TF or RF). For TF legs, the use of fly-over waypoints is not permitted, so only fly-by turns are used.[2] Furthermore, turn angles for TF legs are limited to a maximum of 70 degrees when aircraft are expected to fly-by the fix at altitudes above FL190, and to 90 degrees at and below FL190. An RF leg should be used if these constraints cannot be met.

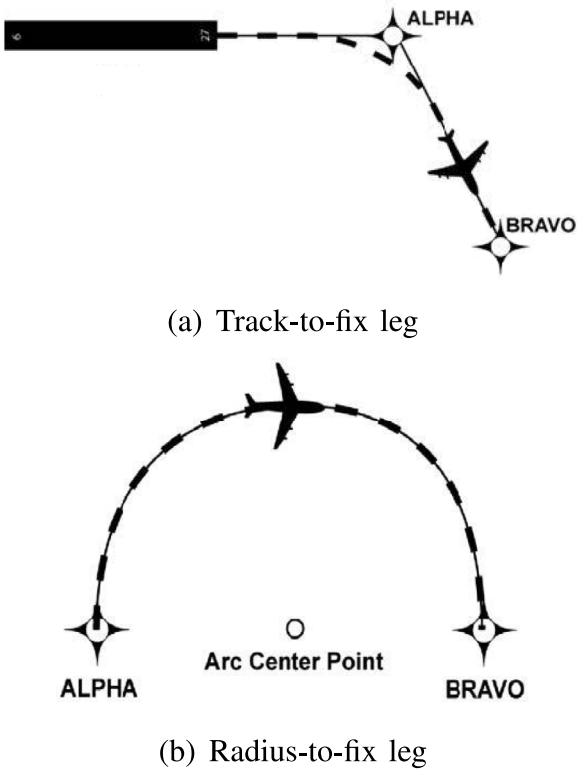


Fig. 3. RNP AR leg types [35].

Table 1
Bank angle window.

Lowest height	Max. ϕ [deg.]
<150 m (492 ft)	3
≥150 m (492 ft)	20

A standard bank angle (ϕ) of 18 degrees is used to calculate the radius of turn (r) applied at fly-by fixes. The turn is assumed to be performed at a given indicated airspeed (IAS) for the fastest aircraft category for which the procedure is designed. The corresponding true airspeed (V_{TAS}) is obtained for the highest altitude allowed in the turn; then, a tailwind component (w), which is also altitude-dependent, is added to it. r is computed as follows:

$$r = \frac{V_{GS}^2}{g \tan\phi}, \tag{1}$$

where V_{GS} is the ground speed ($V_{TAS} + w$) and g is the gravity acceleration.

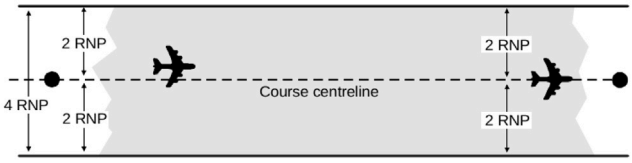
Non-standard bank angles (either lower or higher) are allowed for smooth transitions, maintaining stabilized approaches, lower minima or to achieve specific leg lengths [6]. Table 1 presents the allowed bank angle window as a function of the lowest height above the threshold in RF segments.

[1] The third edition of this document was recently published in 2021. Nevertheless, we use the second edition of the RNP AR Procedure Design Manual by ICAO, published in 2016, which is still applied in procedure design in most of the scenarios tackled in this paper.

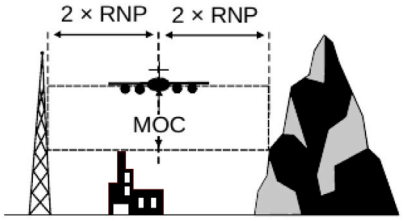
[2] Fly-by waypoints require turn anticipation to allow tangential interception of the next segment of a route or procedure, while flyover waypoints are waypoints at which a turn is initiated in order to join the next segment of a route or procedure, thus, overflying the waypoint.

Table 2
RNP navigation accuracy requirements, minimum obstacle clearance (MOC) & descent gradients.

Segment	RNP navigation accuracy requirements			MOC	Descent gradients	
	Minimum	Standard	Maximum	Standard	Standard	Maximum
Initial	0.1 NM	1 NM	1 NM	984 ft	4% (2.4 deg.)	8% (4.7 deg.)
Intermediate	0.1 NM	1 NM	1 NM	492 ft	≤2.5% (1.4 deg.)	Equal to FAS
Final	0.1 NM	0.3 NM	0.5 NM	Evaluated using OAS	5.2% (3 deg.)	See [6]



(a) Plan view



(b) Cross-section view

Fig. 4. RNP AR segment widths [6].

When using RF legs, the bank angle required for a given V_{TAS}, w and r can be directly computed by isolating ϕ from Eq. (1).

3.3. Segment widths, obstacle clearance and descent gradients

Segments in RNP AR procedures have a protection area width equal to 4 times the RNP navigation accuracy requirement and a semi-width equal to 2 times the RNP navigation accuracy requirement (Fig. 4). The minimum, standard and maximum values for RNP navigation accuracy requirements are presented in Table 2. As shown in Fig. 4(b), for each segment, an obstacle clearance with respect to the obstacles inside the protection area needs to be ensured. This clearance depends on the minimum obstacle clearance (MOC) value, which depends on each segment and is detailed in Table 2.

For the final approach segment (FAS), there is not a fixed value for the MOC and the obstacle assessment surface (OAS) needs to be computed first, as detailed in [6]. Computing the MOC for the vertical error budget (VEB) is needed in order to derive the height of the OAS at any distance from the landing threshold point (LTP).

When merging two segments with different protection-area widths at a given fix, it is necessary to evaluate for both segments the area within ±1 RNP navigation accuracy requirement of the fix. Regarding turns, for RF legs the protection-area width is equal to that of the straight segment, while for fly-by turns between two TF legs the procedure is more complex, as detailed in [6].

For each segment, a given standard and maximum descent gradient (γ) is considered, as detailed in Table 2. For the FAS, the maximum descent gradient depends on the aircraft category, as detailed in [6].

4. Methodology

In this section, we introduce and describe the underlying algorithms and workflow of the flight-procedure generation framework. Paths are generated by combining (1) the RRT* algorithm, a sample-based algorithm described in Section 4.1, and (2) Dubins paths, described in Section 4.2. A Dubins path is a type of trajectory which is used in this work to connect the nodes of the RRT* tree, leading to flyable procedures.

The framework itself is introduced in Section 4.3, where the functionality of each of the modules is explained in detail, as well as all the steps followed in the flight procedure computation process.

4.1. Rrt* path-planning algorithm

Extensively described in [14], the RRT* algorithm is a sampling-based algorithm particularly effective in high-dimensional configuration spaces[3] and environments with obstacles. It efficiently explores the configuration space to find feasible paths from a start configuration to a goal configuration while minimizing a specified cost metric.

The RRT* algorithm constructs a tree where each node represents a configuration and edges represent feasible motions between configurations. The aim is to find a tree that explores the entire free configuration space (i.e., the space free of obstacles) while minimizing the cost of reaching the goal configuration. In the initialization stage, the algorithm generates a graph composed solely of a single vertex or node, which corresponds to the initial state or start configuration.

In RRT*, nodes are not predefined on a grid but are instead dynamically sampled from the configuration space during the algorithm's execution. Edges represent feasible paths between these sampled configurations, accounting for obstacle avoidance and cost minimization.

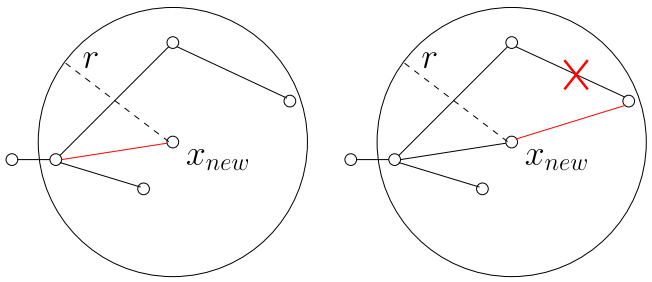
Let $\chi = (0,1)^l$ be the configuration space, where $l \in \mathbb{N}$, $(l \geq 2)$ denotes the space dimension.[4] Let χ_{obs} be an open set, which denotes the free configuration space (i.e., subset of configurations that do not intersect with obstacles) as $\chi_{free} = \mathrm{cl}(\chi \backslash \chi_{obs})$, where $\mathrm{cl}(\cdot)$ denotes the closure of a set. The initial configuration or state is denoted by $x_{init} \in \chi_{free}$ while the goal configuration or state is denoted by $x_{goal} \in \chi_{free}$.

The RRT* algorithm tree can be represented as $T = (V, E)$, a graph in which V is the set of nodes and E is the set of edges connecting these nodes. In order to grow the tree, new nodes are added to V as follows:

- At each iteration, the algorithm randomly samples a configuration $x_{rand} \in \chi_{free}$. Mathematically, x_{rand} follows a probability distribution defined over χ_{free}.
- The algorithm identifies the nearest node to x_{rand} in the current tree, denoted as $x_{nearest}$.
- The algorithm attempts to extend the tree from $x_{nearest}$ to x_{rand}. In general, this extension will be feasible if there are no obstacles in the path linking both configurations. If this is the case, x_{rand} is steered to $x_{nearest}$, resulting in a new node in the tree denoted as x_{new}.

[3] In the application presented in this paper, the configuration space is the geographical area surrounding the airport in which the approach procedure will be generated.

[4] $\chi = (0,1)^l$ is a multidimensional cube within an l-dimensional space. The configuration space is limited within the boundaries of the unit interval (0 to 1) along each of its l dimensions. The choice of the configuration space, its dimensions, and the range of values is problem-specific. The $(0,1)^l$ representation is used in this case to explain the concept of a bounded configuration space. In practice, these values would be adjusted to suit the constraints and characteristics of the real-world problem being addressed. In the problem presented in this paper, l is equal to 2 (i.e., latitude/longitude coordinates in the area surrounding the airport).



(a) New edge created (b) Previous edge removed

Fig. 5. Edge generation in RRT* [8].

- The cost of reaching x_{new} from the root of the tree is calculated as the sum of the cost to reach $x_{nearest}$ and the cost of the motion from $x_{nearest}$ to x_{new}. The algorithm then checks if any nearby nodes—denoted by the set χ_{near} and within a certain distance from x_{new} (known as rewiring radius)—can be reached from x_{new} with a lower cost. In order to create a new edge in the tree two elements need to be considered:
  - The feasibility of the connections between x_{new} and the nodes in χ_{near} is tested, and the one yielding the minimum cost is the one that will result in a new edge to be added to the tree, as shown in Fig. 5(a). This cost, in its simpler and most generic form, is usually the path distance. However, as it will be explained later in this section, the RRT* cost function can be modified in order to take into account other elements.
  - The algorithm conducts a rewiring process, in which new edges linking x_{new} to nodes in χ_{near} are generated if the path through x_{new} involves a lower cost than the cost of the path through the current parent node. In this situation, as illustrated in Fig. 5(b), the edge originally connecting the node to the current parent node is deleted so as to keep the structure of the tree.

Several parameters can be modified in order to obtain different solutions with the RRT* algorithm. The decision of which value to assign to these parameters is mainly driven by the application and/or the scenario considered. Providing the fact that in this paper we are dealing with flight procedure design, one of the conditions that the proposed procedures should ensure is having as many straight paths as possible, which leads to operationally-sound procedures with a higher fuel/time efficiency, a minimization of the workload for both the pilot and the ATCOs and, in general, a simplification of traffic management and navigation, ultimately leading to increased safety levels.

Therefore, the values for the RRT* parameters will be chosen accordingly. Still, depending on each scenario, we might need to slightly change some of the parameters in order to accelerate the convergence of the algorithm. Below, we detail some of the most relevant parameters that were tuned in the experiments considered in this paper. More details about the values chosen and the reasoning behind the choice are given in Section 5.1.

- **Step size**: the step size or the distance between each node added to the tree can influence the straightness of the paths. A larger step size can lead to more significant jumps in the configuration space, which may result in straighter paths. However, a very large step size can also lead to inefficiency and sub-optimal paths.
- **Rewiring radius**: as it was aforementioned, the RRT* algorithm uses a rewiring radius to connect nearby nodes to improve the paths. By adjusting the rewiring radius, it is possible to control the extent to which the algorithm connects neighboring nodes. A larger rewiring radius can lead to straighter connections, but it may also result in more complex trees.
- **Sampling bias**: the RRT* algorithm maintains a balance between exploration and exploitation. By biasing the sampling towards the goal, it is possible to encourage the algorithm to explore the space more efficiently towards the target, potentially leading to straighter paths. Our biasing approach involves the introduction of a probabilistic element to the RRT* sampling step. We generate a random number between 0 and 100, and compare it to a user-defined threshold (a value also between 0 and 100). If the random number is greater than the threshold, we generate a random node as per standard RRT* sampling. However, if the random number is less than or equal to the threshold, we generate a node that is closer to the goal configuration. This method of biasing allows us to explore the configuration space more efficiently, increasing the likelihood of finding a path towards the goal while still ensuring that the RRT* algorithm maintains its asymptotic optimality.
- **Maximum number of iterations**: the RRT* algorithm improves the path quality over time, so increasing the number of iterations will help in finding optimal and straighter paths. More iterations allow the algorithm to explore the configuration space better and converge towards better solutions. However, this comes at the expense of an increase in computational time. Furthermore, it is possible to configure the algorithm to finish once the goal is reached or to continue until reaching the maximum number of iterations.

Apart from the parameters aforementioned, the RRT* algorithm also includes a cost function, which, in its generic form, minimizes the total distance of the path. However, we have modified this cost function in order to encourage straighter paths.

As shown in Eq. (2), we use a weighted cost function where we consider both the distance of the path and the number of track changes. In Eq. (2), we show how we compute the total cost of a given tree branch, where n is the total number of nodes for the branch, d is the distance from node i to its parent node in the tree, w_t is the track change weight and t_i is equal to 1 if there is a change of track between node i and its parent node in the tree (as shown in Eq. (3)).

The value chosen for the weight parameter is detailed in Section 5.1, and it depends on each scenario. By penalizing the number of track changes we might end up with longer paths. However, in our current application (i.e., the design of flight procedures), having a straighter path is more acceptable at an operational level than having a slightly shorter path, but with too many turning waypoints.

$$cost_{branch} = \sum_{i=1}^{n} d_i + w_t \cdot t_i \tag{2}$$

$$t_i = \begin{cases} 1 & \text{if track change} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

A simple diagram depicting the main steps of the RRT* algorithm including all the concepts introduced in this section is shown in Fig. 6. The main steps during the path computation are described below:

- **Initialization**: a tree is generated, consisting only of the initial state. A configuration space is also defined, in which (in the following steps) the path between the initial state and goal state will be computed.
- **Sampling**: at each iteration of the algorithm, a random state is sampled within the configuration space, which represents a potential candidate for expanding the tree. As aforementioned, the step size parameter is used to limit the distance at which the random states are generated, while the sampling is also biased towards the goal.
- **Nearest neighbor search**: the node in the existing tree that is closest to the previously sampled state is obtained.

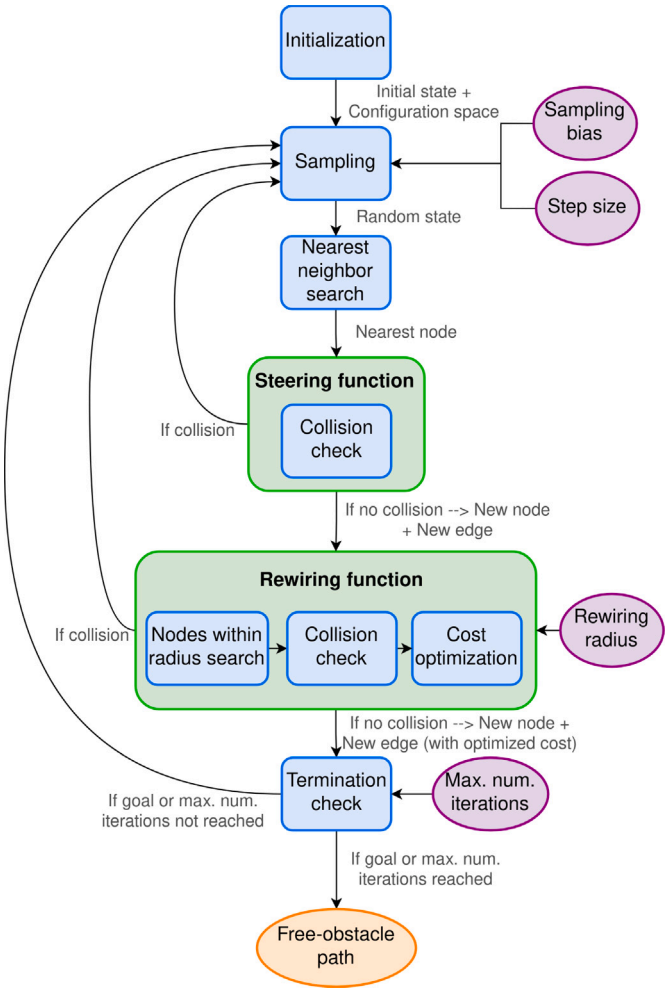


Fig. 6. RRT* diagram.

- **Steering function**: this function is used to generate a new state by moving from the nearest node in the tree towards the sampled state, while ensuring that the generated state is within the valid configuration space. This helps in gradually expanding the tree towards unexplored areas.
  In this step, the algorithm verifies the collision-free status of the path from the nearest node to the newly generated state by discretizing it for collision assessment. If obstacles are present along this discretized path, it is not added to the tree, triggering the sampling of a new random state. However, if the discretized path is collision-free, then the newly generated state becomes a new node in the tree, while the path from the nearest node to the new node is incorporated as an edge in the tree.
- **Rewiring function**: after a new node is added to the tree, the algorithm checks if there are other nodes in the tree within a given radius (i.e., rewiring radius) that can be reached more efficiently through the new node. If a path with a lower cost is found to an existing node, the tree is updated to use that path (providing that this path is obstacle-free).
  The algorithm maintains a cost associated with each node in the tree, which typically represents the length of the path from the initial state to that node. As the tree is expanded, the algorithm optimizes the path costs by considering the cost of the newly added edges. This ensures that the paths found by the algorithm are asymptotically optimal, meaning they get closer to the true optimal path as the number of samples increases.

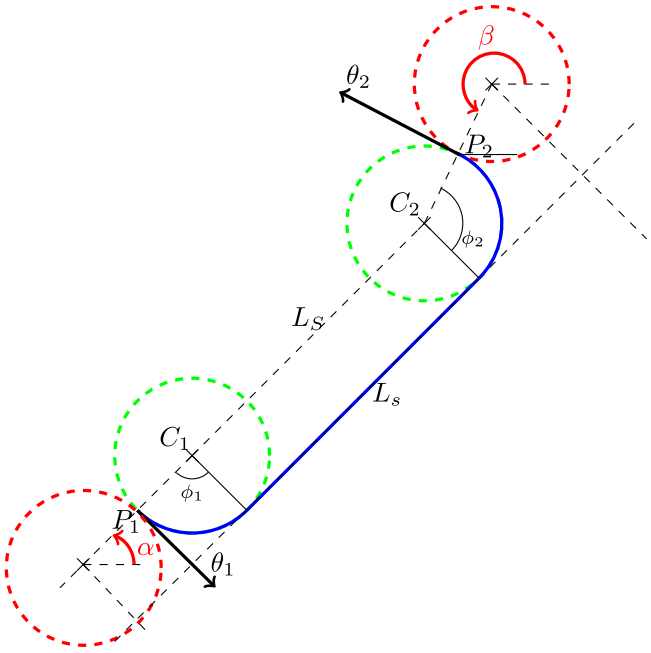


Fig. 7. Left-Straight-Left Dubins Curve [8].

- **Termination**: the algorithm continues to sample, connect nodes, and update the tree until a termination condition is met. This condition could be finding a path to the goal state or running for a predefined number of iterations. Once the termination condition is met, the output of the algorithm is a free-obstacle path connecting the initial and goal states.

4.2. Dubins paths

Dubins paths, first formally defined in [36], are trajectories used in motion planning that connect two configurations in the plane while minimizing the length traveled by a vehicle with a given minimum turning radius r. These paths consist of a sequence of circular arcs and straight lines. Each Dubins path is uniquely defined by a pair of coordinates (x and y) with an associated orientation angle (θ), representing the starting configuration and a destination configuration.

The path is constructed by connecting the starting and destination configurations with circular arcs and straight lines, allowing for smooth transitions between them. The circular arcs have a constant radius, determined by the minimum turning radius r, while the length of the straight lines is variable.

In the problem tackled herein, the orientation angle θ denotes the track angle between two fixes and the minimum turning radius r corresponds to the aircraft radius of turn computed as shown in Section 3.2.

The available types for each segment composing a Dubins path are the following: Right turn (R), Left turn (L), and Straight (S). This classification yields the set of possible Dubins paths: RSR, RSL, LSR, LSL, RLR, LRL.

For illustration purposes, an LSL Dubins path [8,37] is shown in Fig. 7. It is composed of a turn to the left around the first green circle (first arc), a segment of length L_S and a final left turn around the second green circle (second arc). ϕ_1 and ϕ_2 are the angles subtended by the first and second arc, respectively.

In Fig. 7, $P_1 = (x_1, x_2)$ and $P_2 = (x_2, y_2)$ represent the initial and the final points, respectively. θ_1 and θ_2 are the track directions of P_1 and P_2, respectively. Finally, α and β are the final angular positions on the start and end target circles. The target circles (depicted with a red-dashed line in Fig. 7), are the circles defined at the end position of the start and

end turns. The track direction at the final position on the target circle is tangential to the circle, and this direction could be either clockwise or counter-clockwise to the target circle. In this case (i.e. LSL), both final track directions are clockwise tangent to the target circles. When determining a Dubins path, the target circle is not explicitly defined as an independent entity. Instead, it is a consequence of calculating the radii and centers of the circular arcs that make up the turns in the path.

As aforementioned, in this case the final track direction is clockwise tangent to the target circle. Thus, the angular positions α and β are defined as follows:

$$\begin{cases} \alpha = \mod\left(\frac{\pi}{2} - \theta_1, 2\pi\right) \\ \beta = \mod\left(\frac{\pi}{2} - \theta_2, 2\pi\right). \end{cases} \tag{4}$$

The length L_S is equal to the distance between the centers of the green circles in Fig. 7 (i.e., C_1 and C_2, which can be computed using elementary geometry [37]). Then, the length of the LSL path (L_{LSL}) is computed by taking into account this distance and the angles ϕ_1 and ϕ_2 as follows:

$$L_{LSL} = L_S + r(\phi_1 + \phi_2) \tag{5}$$

Using geometry, one can derive L_S, ϕ_1 and ϕ_2 as follows:

$$L_S = \sqrt{(x_2 - r\sin\beta - x_1 + r\sin\alpha)^2 + (y_2 + r\cos\beta - y_1 - r\cos\alpha)^2}, \tag{6}$$

$$\phi_1 = \mod\left(\arctan\left(\frac{y_2 + r\cos\beta - y_1 - r\cos\alpha}{x_2 - r\sin\beta - x_1 + r\sin\alpha}\right), 2\pi\right), \tag{7}$$

$$\phi_2 = \mod(\beta - \phi_1, 2\pi). \tag{8}$$

As it is explained in Section 4.3, the nodes of the RRT* tree are connected with Dubins paths.

4.3. RNP AR APCH procedure-generation workflow

Our flight-procedure generation framework is mainly composed of a main module, which is in charge of running the RRT* algorithm (in which nodes are connected by Dubins paths) and a couple of auxiliary functions or sub-modules specifically tailored for the application tackled in this paper. Moreover, a set of inputs is necessary to operate the framework. Fig. 8 illustrates a basic diagram depicting the interactions among the various modules. The output of the main module, once the RRT* algorithm finishes, is a free-obstacle path from an origin (x_{init}) to a destination (x_{goal}). It should be noted that the path computation operates in a reverse manner, initiating in the runway. Essential inputs include not only the initial and final positions (i.e., x_{init} and x_{goal} in Fig. 8) but also the track angle for an effective aircraft landing (denoted as θ_{origin} in Fig. 8). Moreover, as discussed in Section 4.1, during the execution of the RRT* algorithm, connections from a new random sample to vertices in the RRT* tree within a defined distance (referred to as the rewiring radius) are evaluated using Dubins paths. This distance, alongside the step size for generating random samples, constitutes as well part of the required parameters of the RRT* module. Additionally, a predetermined maximum iteration count is established for the algorithm. Lastly, when introducing a new edge into the RRT* graph, adherence to the RNP AR APCH constraints, as outlined by ICAO [6], is ensured.

The several steps followed to generate this optimal path are described below:

(1) When the RRT* algorithm generates a new random sample and a new connection is tested, the algorithm connects the new sample to the tree by using Dubins paths (the same process is followed during the rewiring process). An example of this process is shown in Fig. 9(a). N_1, N_2 and N_3 are three nodes already existing in the tree. As it can be observed, N_1 and N_2 are already connected with a Dubins path, while N_2 and N_3 are just connected with a straight line as there is no track change. In the current RRT* iteration, a new random state is sampled, corresponding to N_{new}. The connection of this sample to the tree, if feasible (see following steps), is done with Dubins paths, depicted by a red-dashed line. In this case, the inputs needed to build the Dubins path (as explained in Section 4.2) are the initial and final track directions (i.e., θ_1 and θ_2 in Fig. 8) and the initial and final points (i.e., N_2 and N_{new} in Fig. 9(a) and introduced in a generic way as P_1 and P_2 in Section 4.2).
The "Turn radius computation" sub-module (Fig. 8) is in charge of all the required computations regarding the radius of turn. This radius is another of the required inputs for the Dubins paths computation, and is obtained as described in Section 3.2.

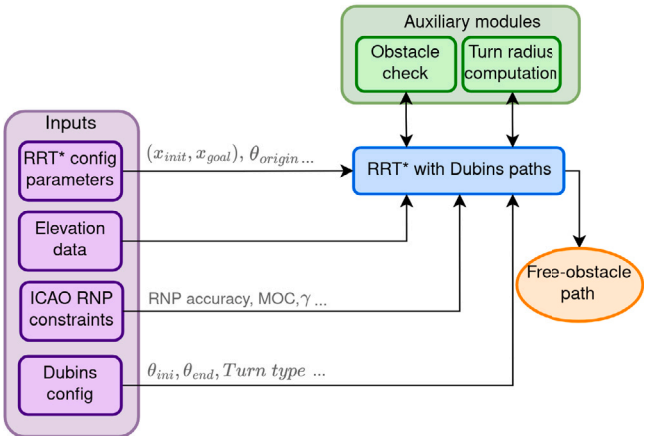


Fig. 8. RNP AR APCH procedure generation framework workflow.

(2) The descent gradient for each approach segment—depicted as γ in Fig. 8 and whose values are detailed in Table 2—is employed to calculate the altitude of the new random sample currently under consideration for addition to the procedure, potentially resulting in a new edge in the RRT* graph.

(3) The final approach fix (FAF) and intermediate fix (IF) altitudes are used to determine in which approach segment(s) the new edge of the RRT* graph will potentially be integrated. Note that this new edge could belong to different segments if a given fix altitude is reached within the edge. Moreover, the edge will involve a Dubins path, as shown in Fig. 9(a). In this paper, the FAF and IF altitudes are obtained from the approach charts for each of the scenarios considered 5.1. In future work, the algorithm will be modified in order to automatically choose these altitudes.

(4) The protection area width for each segment (illustrated by the two solid red lines in Fig. 9) is defined by using the RNP navigation accuracy requirements detailed in Table 2.

(5) The obstacle clearance is assessed for the current segment. Depending on the segment, a different MOC value is used (as detailed in Table 2), which is applied to the terrain within the protection area. If clearance from obstacles throughout the entire terrain within this area is kept, the RRT* graph is updated to include the new edge.
All the required operations are performed by the "Obstacle check" sub-module (Fig. 8). As shown in the simplified scenario of Fig. 9, the only existing obstacle is outside the protection area, so the obstacle clearance requirements are met. This means that the connection between the new random state N_{new} and the RRT* tree is feasible. A new edge is generated as shown in Fig. 9(b) and N_{new} is added to the tree as a new node, called N_3. Note that by creating this edge the previous node N_3 has been eliminated from the tree because of the Dubins paths used to connect the nodes.

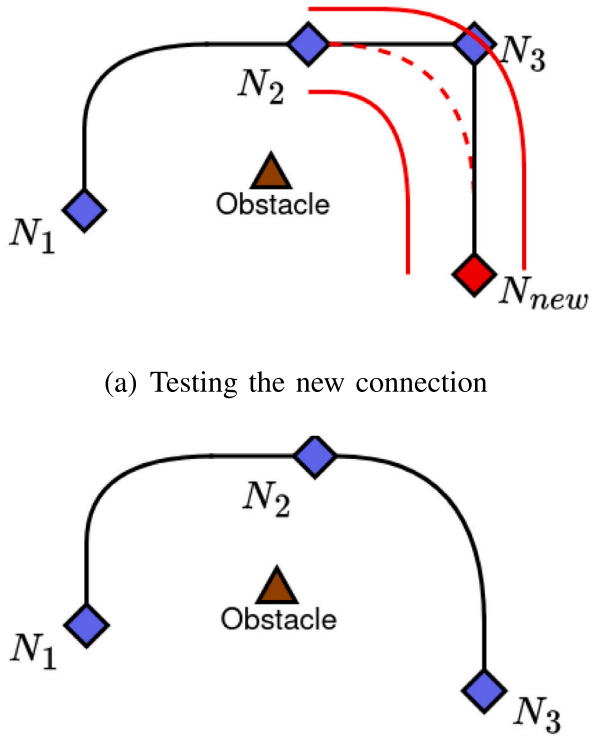


Fig. 9. RRT* tree generation with Dubins paths.

5. Results

In this section, we present the results obtained in this work. First, in Section 5.1, we describe the experimental setup, enumerating the several scenarios tackled in this paper. In Section 5.2, we focus on the flight procedure generation process, showcasing a practical example in which we highlight the main steps the algorithm follows and how the connections are established between the nodes in the RRT* tree.

The several generated procedures for each scenario are presented in Section 5.3. The aim is to show that our framework is capable of generating several potential RNP AR APCH procedures for each of the scenarios considered, some of them resembling those published in the approach charts and some of them slightly different but still viable and compliant with all the constraints defined by ICAO [6]. In Section 5.4, we focus on the computational time of our framework, and discuss its adequateness from the point of view of the aim of this work. In Section 5.5, we compare the approach followed in a previous work [33], based on genetic algorithms, with the approach presented in this work. In Section 5.6, we analyze the effect that the RRT* parameters have on the framework output. Finally, in Section 5.7, we discuss the role of our framework in the flight procedure design process, following several discussions with flight procedure designers.

5.1. Experimental setup

Three airports have been considered in this paper, representing the scenarios tackled: Kumamoto airport (RJFT), in Japan; Innsbruck airport (LOWI), in Austria; and Queenstown airport (NZQN), in New Zealand. From each of the several RNP AR approaches currently published for each of these airports, we chose one per scenario in order to determine the position of the IAF and the altitudes of the IF and the FAF (used as input for our framework as mentioned in step 3 in Section 4.3).

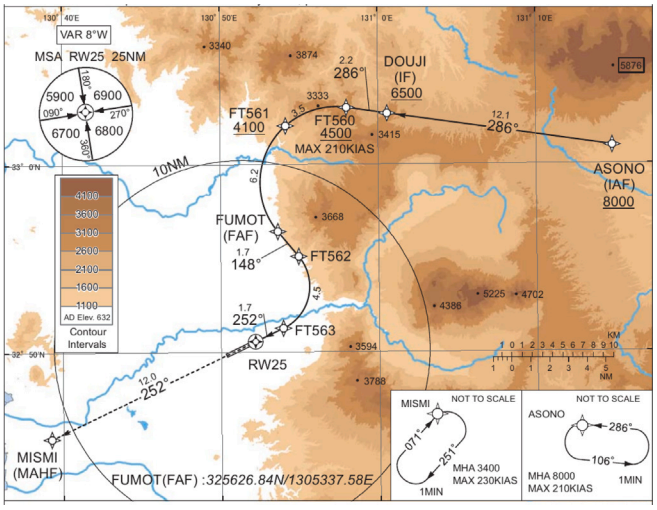


Fig. 10. Kumamoto Airport (RJFT) scenario: RNP AR Y RWY 25. *Source:* Japan AIP.

The chart data pertaining to the designated scenario was obtained from the aeronautical information publication (AIP) of the country in which the procedure is published. The description of the scenarios is detailed below:

- **Kumamoto Airport (RJFT)**: the rugged landscape east of RJFT is referred to as the Aso Volcano. A narrow passage, east of RJFT and flanked by mountains rising to 2600 ft, grants entrance to the caldera. This caldera extends 13.5 NM north to south and 9.7 NM east to west. Within its central expanse, a collection of volcanoes ranges from 4334 ft to 5225 ft in height, enveloped by lower-lying terrain varying between 1300 ft and 1800 ft. Surrounding the caldera are rim mountains, elevating from 2600 ft to 4100 ft. In this scenario, we consider the RNP AR Y RWY 25 approach, as shown in Fig. 10, which surrounds the caldera. A FAF altitude of 3200 ft and an IF altitude of 6500 ft are utilized (i.e., adhering to the minimum altitudes specified in the published chart). The origin altitude (note the algorithm operates in reverse) and heading are 632 ft and 244.6 degrees, respectively.
- **Innsbruck Airport (LOWI)**: this airport is located in a valley known as the Inn Valley, at an altitude of 1906 ft (gradually increasing up to 2400 ft towards the western edge of the valley). Most of the area of interest for this scenario is located in the Lower Inn Valley, which has a width ranging between 2.5 NM and 3.0 NM and is surrounded by mountains as high as 7000 ft. For this scenario, we have chosen the approach RNP AR Z RWY 08 (see Fig. 11), which follows the valley to the west of the airport. The FAF altitude is at 13,000 ft, located very close to the IAF position. The origin altitude and direction are, respectively, 1906 ft and 80.9 degrees.
- **Queenstown Airport (NZQN)**: this airport is located in a very challenging location at 1171 ft, close to the southern end of the New Zealand Southern Alps. Both to the west and the east of the airport high mountains can be found, ranging between 4000 ft and 8000 ft. Narrow corridors at lower altitudes in both runway directions allow for arrivals to the airport, via the Kawarau River from the east and Lake Wakatipu from the west. The RNP AR Y RWY 23 approach was selected for this scenario, depicted in Fig. 12. The FAF altitude is 3300 ft, while the origin altitude and direction are 1171 ft and 233.0 degrees, respectively. In this scenario, the ATKIL waypoint (Fig. 12) was set as the IAF.

A maximum number of iterations of 2000 was considered for all scenarios, while the sampling bias threshold was set to 40. We observed

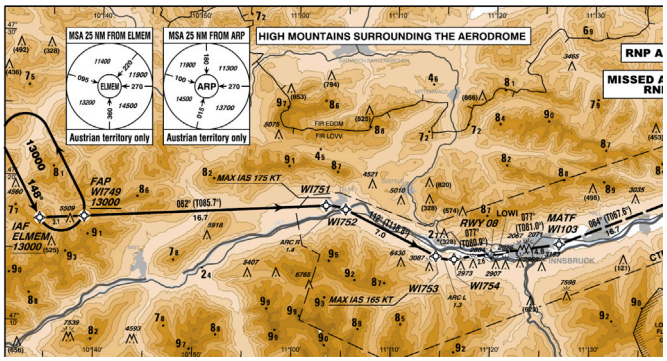


Fig. 11. Innsbruck Airport (LOWI) scenario: RNP AR Z RWY 08.
Source: Austria AIP.

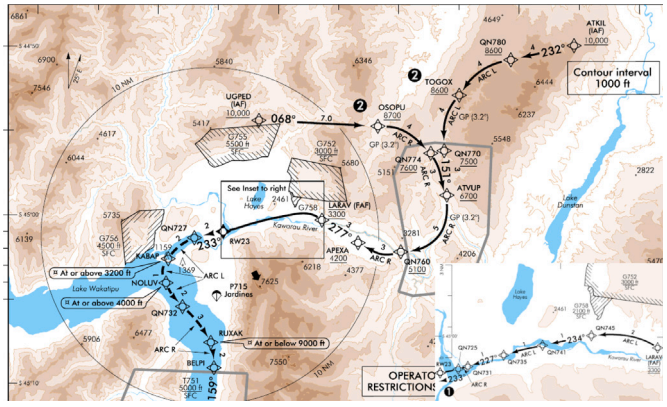


Fig. 12. Queenstown Airport (NZQN) scenario: RNP AR Y RWY 23.
Source: New Zealand AIP.

that this number of iterations was enough for successfully obtaining a path from the initial state to the goal state for all scenarios. Regarding the sampling bias, setting the threshold to 40 helped to obtain straighter paths in the goal direction.

In all cases, the path-generation process was terminated once the goal was reached. The process could continue beyond this point until the maximum number of iterations is reached, but the computational time drastically increases. Furthermore, we observed that only slightly better paths were obtained if allowing the algorithm to reach the maximum number of iterations. As a result, we decided to stop the process once the goal was reached in order to prioritize obtaining a solution in a reasonable amount of time.

The rest of the RRT* configuration parameters used for each of the scenarios are detailed in Table 3. As a reminder, w_t stands for the track change weight, which is used in the RRT* cost function as explained in Section 4.1.

The selection of parameters is inherently scenario-dependent. Rather than following a universal approach, we rely on expert judgment and a minimal trial-and-error process to fine-tune these parameters for each specific scenario. Due to the complexity and variability of the design environment, determining a single optimal set of parameters across all scenarios is impractical. Instead, we tailor the chosen values to the specific conditions of each case to ensure an operationally feasible solution. However, it is important to note that other parameter values could also yield valid, operationally feasible results. For instance, in the RJFT scenario, selecting a rewiring radius of 30 NM or 35 NM would have minimal effects on the generated paths. In Section 5.6, we give some details regarding the effect of these parameters on the framework's output.

Table 3
RRT* configuration parameters.

Scenario	Step size	Rewiring radius	w_t
RJFT	1 NM	30 NM	0.3
LOWI	1 NM	40 NM	0.3
NZQN	1 NM	15 NM	0.06

A step size value of 1 NM was chosen for all scenarios. In general, the scenarios chosen are quite challenging, and a step size of 1 NM was enough in order to allow the RRT* algorithm to generate a path capable of avoiding all the obstacles. The generated random sample states are closer to the existing tree, allowing for short turns to steer clear of obstacles when necessary. Larger step sizes, of more than 3 NM, although sometimes leading to a successful termination of the algorithm, increased substantially the computational time.

Regarding the rewiring radius, we enforced quite a high value in all cases. Although having a high rewiring radius increases the computational time—as every time a new sample state is generated more connections need to be tested with the rest of the nodes within the rewiring radius—we observed that this was one of the crucial parameters in order to ensure straighter paths. For the NZQN case, the rewiring radius was shorter due to the more complex nature of the scenario. In this case, having a longer rewiring radius unnecessarily increased the computational time without leading to better paths. This is due to the fact that, in this scenario, many turns are required over a short distance in order to avoid all the obstacles. Therefore, enforcing a longer rewiring radius would just lead to more iterations when testing feasible connections between the generated random sample state and the nodes in the tree. However, most of these iterations would not be feasible, as there would be many obstacles in the way. The effect of the rewiring radius is analyzed in Section 5.6.

Finally, the track change weight was the same for the RJFT and LOWI scenarios. Assigning a value to the track change weight ensured that the path would not involve unnecessary turns leading to non-operational flight procedures. In the NZQN scenario, as aforementioned, more turns were needed over a shorter distance. As a result, we needed to use a lower track change weight. Conversely, higher track angle weights could drastically increase the computational time of the framework: the algorithm would try to minimize the turns but, in this case, turns are strictly necessary to avoid the multiple obstacles.

The rest of the parameters used in our framework are described below:

- **Initial and goal states**: the RRT* algorithm was initialized with the runway location as its initial state, while the orientation of the runway was also used to determine the initial track of the generated path. For the goal state, the location of the initial approach fix (IAF) was employed.
- **IF and FAF**: in all scenarios, we employed the IF and FAF altitudes specified in the published charts to establish the starting and ending altitudes for each approach segment. It is important to note, however, that the specific locations of the IF and FAF (i.e., latitude and longitude) were not enforced. Therefore, the algorithm is responsible for determining these locations, which correspond to the points along the generated path where the altitude matches the IF and FAF altitudes from the approach chart, respectively. In future work, we aim to enhance our framework to autonomously determine these altitudes.
- **Stabilization segment**: for all scenarios, we imposed an initial constraint on the RRT* algorithm, forcing the generation of a straight segment within the FAS. This segment is aligned with the runway direction and has the same length as the one specified in the published charts. This constraint was crucial to prevent the RRT* algorithm from generating arbitrary points in terms of direction from the origin (i.e., the runway). By introducing this

constraint, we ensured the presence of a runway-aligned segment to facilitate the stabilization of arriving aircraft before landing. While the minimum length of this segment can be calculated in accordance with the ICAO RNP AR APCH procedure design guidelines [6], it is worth noting that in this paper we adopted the length specified in the published charts.

- **RNP navigation accuracy values, MOC and bank angle**: standard values as defined by ICAO [6] were applied to all scenarios.
- **Descent gradients**: contrary to the RNP navigation accuracy values, MOC values and bank angle values, the descent gradients used in the experiments conducted in this paper were not the standard ones for all scenarios. FAS descent gradients of 3.6 degrees and 3.2 degrees were used in LOWI and NZQN scenarios, respectively, while standard values were used for the initial and intermediate segments. These values were obtained from the respective approach charts, and were required due to the challenging nature of these scenarios, where the surrounding mountains made impossible to use FAS standard values in order to clear the obstacles. For RJFT, standard descent gradients values were used for all segments, as done in the currently published chart.
- **Leg types**: in this work, we only focused on RF legs due to their importance in representing curved flight paths with precision. TF legs, while frequently used together with RF legs to model turns, are not included in the current framework. Their integration is planned for future work to enhance the model's generalizability. Nevertheless, the RF leg-only approach has proven effective in generating feasible procedures that meet ICAO requirements in the scenarios considered.
- **Terrain-elevation data**: terrain information was retrieved from the topographic data generated from NASA's Shuttle Radar Topography Mission (SRTM) [38,39]. The SRTM payload flew aboard the Space Shuttle Endeavour during the STS-99 mission. SRTM collected topographic data over nearly 80% of Earth's land surfaces, creating the first-ever near-global dataset of land elevations. The topographic maps generated during this mission conform the digital elevation model (DEM) data used in this work, which were downloaded in digital terrain elevation data (DTED) format. This data has a sampling of 1 arc-second of latitude and longitude, which translates into elevation data with 30-meter intervals.

  In previous publications [9], we used DEM data provided by the Geospatial Information Authority of Japan [40]. This data has a gridded format with 5-m intervals, leading to a total number of points—including latitude, longitude and elevation information—of 43,605,000 for the Kumamoto scenario. However, this kind of data is mainly limited to Japan, has several regions which are not covered (e.g., military airspace) and its precision is too high, which leads to a drastic increase in the computational time. Still, it might be worth considering it in the future if more specific studies are performed in scenarios located in Japan.

While the algorithm requires parameter adjustments per scenario, this reflects the inherent variability in flight procedure design due to different terrain, obstacles, and constraints at each airport. These adjustments ensure compliance with ICAO requirements and operational soundness. Although flexibility is crucial in the current approach, automating these adjustments will be a focus of future work to further enhance efficiency.

Table 4 summarizes the source of the parameters used by our framework (note the list of parameters is not exhaustive), namely, the ICAO design rules for RNP AR APCH, the published charts or obtained as a result of our framework route-optimization process.

Table 4
Source of the framework parameters.

Parameter	Source
MOC	ICAO
Segment width	ICAO
Descent gradient	ICAO
Bank angle	ICAO
FAS Descent gradient	Chart
Stabilization segment length	Chart
FAF & IF altitude	Chart
Origin	Chart (i.e., RWY)
Goal	Chart (i.e., IAF)
FAF & IF location	Optimized
Route (i.e., lateral path)	Optimized

5.2. RNP AR APCH generation process: RJFT example

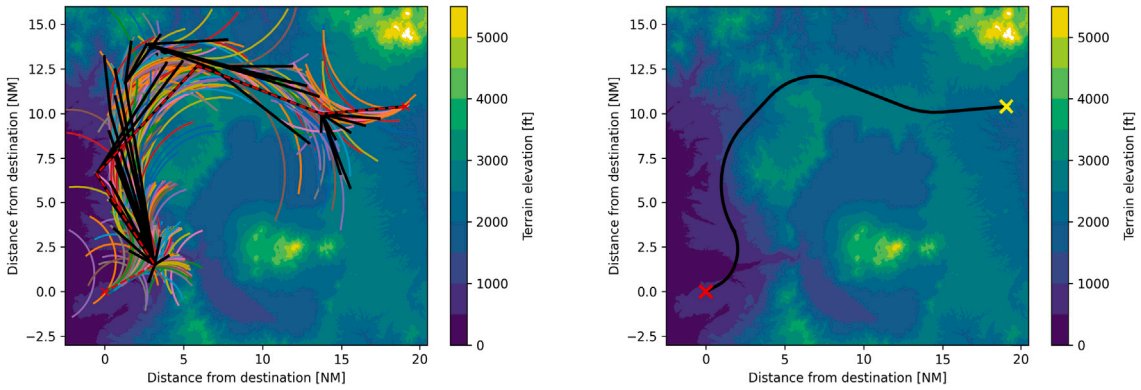
In this section, we describe the process that the framework follows in order to generate a free-obstacle path, focusing on the RJFT scenario. This process is depicted in Fig. 13. The elevation information, extracted from the DEM data, is represented with a color code, from the lower elevations in dark blue to the higher elevations in yellow. This color code will be used throughout the following sections when presenting the results for all the scenarios. Moreover, the horizontal and vertical axis represent, respectively, the horizontal and vertical distance from the geodetic coordinates in the DEM grid to the destination airport in nautical miles. This distance was computed by using the Vincenty formulae [41].

In Fig. 13(a), the black lines represent the RRT* resulting tree, with all the generated nodes and edges. The red-dashed line represents the output of the RRT* algorithm, and is composed of the edges and nodes of the RRT* tree that link the origin with the goal with the minimum cost for a specific number of iterations (recall that in this work the algorithm stops once the goal node is reached). We also show the several Dubins paths that the algorithm generates in each iteration during the steering and rewiring processes, which will be used to ultimately connect the RRT* tree nodes (i.e., the nodes integrating the black paths). As it can be observed, many Dubins curves are generated: the bigger the tree becomes, the more connections are tested during the rewiring process, which leads to this big number of Dubins curves. Furthermore, the bigger the rewiring radius, the more connections need to be tested between a new random sampled state and its neighboring nodes (within the rewiring radius). For each of the Dubins curves, the algorithm checks if all the constraints set by ICAO [6] are met. If they are met, the connection between the sampled state and a given node in the tree is established (leading to a new node and edge in the tree); if they are not met, the connection is not established. Still, for illustrative purposes, we have depicted all Dubins paths in Fig. 13(a), both the ones that lead to a new node and edge in the tree and those that do not. The colors used for depicting the multiple Dubins curves do not have any specific meaning; different colors were used with the only purpose of visualizing clearly the several curves that are needed while the algorithm is running.

In Fig. 13(b), we depict the final output of the framework: a free-obstacle path after the RRT* successfully terminates. The red cross represents the origin of the RRT* algorithm (i.e., the runway), while the yellow cross represents the goal node (i.e., the IAF). The nodes in the RRT* are connected with Dubins curves, which means that every time a track change occurs in the path, a Dubins curve will be generated. In a flight procedure, this translates into an RF leg. In the specific case shown in Fig. 13(b), the path is composed of 4 turns, so 4 Dubins curves are needed to connect the nodes.

5.3. Proposed RNP AR APCH procedures

In this section, we present the results for the scenarios introduced in Section 5.1. As previously explained, due to the random nature of



(a) Computation process: RRT* with Dubins

(b) Final output

Fig. 13. Route generation process for the Kumamoto Airport (RJFT) scenario.

the RRT* algorithm, each time we run our framework a different result is obtained. For each scenario, we ran our framework 5 times, thus, leading to 5 approach procedures per scenario.

In all the plots presented in this section, the origin of the RRT* algorithm, which corresponds to the runway, is depicted as a big red cross, while the goal, corresponding to the IAF, is depicted as a big yellow cross. Both the runway and the IAF, as specified in Table 4, are obtained from the published approach charts.

The published approach procedure, as shown in the airport approach charts for each scenario, is depicted in black, while its waypoints are depicted as small red crosses. Several colors are used to depict each of the approach procedures computed by the framework, allowing for a direct comparison as all procedures are visualized in the same plot.

The white areas that can be observed in some regions of the LOWI and NZQN scenarios (Figs. 15 and 16) are a normal behavior that has been widely observed in SRTM DEM data. This usually happens in mountainous regions due to various reasons: steep terrain or tall mountains can create shadows that radar signals cannot penetrate or accurately measure and, also in this kind of rugged landscapes, radar signals might be obstructed by tall features such as cliffs, large boulders or dense vegetation, resulting in areas where data is not captured. Other reasons like radar interference or data processing issues (e.g., due to anomalies in the radar returns) could also lead to these blank areas. In order to mitigate these issues, various techniques such as data fusion with other sources, interpolation, or specialized processing methods could be employed in the future to fill in the missing data or improve the accuracy of elevation models in these regions. Still, this is not a big issue for the scenarios tackled in this paper.

The results for each scenario are explained below:

- **Kumamoto Airport (RJFT)**: the approach procedures generated for the RJFT scenario are depicted in Fig. 14. While each execution of the RRT* algorithm yielded different results, the abundance of constraining factors imposed by the mountainous terrain consistently resulted in a high similarity between the approach procedures generated.
  In general, starting from the IAF (i.e., yellow cross), the approach routes first follow a westward direction, following the caldera north rim. This direction is kept until some of the highest mountains of the caldera rim are surpassed. After that, the routes turn to continue with a southward direction, keeping a safe distance from the high mountains of the caldera west rim. A final turn is performed in order to align with the airport runway.
  All routes proposed by the framework are very similar to the one in the published chart, although some of them, specially the green and purple routes, are located slightly closer to the mountains of the caldera west rim than the route published. Still, all the constraints regarding the obstacle clearance are met.
  The approach routes computed by our framework are composed of three to four RF legs and their total lengths range between 30.2 NM for the shortest route and 33 NM for the longest route. On the other hand, the length of the published route is 31.9 NM.
  Nevertheless, the priority of the RNP AR APCH route generation process presented in this paper is not to obtain the shortest possible route; instead, it is more important to obtain operationally feasible routes (e.g., minimizing the number of undesired turns) that avoid all the obstacles on the way.
  In this study, the scenarios were chosen to validate the performance of our framework against existing published approach procedures. The fact that the framework generated results are similar to the published procedure in this scenario is indeed a positive outcome. It demonstrates the framework's ability to produce high-quality, ICAO-compliant procedures in a fraction of the time required by traditional manual design efforts.
  The key advantage of our framework is its ability to automate the procedure design process, saving significant time for flight procedure designers. This is especially beneficial in more complex environments, where terrain, obstacles, or airspace constraints make it challenging to manually identify the optimal procedure. In such cases, the framework can efficiently generate safe and

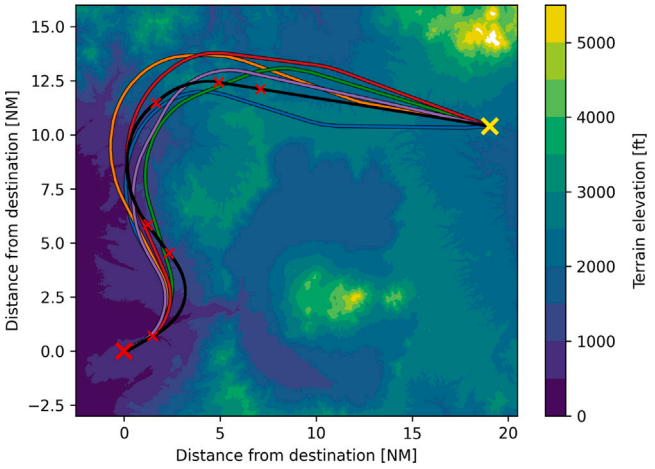


Fig. 14. Comparison of computed approach procedures and published procedure for Kumamoto Airport (RJFT) scenario.

operationally sound flight procedures, potentially identifying solutions that may not be obvious through manual design efforts. Moreover, the flexibility of the framework makes it well-suited for the design of next-generation procedures, such as RNP to xLS (e.g., ILS) transitions. This capability enables the framework to support evolving operational needs, offering an adaptable tool for future flight procedure development.

- **Innsbruck Airport (LOWI)**: the 5 approach routes generated for this scenario, depicted in Fig. 15, are all very similar. Starting from the IAF, the routes first follow an eastward direction seeking lower altitudes over the valley, and avoiding some 6000 ft mountains on the way. Once this is achieved, the routes follow the valley in the airport direction, until a final turn is performed in order to align with the runway.
  As it can be observed, and similarly to what was observed for some of the routes in the RJFT scenario, the routes computed by our framework are closer to the mountains than the route published. Generally, the algorithm does not have a problem in generating this kind of routes, as far as all the constraints are met. However, in the future, it might be interesting to add some additional constraints to the algorithm in order to avoid ending up with this kind of routes that are unnecessarily too close to the mountains. Even if it potentially means flying a longer distance, in this specific case it might be better to fly a bit further from the mountains, and stay in the middle of the valley. Other considerations like noise effects over populated areas should also be considered in the future when adding these additional constraints.
  In this scenario, both the distance of the published route and the computed route were very similar, around 32 NM. Furthermore, in both cases 2 RF legs are employed.
- **Queenstown Airport (NZQN)**: in this scenario, several behaviors were observed for each of the approach routes computed by our framework, as depicted in Fig. 16.
  Starting from the IAF, the blue and orange routes follow a westward and south-westward directions, respectively. Both keep a safe distance from the mountains in the area, especially the 6346 ft mountain to the west of the IAF (Fig. 12). Once these mountains are avoided, both routes turn to the south in direction of the airport, finishing with a turn to the west to align with the runway.
  On the other hand, the purple, red and green routes follow a southward direction from the IAF. The high mountains at the beginning do not suppose a safety issue due to the high altitude of the aircraft at the IAF. All routes continue south until reaching the valley, after which they turn to the west to finally align with the airport runway.
  In this scenario, the routes differ with respect to the published route. The RRT* algorithm was not able to find a path via the corridor where the ATVUP waypoint in the published route is located (Fig. 12). Instead, the paths generated followed other directions in order to avoid the obstacles.
  This was the most challenging scenario for the framework, with many obstacles and not many possibilities to avoid them. Several RF turns were needed to avoid these obstacles, between 3 and 6 depending on the route. The distance of the routes is between 27 NM for the shortest route and 33 NM for the longest one. The length of the published route is 31.8 NM.
  Although the obtained routes do not resemble the one published in the chart, we could consider the results to be positive: as aforementioned, the goal of this study is to compute multiple feasible routes that could be later refined by flight procedure designers. We are not aiming to obtain the optimal route; we are rather interested in finding several potential routes which could potentially represent viable options not previously contemplated by flight procedure designers, potentially enabling the improvement of the published flight procedure.

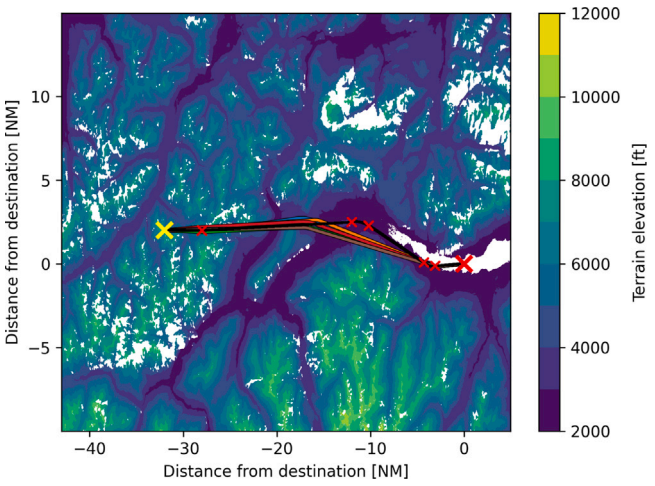


Fig. 15. Comparison of computed approach procedures and published procedure for Innsbruck Airport (LOWI) scenario.

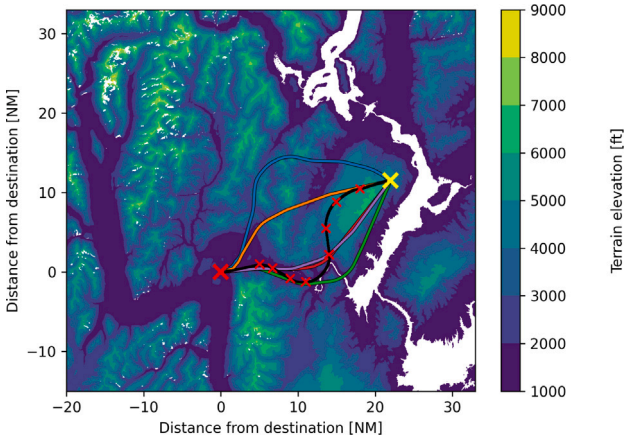


Fig. 16. Comparison of computed approach procedures and published procedure for Queenstown Airport (NZQN) scenario.

5.4. Computational effort

All experiments were conducted in a laptop with Ubuntu 22.04 LTS, with an Intel Core i7-1280P processor and 32 GB of memory. The code for the proposed framework was developed using Python version 3.10.12. In all scenarios, the computational time ranged from 2 to 15 min, varying with each run.

It is worth highlighting that the RRT* algorithm typically takes longer than the non-optimal version (RRT) due to its additional optimization steps aimed at finding optimal paths. These steps involve rewiring the tree structure and evaluating nearby nodes for potential improvements in path cost. Additionally, the RRT* algorithm requires more parameter tuning and sampling biases to achieve its optimization goals, leading to increased computational complexity and runtime. Consequently, the obstacle-avoidance function needs to be run multiple times during the path generation process, which, together with the fact that we have very precise DEM data, leads to an increased computational effort.

Because of the random nature of the algorithm, there are instances where the sampled states enable easy avoidance of obstacles, leading to a significant reduction in computational time. However, in other cases, numerous iterations may be required to identify a path free of obstacles. This was specially noticeable in the NZQN scenario, in which many iterations were needed to successfully terminate the optimization

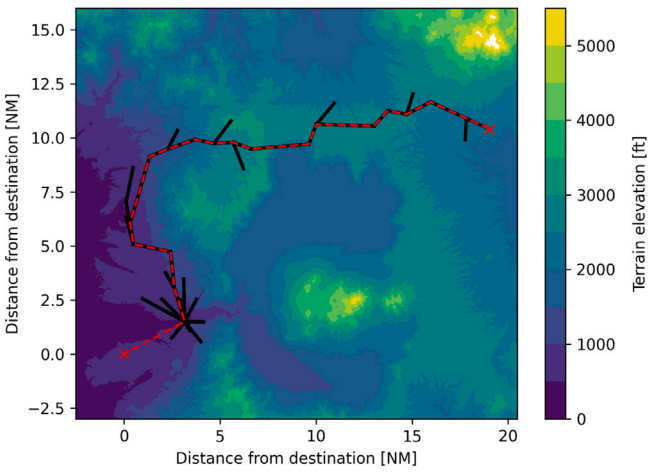


Fig. 17. Effects of a rewiring radius equal to 5 NM in the RJFT scenario.

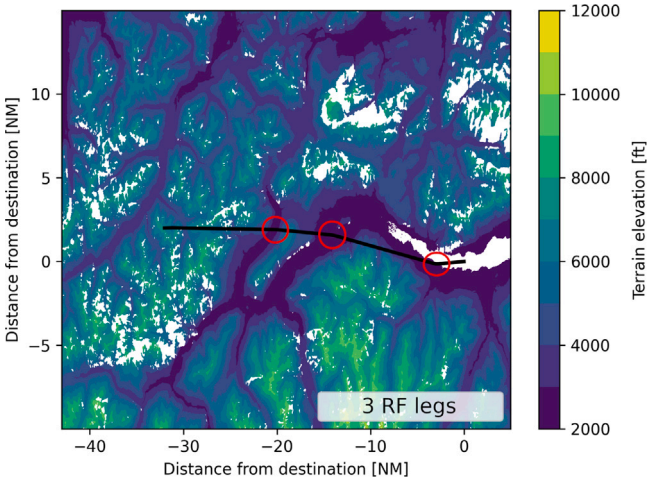


Fig. 18. Effects of not considering a track change weight (w_t) in the cost function in the LOWI scenario.

process, usually around 1500 iterations. On the other hand, the RJFT and LOWI scenarios usually required a lower number of iterations to reach the goal, most of the time less than 1000 iterations.

It is also worth noting the fact that in some isolated cases the algorithm did not converge after reaching the maximum number of iterations. In such cases, the algorithm automatically started again after reaching the maximum number of iterations. In the future, some mechanisms could be created in order to stop the path-generation process in those cases in which no progress—i.e., the path is not getting closer to the goal—is observed after a given number of iterations.

By using more sophisticated methodologies it would be possible to enhance the computational speed and optimize the RRT* search process. For instance, the Gaussian mixture model proposed in [25] aims to expedite convergence by reducing the search space. Furthermore, the implementation of alternative versions of the RRT* algorithm, such as a bi-directional RRT* (which generates paths from both the origin and goal), could also be explored.

Furthermore, parallelizing the computation process, enabling to generate several routes at the same time for any given scenario, would be an easier an equally effective option to implement.

Still, approach procedures were generated in an acceptable amount of time for the purpose of this work. Currently, flight procedures are still designed manually by using dedicated flight procedure design software. Thus, generating a feasible flight procedure in 2 to 15 min

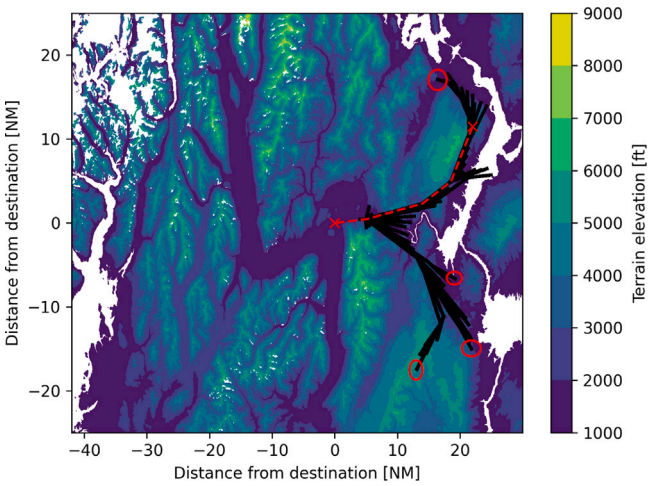


Fig. 19. Effects of terminating the process when reaching the maximum number of iterations (800) in the NZQN scenario.

supposes a clear advantage in the flight-procedure-design process. Currently, we are in contact with flight procedure designers from the Japan Civil Aviation Bureau (JCAB), who showed interest in the results of our framework. In future work, we are planning to provide them with our generated flight procedures so that they can validate them with PANADES (i.e., the flight procedure design software used in JCAB). The current computational times obtained with our framework would enable us to send them not only 1 procedure per scenario, but several procedures that could be tested and refined in PANADES.

5.5. Comparison with a genetic algorithm approach

There are limited prior studies on generating flight procedures that comply with ICAO constraints. However, Hasegawa et al. [33] proposed a path-planning generation framework based on GAs that was capable of generating RNP AR APCH paths. In this study, we instead utilized RRT* as the core algorithm, considering its strengths and weaknesses for flight procedure design.

While GA is a global optimization algorithm with the potential to achieve higher optimality than RRT*, it comes with significant drawbacks in terms of computational time. GAs can easily incorporate both equality and inequality constraints, such as setting a minimum length for a particular flight leg. However, they require substantial computation time, particularly when dealing with constraints that demand high processing power, such as terrain considerations. For instance, in the case of RJFT, the RRT*-based algorithm can find a solution within minutes, whereas the GA-based approach may take several days.

Moreover, GAs require manual input to define the number and types of legs and to specify which waypoints serve as IAF, IF, and FAF prior to computation. For example, in the case of RJFT, the sequence would need to be pre-determined as (IAF)-TF-(IF)-TF-RF-RF-(FAF)-TF-RF-TF. In contrast, the RRT*-based algorithm does not require such predefined inputs.

As aforementioned, flight procedure designers typically prioritize tools that can quickly generate feasible flight procedures over those offering highly optimal solutions. Therefore, RRT* is deemed more suitable than GA for addressing the practical needs of designers.

5.6. Effect of RRT* parameters

In this section, we show the effect that some of the RRT* configuration parameters can have on the framework output and on the computational time. We do not intend to present a comprehensive sensitivity analysis; instead, the aim of this section is just to give

some insights about these parameters and analyze how can they affect the results, including recommendations on how to configure them in order to successfully obtain free-obstacle paths. Still, it is important to highlight the fact that the effect of the parameters analyzed in this section might depend on each RRT* run and on the scenario. In the following list, we will discuss the effect that each parameter can have on the solution:

- **Rewiring radius**: Fig. 17 depicts the resulting path if a rewiring radius of 5 NM is used for the RJFT scenario. As discussed in previous sections, each time a new random state is sampled, connections between this state and neighboring nodes in the tree within the rewiring radius are tested.
  If the rewiring radius is small, like in this case, the number of neighboring nodes is reduced. As a result, the algorithm cannot test connections with nodes that are located further and that could potentially lead to straighter paths. Therefore, in general, the smaller the rewiring radius the less straight the paths will be.
  In the specific case depicted in Fig. 17, we can observe several turns. It was impossible to generate RF legs in most of the turns (i.e., Dubins paths cannot be generated by the algorithm), as they are too sharp/short and they would violate the aircraft turn constraints. In this situation, TF turns could potentially be flown following an assessment of the protection areas. Note, however, that even TF turns might not be possible either due to the potential violation of the RNP AR APCH design constraints. In such a case, the only solution would be to increase the rewiring radius to obtain a feasible solution. Still, no matter if TF turns were possible, it is not acceptable from an operational point of view to fly such a kind of procedure due to the great number of turns involved.
  Most of the time, although it can slightly increase the computational time (as more connections need to be tested), it is preferable to use higher values for the rewiring radius. A value between 15 NM and 40 NM was deemed adequate for the scenarios considered in this paper. For scenarios involving longer approach procedures, longer rewiring radii might be needed.
- **Cost function weight**: the weight used in the cost function also has an effect on the results. Fig. 18 depicts the resulting path in the LOWI scenario when no track change weight is considered (i.e., w_t equal to 0). As it can be observed, the number of turns in this case is three (highlighted in red), while the number of turns for all the paths presented in Section 5.3 in Fig. 15 (with a w_t equal to 0.15) was two. Therefore, it is evident that considering a significant enough track angle weight in the cost function helps to obtain straighter and, thus, paths more acceptable at an operational level.
- **Sampling bias and step size**: in general, changing the sampling bias towards the goal and step size did not have major effects on the resulting path shape. Just in some isolated cases we noticed that having a too low sampling bias could lead to less straight paths. However, modifying these values led to some changes in the computational times.
  The framework was run 50 times for each value and several statistical indicators were obtained, as shown in Tables 5 and 6, which show, respectively, the results for the sampling bias and step size. The average computational time, standard deviation (SD) and 95% confidence interval (CI, assuming a normal distribution) are shown in these tables. We used the RJFT scenario to conduct this analysis.
  We did not observe major changes in the computational time for a sampling bias between 20 and 60. However, as aforementioned, lower sampling bias values could risk ending up with less straight paths. On the other hand, for a sampling bias of 80, the computational time drastically increases. This might be due to the fact that if the majority of the sampling is directed towards the goal region, especially in situations with complex obstacle configurations like the one in RJFT, the algorithm might get trapped in local minima. Biasing the sampling excessively towards the goal can limit exploration around obstacles, leading to suboptimal paths or getting stuck in constrained spaces. Focusing sampling around the goal might ignore potential paths or routes that could circumvent obstacles.
  Regarding the step size, generally smaller values lead to longer computational times, as more nodes are generated, which leads to more operations during the rewiring process. As the step size increases, the computational time is also reduced. However, the gains are not that remarkable. Furthermore, depending on the scenario, having too large step sizes could also risk leading the algorithm towards dead-end paths or local minima. As a result, the algorithm would spend more time backtracking or re-exploring to find alternative routes, increasing the overall computation time. Conversely, smaller step sizes allow for finer and more precise exploration, enabling the algorithm to navigate narrow passages or intricate spaces more effectively.
  Finally, the standard deviations for both the sampling bias and step size effects on computation time are relatively low, each less than 20 s. This results in a narrow 95% confidence interval (CI), indicating low uncertainty. However, depending on the specific scenario in which our framework is run, different results may be obtained.
- **Maximum number of iterations**: another of the parameters that can be tuned in the RRT* algorithm is the maximum number of iterations. Not only this number can be changed, but also it can be chosen whether the algorithm terminates once the goal is reached or once the maximum number of iterations is reached.
  Fig. 19 depicts the framework output for the NZQN scenario in a situation in which the algorithm terminates once the maximum number of iterations is reached. In this specific case, this number was set to 800 iterations.
  In general, allowing the algorithm to continue after reaching the goal leads to better paths. However, this comes at the expense of an increase in the computational time. Furthermore, we have observed that most of the time these gains in the path quality are not very significant. In the particular case depicted in Fig. 19, the goal was reached after 250 iterations. Then, the algorithm continued until reaching 800 iterations.
  In this specific case, the quality of the path from the origin to the goal was not improved even after allowing the algorithm to run for extra iterations. As aforementioned, this was usually the case in the other scenarios as well, where only some small improvements were observed in some runs.
  Even if the path towards the goal is not significantly improved, allowing the algorithm to run for more iterations could help to find potential IAFs in a given scenario. In Fig. 19, these potential IAFs are highlighted in red. This feature could be useful to design new approach procedures or modify some of the existing ones for a given airport. Still, the definition of the IAFs location would depend on the specific scenario, conflicting traffic flows, populated areas, etc.

Table 5
Sampling bias effect on the computation time [minutes].

Sampling bias [–]	Average	SD	95% CI
20	3.81	0.20	(3.75, 3.87)
40	4.68	0.15	(4.63, 4.73)
60	4.54	0.10	(4.50, 4.58)
80	8.17	0.24	(8.09, 8.25)

Table 6
Step size effect on the computation time [minutes].

Step size [NM]	Average	SD	95% CI
0.5	7.40	0.23	(7.31, 7.49)
1	5.12	0.16	(5.05, 5.19)
2	4.56	0.21	(4.45, 4.67)
3	4.31	0.13	(4.25, 4.37)

5.7. Framework integration in the flight procedure design process

The potential role of our framework in the flight procedure design process was discussed with flight procedure designers from the Japan Civil Aviation Bureau (JCAB), who provided valuable insights into its application in their design workflow.

The integration of our framework into the flight procedure design process offers potential improvements in both efficiency and accuracy, particularly during the early stages of the design process. Currently, flight procedure designers rely on an iterative process that begins with generating multiple design concepts, followed by detailed evaluations and refinements using specialized tools such as PANADES. This approach, while thorough, is time-consuming, especially during the preliminary stages, where a broad range of possible solutions must be explored before narrowing down to viable candidates.

By automating the initial route generation process, the proposed tool can assist designers in quickly producing multiple potential routes that comply with key constraints such as terrain, obstacle clearance, and airspace structure. This not only speeds up the early stages of design but also provides a systematic approach to exploring a larger solution space, which would be difficult to achieve manually. Importantly, the tool incorporates the use of RRT* algorithms combined with Dubins paths, which ensures the generated procedures are continuous and account for aircraft dynamics, a notable novelty that aligns well with the need for efficient trajectory planning in constrained environments like RNP approaches.

5.7.1. Early stage utility in concept generation

In practice, the tool's strength lies in automating the preliminary phase of design—what procedure designers often refer to as the "concept generation" stage. Here, the tool can generate an initial set of route options that adhere to constraints such as terrain avoidance and lateral airspace requirements. The ability to automate this phase could alleviate the burden on designers, allowing them to focus their attention on evaluating and refining a smaller, more manageable set of options rather than manually crafting each candidate from scratch.

For instance, designers noted that the tool could reduce the time spent in this early phase by quickly generating paths that account for essential PBN criteria, as well as the geometry of constrained airspaces such as those around obstacles or restricted areas. This capability not only accelerates the process but also improves the consistency of generated solutions by leveraging the computational precision of the tool, potentially reducing human error in these initial drafts.

5.7.2. Limitations in later stage design

Despite its advantages, our framework is not intended to replace the comprehensive evaluations required in later stages of flight procedure design. For more advanced stages, designers must turn to detailed assessments using tools such as PANADES, where terrain clearance and obstacle clearance altitude (OCA) must be calculated with high precision. These later evaluations require deep integration with airspace data, environmental considerations, and adherence to specific regulatory standards, which our current framework does not fully address. As such, our framework is best suited as a preliminary aid, offering rapid generation of feasible solutions that will eventually undergo further refinement and verification.

Designers emphasized that while our framework is valuable for generating initial concepts, the complexities of real-world airspace, aircraft performance, and safety regulations necessitate a more nuanced approach in later stages. For example, in areas with dense air traffic or challenging terrain, the automatically generated routes may still need significant adjustments to meet all operational and regulatory requirements.

Still, our framework could potentially incorporate additional constraints through iterative collaboration with flight procedure designers, especially in cases where local authorities impose specific requirements. For instance, constraints such as noise-sensitive areas or restricted zones could be integrated into the generated routes based on feedback from designers working in specific geographic contexts. Although these constraints are not currently included in our framework, they could be added during the iterative process to meet the localized needs of particular airspace environments

5.7.3. Complementary role in the overall workflow

Overall, our framework could function as a complementary resource to the existing flight procedure design process rather than a standalone solution. Its primary contribution would be in automating the exploration of potential paths that adhere to ICAO constraints, freeing up time and resources for designers to engage in more detailed evaluations. This approach allows designers to more efficiently iterate through the early phases of design while keeping the precision required in later stages, where the final procedure must meet strict regulatory and safety standards.

By facilitating quicker concept generation and providing a systematic method for exploring a wide range of solutions, the tool could help bridge the gap between the time-consuming manual work of early design and the more complex evaluations of the final stages. In doing so, it stands to improve the overall workflow, making the design process more efficient and effective while maintaining the high standards required for safe and reliable flight procedures.

6. Conclusions

This paper introduced an automated framework designed for generating aircraft RNP AR APCH procedures using the RRT* algorithm. The nodes within the tree structure are connected using Dubins curves, accounting for the aircraft's turn radius constraints. Moreover, the algorithm cost function was tuned in order to obtain operationally sound approach procedures. The set of constraints defined by ICAO for RNP AR APCH procedure design was also considered by the framework.

In order to increase the acceptability of the flight procedures at an operational level and to extend the applicability of our solution to a wider range of scenarios, we intend to investigate additional features to incorporate into the framework. Firstly, noise-abatement procedures could be considered, which is a crucial aspect in airports located close to highly-populated areas, such as San Francisco International Airport. In addition, other traffic flows and airspaces could be considered when running the framework. All these elements could translate into additional constraints in the formulation and potential changes in the cost function, leading to a multi-objective optimization.

Other potential improvements would involve the generation of TF legs when considering turns in the procedure, as currently only RF legs are considered. Still, the aim of this study is not to replace the role of the flight procedure designer. Thus, at the current stage of our work, some of the RF legs generated by the framework could easily be replaced by TF legs following an assessment of the turn constraints (e.g., protection areas, obstacle clearance, ...) with a flight procedure design software.

We are also planning to add more automation features to the framework in the future: instead of obtaining the values for the RNP AR APCH design parameters (e.g., descent gradients, protection area widths, etc.) directly from the charts, we aim at automatically finding

those values. Nevertheless, the values published in the charts (or the standard values recommended by ICAO) could still be used as an initial guess in a process aiming at finding the optimal value for the RNP AR APCH design parameters. A similar process could be followed in order to choose the RRT* configuration parameters.

Although some manual adjustments are required, the framework already includes significant automation for generating ICAO-compliant procedures, such as automated obstacle clearance checks. These features enable safe, operationally feasible procedures with minimal intervention. The current manual tuning of some parameters reflects the variability between airports, but the framework is designed to adapt automatically to different environments. Future work will focus on further automating parameter tuning to enhance scalability and generalizability.

The significance of having an automated tool cannot be overstated, especially in complex scenarios like the ones presented in this paper, in which it is very difficult to define an approach procedure manually. Our framework's ability to automatically generate approach procedures becomes invaluable in such cases, representing a valuable asset for flight procedure designers, who could refine the framework's output using dedicated flight-procedure design software.

In future work, we intend to collaborate closely with flight procedure designers to validate our framework across various challenging scenarios. These collaborations hold the potential to introduce new constraints into the formulation, aligning with the preferences of flight procedure designers. Furthermore, it is worth noting the fact that the framework developed in this work could be easily extended to generate not only RNP AR APCH procedures, but also other kinds of procedures involving different sets of constraints.

CRediT authorship contribution statement

Raúl Sáez: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Daichi Toratani:** Writing – review & editing, Supervision, Resources, Conceptualization. **Ryota Mori:** Writing – review & editing, Conceptualization. **Xavier Prats:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M.E. Hampton, Office of Inspector General Audit Report: FAA Faces Significant Obstacles in Advancing the Implementation and Use of Performance-Based Navigation Procedures, U.S. Department of Transportation, 2014.

[2] New Zealand Aeronautical Information Publication, Queenstown airport approach chart: RNP Y RWY 05 (AR), 2023, https://www.aip.net.nz/. (Accessed 15 December 2023).

[3] USA Aeronautical Information Publication, John F. Kennedy international airport approach chart: RNAV(RNP) Z RWY 04L, 2023, https://skyvector.com/files/tpp/2312/pdf/00610RRZ4L.PDF. (Accessed 15 December 2023).

[4] USA Aeronautical Information Publication, San francisco international airport approach chart: RNAV(RNP) Y RWY 28R, 2023, https://skyvector.com/files/tpp/2312/pdf/00375RRY28R.PDF. (Accessed 15 December 2023).

[5] ICAO, Procedures for Air Navigation Services: Aircraft Operations. Doc 8168, sixth ed., ICAO, 2018.

[6] ICAO, Required Navigation Performance Authorization Required (RNP AR) Procedure Design Manual. Doc 9905, second ed., ICAO, 2016.

[7] D. Toratani, R. Mori, RNP AR approach route optimization using a genetic algorithm, in: 2022 Integrated Communication, Navigation and Surveillance Conference (ICNS), IEEE, Dulles, VA, USA, 2022.

[8] R. Sáez, H. Khaledian, X. Prats, A. Guitart, D. Delahaye, E. Feron, A fast and flexible emergency trajectory generator: Enhancing emergency geometric planning with aircraft dynamics, in: Proceedings of the 14th USA/Europe Air Traffic Management Research and Development Seminar, Eurocontrol and FAA, Virtual, 2021.

[9] R. Sáez, D. Toratani, R. Mori, X. Prats, Generation of RNP approach flight procedures with an RRT* path-planning algorithm, in: 42nd IEEE/AIAA Digital Avionics Systems Conference, DASC, IEEE, Barcelona, Spain, 2023.

[10] NTT Data, PANADES, 2023, http://www.airpalette.net/panades. (Accessed 18 May 2023).

[11] CGX Aero, Instrument flight procedure design, 2023, https://www.cgx-group.aero/products-ifpdesign. (Accessed 18 May 2023).

[12] R.M. Unkelbach, T. Dautermann, Development and evaluation of an RNP AR approach proceudre under tight airspace constraints, CEAS Aeronaut. J. 13 (2022) 613–625.

[13] S.M. LaValle, Rapidly-exploring random trees : a new tool for path planning, Annu. Res. Rep. (1998).

[14] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846—894.

[15] L. Janson, E. Schmerling, A. Clark, M. Pavone, Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions, 2015, arXiv:1306.3532 [cs] URL http://arxiv.org/abs/1306.3532.

[16] E. Andrés, D. González-Arribas, M. Soler, M. Kamgarpour, M. Sanjurjo-Rivo, Informed scenario-based RRT* for aircraft trajectory planning under ensemble forecasting of thunderstorms, Transp. Res. C 129 (103232) (2021).

[17] X. Qiu, Y. Li, R. Jin, Z. Zhao, J. Li, D. Lu, L. Ma, Improved F-RRT* algorithm for flight-path optimization in hazardous weather, Int. J. Aeros. Eng. 2022 (1166968) (2022).

[18] Z. Wei, L. Liu, T. Long, Z. Wang, RRT*-based threat-avoidance trajectory planning for aircrafts (IEEE/CSAA GNCC), in: 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), IEEE, Xiamen, China, 2018.

[19] D. Candemir, I. Kök, S. Özdemir, Narrowed regions-based bidirectional path planning using RRT-connect for single aircraft missions, Procedia Comput. Sci. 231 (2024) 703–708.

[20] X. Meng, N. Wang, Q. Liu, Aircraft parking trajectory planning in semistructured environment based on kinodynamic safety RRT*, Math. Probl. Eng. 2021 (1) (2021) 3872248.

[21] A. Fallast, B. Messnarz, Automated trajectory generation and airport selection for an emergency landing procedure of a CS23 aircraft, CEAS Aeronaut. J. 8 (3) (2017) 481–492.

[22] P. Pharpatara, B. Hérissé, Y. Bestaoui, 3-D trajectory planning of aerial vehicles using RRT*, IEEE Trans. Control Syst. Technol. 25 (3) (2017) 1116–1123.

[23] D.J. Webb, J. van den Berg, Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints, 2012, arXiv:1205.5088 [cs]URL http://arxiv.org/abs/1205.5088.

[24] A. Khanal, RRT and RRT* using vehicle dynamics, 2022, arXiv:2206.10533 [cs]URL http://arxiv.org/abs/2206.10533.

[25] P. Sharma, A. Gupta, D. Ghosh, V. Honkote, G. Nandakumar, D. Ghose, PG-RRT: A Gaussian mixture model driven, kinematically constrained bi-directional RRT for robot path planning, in: Procceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, Prague, Czech Republic, 2021.

[26] S. Yang, Y. Lin, Development of an improved rapidly exploring random trees algorithm for static obstacle avoidance in autonomous vehicles, Sensors 21 (6) (2021) 2244.

[27] B. Shi, P. Cheng, N. Cheng, 3D flight path planning based on RRTs for RNP requirements, in: Procceedings of the IEEE International Conference on Information and Automation, IEEE, Shenyang, China, 2012.

[28] H. Visser, Generic and site-specific criteria in the optimization of noise abatement trajectories, Transp. Res. D 10 (5) (2005) 405–419.

[29] X. Prats, V. Puig, J. Quevedo, A multi-objective optimization strategy for designing aircraft noise abatement procedures: case study at Girona airport, Transp. Res. D 16 (1) (2011) 31–41.

[30] S. Hartjes, H. Visser, Efficient trajectory parameterization for environmental optimization of departure flight paths using a genetic algorithm, Proc. Inst. Mech. Eng. G: J. Aeros. Eng. 231 (6) (2017) 1115–1123.

[31] V. Ho-Huu, S. Hartjes, H. Visser, R. Curran, An efficient application of the MOEA/D algorithm for designing noise abatement departure trajectories, Aerospace 4 (4) (2017) 54.

[32] S. Khardi, L. Abdallah, Optimization approaches of aircraft flight path reducing noise: comparison of modeling methods, Appl. Acoust. 73 (4) (2012) 291–301.

[33] T. Hasegawa, T. Tsuchiya, R. Mori, Optimization of approach trajectory considering the constraints imposed on flight procedure design, Procedia Eng. 99 (2015) 259–267.

[34] J. Chevalier, Departure and Arrival Route Optimisation Approaching Big Airports (Ph.D. thesis), Université Paul Sabatier, 2020.

[35] Federal Aviation Administration (FAA), United States of america aeronautical information publication (AIP), 2023.
[36] L.E. Dubins, On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, Amer. J. Math. 79 (3) (1957) 497–516.
[37] G. Satyanarayana, D.C. Manyam, A.L. Von Moll, Z. Fuchs, Shortest dubins path to a circle, in: AIAA Scitech 2019 Forum, San Diego, California, USA, 2019.
[38] NASA, Shuttle radar topography mission, 2023, https://www2.jpl.nasa.gov/srtm/. (Accessed 15 August 2023).
[39] United States Geological Survey (USGS), USGS earth explorer, 2023, https://earthexplorer.usgs.gov/. (Accessed 15 August 2023).
[40] Geospatial Information Authority of Japan, Base geospatial information download service, 2020, https://fgd.gsi.go.jp/download/menu.php. (Accessed 15 February 2023).
[41] T. Vincenty, Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations, Surv. Rev. 23 (176) (1975) 88–93.