CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Measurement

Master's thesis

Multi-failure Risk-aware Trajectory Planning for Urban Air Mobility

Bc. Jáchym Herynek

Supervisor: Ing. Jakub Sláma

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Herynek Jáchym** Personal ID number: **483733**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Computer Engineering**

II. Master’s thesis details

Master’s thesis title in English:

Multi-failure Risk-aware Trajectory Planning for Urban Air Mobility

Master’s thesis title in Czech:

Rizikové plánování trajektorií pro městskou leteckou mobilitu s uvažováním více poruch

Guidelines:

1. Study the dynamic model of a gliding aircraft [1].
2. Get familiar with the algorithm for determining the minimum safe altitude for a gliding aircraft [2] and risk-aware trajectory planning [3].
3. Based on safety reports [4, 5], determine the suitable failures to be addressed and propose an appropriate multi-failure aircraft model.
4. Propose and implement a solution for s risk assessment of in-flight failure.
5. Employ the developed assessment and proposed multi-failure aircraft model in risk-aware trajectory planning.
6. Computationally evaluate the proposed solution and assess its performance.

Bibliography / sources:

[1] Beard, R. W., & McLain, T. W. (2012). Small unmanned aircraft: Theory and practice. Princeton university press.
[2] Váňa, P., Sláma, J., Faigl, J., & Pačes, P. (2018). Any-time trajectory planning for safe emergency landing. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5691-5696.
[3] Sláma, J., Váňa, P., and Faigl, J. (2021). Risk-aware Trajectory Planning in Urban Environments with Safe Emergency Landing Guarantee, IEEE International Conference on Automation Science and Engineering (CASE), pp. 1606-1612.
[4] Air Safety Institute. (2015). 27th Joseph T. Nall Report.
[5] Australian Transport Safety Bureau (2016). Engine failures and malfunctions in light aeroplanes 2009 - 2014, Investigation number AR-2013-107, pp. 1–38.

Name and workplace of master’s thesis supervisor:

Ing. Jakub Sláma Department of Computer Science FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **17.02.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until:
by the end of summer semester 2023/2024

Ing. Jakub Sláma	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I declare that the presented work was developed independently ant that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethiclal principles in the preparation of university theses.

Prague, May 26, 2023

. . . . . .
Bc. Jáchym Herynek

Acknowledgement

I would like to thank my supervisor Ing. Jakub Sláma, for all his invaluable feedback, expertise, and assistance throughout this work. His patience has been remarkable. I would also like to thank my friends, their help and support has been more important than I realized at the time. Lastly, I have to acknowledge the support my family provided throughout my studies.

Abstrakt

Mechanická porucha v průběhu letu představuje zásadní rizko nejen pro posádku letadla, ale také pro lidi na zemi. Je proto žádoucí, aby při plánování trajektorií pro letadla v městkém prostření bylo toto riziko zohledněno. V této práci je navržen plánovač, který minimalizuje rizika daná mechanickými poruchami letadla. Minimalizováno je riziko fatální poruchy, která vede ke ztrátě kontroly nad letadlem, a riziko částečné poruchy, která má za důsledek závažné omezení manévrovacích možností letadla. Takovou částečnou poruchou může být například ztráta tahu v důsledku porchu motoru. Navrhovaná metoda buduje strom nouzových přistání, který může poté být opakovaně použit pro naplánování nejlepšího dostupného nouzového přistání a pro odhad rizika, které je s takovým nouzovým přistáním spojené. Výpočet tohoto stromu stojí na diskretizaci stavového prostoru, díky které je možné předpočítat všechny manévry, které jsou ve výpočtu použity, čímž se zásadním způsobem snižuje výpočetní náročnost celého algoritmu. Tento strom je poté použit ve výše zmíněném rizikovém plánovači, kde je důležitou součástí výpočtu rizika. Obě tyto metody, strom nouzového přistání a rizikový plánovač, byly otestovány na realistických datech. Navržený strom nouzovýh přistání dokáže vyhodnotit rizika různých nouzových přistávacích ploch a výpočet odhadu je dostatečně efektivní, aby mohl být použit v rizikovém plánovači. Díky tomu dokáže navržený rizikový plánovač vyhodnocovat rizika spojená s různými částečnými poruchami a výsledné trajektorie v porovnání s nejkratšími výrazně snižují riziko.

Klíčová slova: Plánování nouzových přistání; Rizikové plánování; Trajektoriální plánování

Abstract

An in-flight failure poses a risk to both the people on board the aircraft and the people on the ground. It is therefore desirable that the path planning takes into account the potential risk a failure poses. This work proposes a risk-aware trajectory planner, which is able to minimize the risk induced by a fatal failure, which leads to an uncontrollable crash, as well as the risk induced by partial failure, which leaves the aircraft controllable but severely inhibits its maneuvering capabilities. The method builds an emergency landing tree structure, which can then be repeatedly used to determine the best available emergency landing location and to evaluate the risk such an emergency landing poses. The emergency landing tree computation relies on a discretization of the configuration space, which allows for the use of precomputed maneuvers and overall significantly reduces the computational complexity. The emergency landing tree has been employed in the risk-aware path planner as an integral part of the risk function. Both the emergency landing planner and the risk-aware planner have been empirically evaluated in a realistic urban scenario. The proposed emergency landing tree is able to consider multiple emergency landing locations with different risks, and the risk estimate query has been shown to be efficient enough to be used in the risk-aware planner. Thanks to this, the risk-aware planner is able to evaluate the risk induced by several different partial failures, and the paths it produces have been shown to be significantly less risky than the shortest paths they have been compared with.

Keywords: Emergency landing planning; Risk-aware planning; Trajectory planning

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

This work presents a risk-aware trajectory planner for unmanned aerial vehicles in urban environments, which would provide the safest emergency landing trajectory available in case of a partial failure. This problem is motivated by the overall rise of air mobility and the potential that unmanned aerial vehicles offer for general transportation.

As with any means of transport, air traffic has inherent risks that cannot be eliminated. In contrast to other options, the risk associated with air travel is relatively low; the number of fatalities (with respect to passengers and distance traveled) is a hundred times lower than that of car transport, and the only comparable transit option is public transport (bus, rail) [2]. Modern aircraft are safer, among other reasons, because they are technically reliable and comparatively safe, thanks to the high level of safety regulations and standards for pilot training, manufacturing, and maintenance. However, even though flying is relatively safe, with increasing of flights, the total number of accidents would increase as well if the safety of flying did not increase as well.

In addition to the increase in numbers of conventional flights, the concept of Urban Air Mobility (UAM) has gained much public attention in recent years. Multiple studies and analyses were conducted, focusing on different aspects of the topic, such as [3], which discusses the legal and societal barriers and the potential usages, and [4], which focused on market assessment and public acceptance in Europe. Urban air mobility has a wide range of potential usages, from air taxis, public services (such as air ambulance), and goods delivery, to security, agriculture and entertainment, and much more. However, several challenges must be addressed before the wide population can see the rise of personalized aircraft and drone-based delivery. Technological advances in recent years have brought the physical shell to the point that the first commercial short-range aircraft might become publicly available and potentially even common in the near future, however, even the most advanced vehicles are still in the testing phase of development, and a lot of work still needs to be done before the future can become the present. In addition to the technical obstacles, some legal issues need to be resolved, issues with the acceptance by the general public, and, importantly, safety concerns, as shown by the public response discussed in [3] and [4].

According to questionnaire data [3], people seem open to the idea of UAM. However, most respondents expressed some concerns, among others, about safety, and most would be reluctant to use unmanned vehicles. Most companies building such aircraft are aware of this and regard the safety concerns as a critical issue to address. Outside of the theoretical studies and air-traffic-related problem analyses, several different companies are racing to provide the first air taxi or private flying car. Designs range from rotary aircraft to fixed-wing vehicles, including an airborne equivalent of a dirt bike, and are quite often wholly outside conventional aircraft designs, and already there is a commercially-available FAA[1]-approved unmanned aircraft.

And with increasing numbers, the risk increases as well. Furthermore, with the expected rise of urban mobility, aircraft will come much closer to heavily populated areas, where any accidents have

[1] Federal Aviation Administration, the agency responsible for air traffic regulation, pilot licensing, airport security and similar issues in the United States

potentially much worse consequences. Therefore, safety is a high priority and an important research topic in this area.

1.1 Aviation Accidents

Even though flying is relatively safe, there were a lot of accidents, and a lot of the accident data is very well documented and publicly available. In the United States, the National Transportation Safety Board (NTSB) and the Aircraft Owners & Pilots Association (AOPA) collect and publish data about general aviation (non-military and non-commercial flights) aircraft accidents [5] [6]. Similar databases are available in other countries as well, for example by the Australian Transport Safety Bureau (ATSB) [7], and in Canada as well, by the Transportation Safety Board of Canada.

According to these databases, most of the accidents are caused by the pilot (around 60 % in 2019, according to [6]), and of the rest, about half is classified as other/unknown, and the other as mechanical issues. However, even though mechanical issues are responsible for a relatively small portion of the overall accident rates, that still leaves hundreds of accidents every year caused by some mechanical issue. Out of the mechanical issue-related accidents, the vast majority were caused by powerplant failure. Powerplant failure leads to loss of thrust and may even lead to the loss of other control surfaces further, inhibiting the maneuverability of the vehicle. More detailed information is available in the NTSB databases [5], which include detailed descriptions of each accident. According to this database, other relatively common mechanical/technical accident causes are fuel-related and non-power-related component failure, which includes, for example, control surface failure and landing gear failure.

Following a partial failure, such as loss of thrust, reduced lift, or control surface failure, a skilled and lucky pilot may be able to guide the aircraft safely to the ground and execute an emergency landing on a nearby field or even return to the airport. However, such a failure often leads to a crash landing. Additionally, the flying aircraft gives little time to plan and evaluate options. Even an extra second of hesitation could lead to a stall and subsequent crash, which could have been avoided. If such a failure occurs during a flight of an unmanned aircraft, the autopilot must react in order to minimize the risk. In the case of an unmanned flight over an unpopulated area, even in case of complete system failure, it is very unlikely that anyone or anything but the aircraft itself is endangered. However, if unmanned aerial vehicles are to be used in urban areas, some safety mechanisms must be in place, which would minimize the risk to people on the ground and guide the aircraft to the safest possible emergency landing location. This work focuses on minimizing the risk through planning and on providing the safest possible emergency landing trajectories in case of a partial failure.

1.2 Structure of the Work

Chapter 2 describes the related work. It focuses on the works on constraint satisfaction and path planning, on emergency landing approaches, and concludes with a description of the set cover problem, which is closely related to the selection of landing locations post-failure. The studied problem is formally described in Chapter 3. The failures that the proposed approach can simulate are discussed in Chapter 4. The proposed method itself is described in Chapter 5, where the emergency landing planner and the risk-aware planner are described in detail. The properties of the proposed method are discussed in Chapter 6. The work concludes with Chapter 7, which summarises the proposed method and its results.

Chapter 2
Related Work

The herein studied problem stands for finding the least-risky path for a fixed-wing aircraft. The risk is determined as the risk associated with fatal failure resulting in a ballistic fall and by a partial failure, such as loss of thrust, which results in inhibited maneuverability of the aircraft. In the case of a partial failure, the aircraft is still controllable and an emergency landing is possible.

The problem poses several challenges, and this section provides an overview of both the contemporary and the older methods found in the literature that have been used to address similar tasks. Firstly, the challenge of satisfying the vehicle constraints is addressed in Section 2.1. Next, in Section 2.2, the path-planning algorithms are handled. Those algorithms form the basis for both the emergency trajectory landing computation and the Risk-Aware path planner. The Section 2.4 discusses the works on emergency landing site selection and trajectory planning. At the end, in Section 2.5, a short summary of works on set cover problems is presented, as it is closely related to the selection of potential landing sites.

2.1 Fixed-Wing Aircraft Trajectory Planning

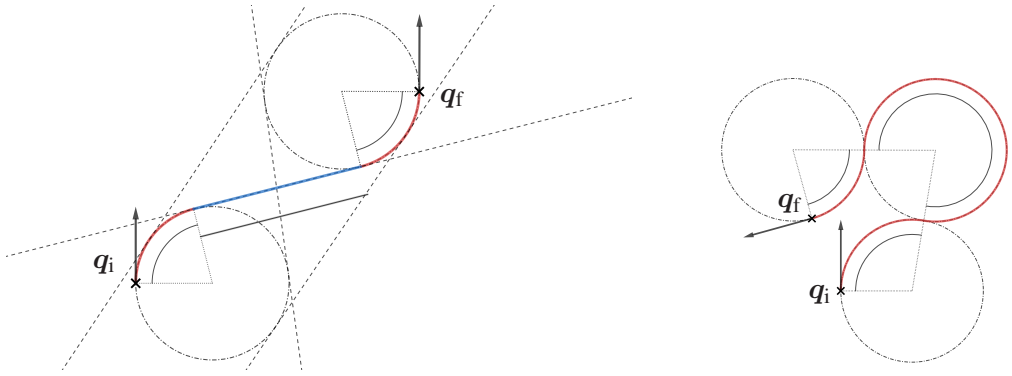
In order to plan a feasible trajectory for a fixed-wing aircraft, its motion constraints have to be satisfied. Since this work handles the task of risk-aware trajectory planning for small fixed-wing vehicles, for the rest of this section, it is assumed that only this type of aircraft is considered. This section describes the works handling the motion constraints and discussing the trajectory primitives used for planning for this kind of aircraft.

The fixed-wing aircraft motion has been thoroughly studied since the first airplanes were built, both from the point of view of construction and design of such an aircraft, as well as from the pilot's seat. The physics and aerodynamics of airplanes are addressed, for example, in [8]. The book covers kinematic and dynamic equations, aerodynamic forces and moments, and more, with a focus on autopilots and simulation. A more practically oriented point of view is presented by the U.S. Federal Aviation Administration in [9], which aims to provide a basic understanding of aerodynamics for pilots.

However, these works handle the aerodynamic forces and physics of the flight itself. This leads to a high level of accuracy, which, however, comes at the cost of complexity. For the purposes of trajectory planning, this level of precision is unnecessary, and because of its complexity, the computational burden is very high. Therefore, simpler geometrical models are used.

A commonly used path-planning technique is to utilize Dubins curves. A Dubins curve is the shortest path with constrained curvature connecting two configurations in the 2D plane. It can be used to describe the trajectory of a simplified car-like vehicle, that is, a vehicle with a constant forward speed and a limited turning radius. The problem of finding the shortest curvature constraint path was originally proposed by Andrey Markov in the nineteenth century but was first solved in 1957 in [10]. For any two configurations, the shortest connecting curve consists of three segments, each of which is either a circular arc or a straight line. In addition, any such path has one of two forms: two curved ones and one straight segment between them, or three curved segments if the two configurations are too close to each other. Theses two types are commonly referred to as a CSC and CCC type curves,

respectively. Examples of each of the Dubins maneuvers are depicted in Figure 2.1.



(a) CSC-type (b) CCC-type

Figure 2.1: Depictions of the two basic types of Dubins curves. If the distance between the two configurations is sufficiently large relative to the minimal turning radius, the CSC-type curve is the shortest possible path. If the two configurations are too close, the CCC-type curve is shorter.

In [11], another formalization of the shortest paths was presented. The initial and final configurations are transformed so that the initial configuration is at the origin, and the input is scaled so that the turning radius limit is equal to one. This means that the input is then fully described by the two heading angles and the distance. This highlights the symmetries of the problem regarding the types of the curves, CCC or CSC, with respect to the heading angles. Additionally, a mathematical formulation is presented, which allows for the same equations to be used to describe both the straight segments and the circular arc segments.

The most straightforward extension of the problem to 3D only extends the configuration space by introducing two additional dimensions, the altitude and the pitch angle. The configuration space is $\mathbb{R}^3 \times \mathbb{S}^2$, and no further restrictions are introduced. A solution for this problem is presented in [12]. However, this approach is not applicable to the type of aircraft studied in this work since the model does not restrict the pitch angle in any way, which can lead to way too steep trajectories.

Another extension of this problem was presented in [13]. In addition to the curvature constraint, this work also restricts the pitch angle. The presented model, called the Dubins Airplane model, is a four-dimensional system, with configurations $\boldsymbol{q} \in \mathbb{R}^3 \times \mathbb{S}$, the roll angle, and the pitch angle being relaxed. The proposed solution handles the task in three separate cases, divided by the altitude difference between the initial and final configurations. For the low altitude difference case, the 2D Dubins path is extended to 3D by scaling the path along the z-axis. If the altitude difference is sufficiently low, this extension does not violate the pitch angle constraint and thus can be used as is. The high altitude difference case is the situation where the shortest possible helical segment cannot gain (or lose) enough altitude. In this case, one or more helical turns are inserted, with the radius set to exactly compensate for the altitude difference. The last case is the situation where the altitude difference is not enough to warrant a helix curve but too high for the 2D Dubins path extension to be feasible. In this case, the curve is elongated, for example, by replacing a straight segment with a wave-like curve.

In later works on this topic, the model was further restricted by considering the pitch angle change limitation of the vehicle. In [14], the problem is addressed by inserting two short segments at the beginning and the end of the trajectory, which handle the changes of the pitch angle, while the rest of the path is computed as a 2D Dubins path, with added altitude component. The minimum and maximum pitch angle is not considered, just its change rate is limited.

Similar to the Dubins airplane approach [13], a common maneuver used to handle big altitude differences is a helix curve. Such approaches were proposed, for example, in [15] and [16]. Another method is presented in [17]. This method handles the vertical profile and the horizontal path as separate Dubins curves. If the horizontal trajectory with the vertical profile applied violates some constraints, the parameters of the curves are iteratively tuned until a feasible solution is found.

Alternative ways of finding curvature-constrained paths have also been proposed in the literature. For example, in [18] 7th order bezier curves have been used for generating curvature-constrained paths in 3D that also comply with the pitch angle limitations. This work was one of the earliest works that addressed the pitch angle change rate, as well as the pitch angle limitation, and the proposed bezier-curve-based method achieves a continuous pitch angle. However, in terms of the length of the generated paths, this method is considerably worse, even though the resulting paths are smoother.

Another variant in terms of path-planning for aerial vehicles is to use trochoids, as proposed in [19]. Trochoid curves are similar to the Dubins curves but allow the planner to take into account wind conditions. In no-wind scenarios, they are exactly the same, and in the case of wind, the trochoidal curves alter the turning radius. When the heading of the aircraft is the same as the direction of the wind, the turning radius is increased, and when the heading is in opposition to the wind, the turning radius is decreased.

2.2 Planning Algorithms

Another aspect of the studied risk-aware trajectory planning problem is the planning itself. This section presents the algorithms that are used to form path plans from the individual maneuvers presented so far.

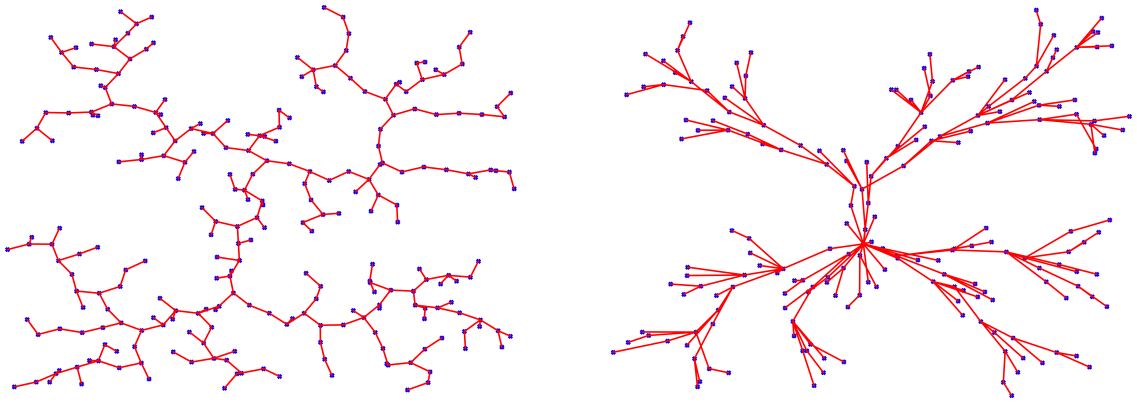
There are two main categories of path-planning algorithms. First of those are the graph-based algorithms. Those algorithms search a discretized state space, which can be represented by a graph. This graph can then be searched using graphs-search algorithms and heuristics to guide the search. Those algorithms always find the optimal path if a feasible path exists, but representing the whole state space as a graph might be computationally infeasible. The second big group is the randomized sampling based-algorithms. Those algorithms use random sampling of the state space, which allows them to build a graph-from a continuous state space. They are effective even in high dimensions, but they do not guarantee optimality and may be sensitive to the selected sampling method.

The simplest graph based algorithm is the BFS, which is the basis of more advanced algorithms. A well-known derivative of the BFS algorithm is Dijkstra's Algorithm [20]. In a graph with weighted edges, Dijkstra's algorithm finds the minimal path between any two nodes in the graph. The algorithm starts by expanding the start node, with cost zero. Whenever a node is expanded, all of its unexpanded nodes are checked and added to a priority queue, with priority equal to the cost of the current node plus the cost of the connection between them. After the expansion of a node is done, the node at the top of the priority queue is expanded, and this is repeated until the target is reached or until the queue is empty. If the algorithm finds a path to the target node, the path is guaranteed to be the shortest possible, and if it does not find any path at all, it means that no possible path exists.

However, Dijkstra's algorithm searches through the whole state space, which might be very computationally demanding. Therefore, a further improvement was proposed in the A* algorithm [21], which utilizes a heuristic function to guide the search toward the goal. It works similarly to Dijkstra's algorithm, but in addition to the cost of the node determined based on the cost of the graph edge, it adds the heuristic value of the node to the priority. The optimality of A* depends on the properties of the

heuristic function in that if the heuristic is admissible, the resulting path is optimal. An admissible heuristic is non-negative and never overestimates the cost of getting to the goal. D*-lite [22] is a variant of the A*, which works on a grid graph and can handle a dynamically changing environment by storing additional information about the nodes. When the environment changes, the affected nodes are recomputed, and the change propagates to their neighbors. In this manner, only the parts of the graph affected by the change are actually recomputed.

A second big category of planning algorithms is the randomized sampling-based algorithms. They can be further divided into two classes, roadmap-based (PRM) and tree-based (RRT). The probabilistic roadmap algorithm (PRM) [23] builds a graph from randomly sampled configurations from the search space and connects the sampled configurations to other nearby samples if the connections are feasible. It works in two phases: the first phase, the learning phase, builds the road map, and the second, the query phase, finds a path through the roadmap. During the learning phase, a predetermined number of samples is randomly generated, and each of those then represents a sampled configuration. For each of those configurations, a path to other near nodes is constructed and added to the graph as an edge. Each edge between two vertices then represents a feasible connection between two configurations, such as a path from one to the other. Once the graph is constructed, a graph planning algorithm, such as Dijkstra's algorithm or A* discussed above, can be used to plan a path, and this is the query phase. A useful property of the algorithm is that the same road map can be used to resolve any number of queries. This algorithm is probabilistically complete but is not asymptotically optimal. There are multiple variants of this algorithm, such as simplified PRM (sPRM) [24], which simplifies the connection of the nodes used in the original algorithm. Other variants are lazy PRM [25] (or a very similar fuzzy PRM [26]), which checks the feasibility of the path at the last possible moment. These variants assume all connections to be feasible, and only when the graph search algorithm finds a solution is it checked for collision. The colliding edges are then removed, and the search starts anew. This means that the computationally intensive collision check is performed only for connections that are actually considered to be used in the resulting path.



(a) Tree generated by the RRT algorithm.

(b) Tree generated by the RRT* algorithm.

Figure 2.2: Comparison of the RRT and the RRT* trees. The rewiring step of RRT* enables the algorithm to find much more straight paths. In contrast, the RRT uses the shortest path to the new node at the time of insertion, and the connections are never updated.

Similarly to the PRM-style algorithms, the Rapidly-exploring Random Tree algorithm (RRT) [27] generates random samples from the configuration space, which it connects to the already sampled graph. However, instead of generating all the samples at once and then searching for possible connections to all near nodes, each sample is connected immediately when it is generated. Furthermore, any new node is connected to the tree by a single edge, which creates a tree. Additionally, the RRT algorithm

uses a so-called steer function. The steer function expands the tree towards the randomly generated configuration by applying a control input from the selected parent node in the general direction of the randomly selected sample. Thanks to this step, the RRT algorithm does not need a point-to-point planning primitive to generate a feasible plan but can use control-based sampling. This property greatly increases the applicability of this algorithm. A major disadvantage of the RRT algorithm is, that it is only a single query. That means that the whole algorithm has to run again if the start and goal configuration changes, as opposed to PRM, which can reuse the learned map to process additional queries. However, if only the goal (or the start, if the search is reversed) configuration changes, the RRT tree generated so far can be reused.

Algorithm 1: RRT* [1]

Input: $\boldsymbol{g}$ – Goal Configuration.
Input: $\boldsymbol{s}$ – Starting Configuration.
Parameter: n – Number of nodes.
Parameter: k – Near node count.
Parameter: Δt – Time step.
Output: Constructed Graph G

1 $G \leftarrow \{\mathbf{E} \leftarrow \emptyset, \mathbf{V} \leftarrow \{\boldsymbol{s}\}\}$
2 **for** $i = 1...n$ **do**
3 $\quad \boldsymbol{q} \leftarrow \text{RandomSample}()$
4 $\quad \boldsymbol{q}_{\text{nearest}} \leftarrow \text{Nearest}\,(\mathbf{V}, \boldsymbol{q})$
5 $\quad \boldsymbol{q}_{\mathbf{0}} \leftarrow \text{Steer}\,(\boldsymbol{q}_{\text{nearest}}, \boldsymbol{q}, \Delta t)$
6 $\quad \mathbf{V} \leftarrow \mathbf{V} \cup \{\boldsymbol{q}_{\mathbf{0}}\}$
7 $\quad Q_{\text{near}} \leftarrow \text{Near}\,(\mathbf{V}, \boldsymbol{q}_{\mathbf{0}}, k)$
8 $\quad \boldsymbol{q}_{\text{best}} \leftarrow \boldsymbol{q}_{\text{nearest}}$
9 $\quad$ **foreach** $\boldsymbol{q}_{\text{near}} \in Q_{\text{near}}$ **do**
10 $\quad\quad$ **if** $\text{Cost}\,(\boldsymbol{q}_{\text{best}}) > \text{Cost}\,(\boldsymbol{q}_{\text{near}}) + \text{Cost}\,(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\mathbf{0}})$ **then**
11 $\quad\quad\quad \boldsymbol{q}_{\text{best}} \leftarrow \boldsymbol{q}_{\text{near}}$
12 $\quad \mathbf{E} \leftarrow \mathbf{E} \cup \{(\boldsymbol{q}_{\mathbf{0}}, \boldsymbol{q}_{\text{best}})\}$
13 $\quad$ **foreach** $\boldsymbol{q}_{\text{near}} \in Q_{\text{near}}$ **do**
14 $\quad\quad$ **if** $\text{Cost}\,(\boldsymbol{q}_{\mathbf{0}}) + \text{Cost}\,(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\mathbf{0}}) < \text{Cost}\,(\boldsymbol{q}_{\text{near}})$ **then**
15 $\quad\quad\quad \mathbf{E} \leftarrow \mathbf{E} \setminus \{(\boldsymbol{q}_{\text{near}}, \text{Parent}\,(\boldsymbol{q}_{\text{near}}))\}$
16 $\quad\quad\quad \mathbf{E} \leftarrow \mathbf{E} \cup \{(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\mathbf{0}})\}$
17 **return** G

RRT is probabilistically complete but not optimal. Once a node has been added to the tree, its parent is set and cannot change. Thus, once a path to a configuration has been found, there is no way to improve it. The parent node is selected based on the distance (or any other metric) to the new node, so the length of the path from the parent to the root does not play any role in the selection. The optimality is addressed in other variants of the algorithm, Stable Sparse RRT (SST) [28], which is asymptotically near-optimal, or RRT*, which is asymptotically optimal.

The RRT* algorithm is described by Algorithm 1. In each iteration, a random sample is generated, and a parent node is selected. The parent node is expanded using the steering function (same as the RRT), and the resulting node is added to the tree. Once the node is added to the tree, the rewiring step searches for possible improvements by considering the connections to other near nodes and not only the single nearest one. If a node offers a better path from the root to the newly added node than the current parent node, it replaces that parent. This step is shown in the Algorithm 1 on Lines 9 to 11.

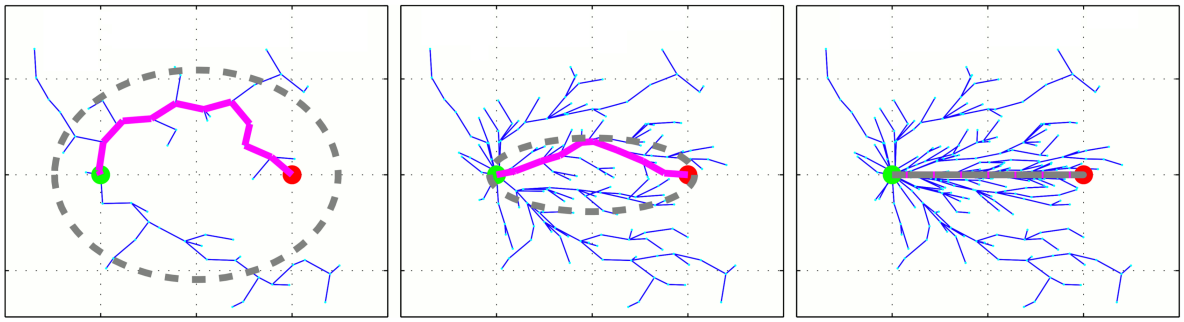


Figure 2.3: Example of the informed RRT* algorithm run. After an initial solution is found, all possible shorter solutions are restricted to an elliptical area around the start and goal configurations. Any path through a node outside of this ellipse is necessarily longer. Thus, no samples outside of this area are generated. As the solution improves, the maximal length of any improving path decreases, and the ellipse shrinks. Image is courtesy of [29]

Once the best parent is established, the new node is evaluated as the parent for the other near nodes. If the newly added configuration is better a parent for another node already in the tree, the previous parent is replaced, and the cost improvement is propagated to its children, as shown on Lines 13 to 16, Algorithm 1.

As opposed to the RRT and SST, RRT* requires not only forward propagation but also needs to be able to compute a connection between two configurations. However, unlike the base RRT algorithms, it is capable of finding the optimal solution. Being able to connect any two configurations allows RRT* to select a better parent node and to execute a rewiring step, in which the neighborhood of the newly added configuration is checked for nodes that can improve the tree. Figure 2.2 shows an example of the RRT tree and of the RRT* tree.

There are multiple other variants of the RRT/RRT* algorithm. Bi-directional RRT [30] executes the search from both the goal node and the start node, which greatly reduces the configuration space that has to be searched. Some other variants are based on informing the sampling of the points based on some heuristic. One such algorithm is the Informed RRT* [29], an example of which is shown in Figure 2.3. The general idea is that once a feasible solution has been found, a large part of the configuration space becomes irrelevant because any path leading through it must necessarily be longer than the already known solution. Therefore, the sampling can be restricted to the area, which can achieve some improvement. This idea is further developed in [31] in an algorithm called Batch Informed Trees (BIT*). The search is initialized by uniformly sampling the search space, and then the algorithm employs a heuristic search until a solution is found. When the samples can no longer improve the solution, the sampling is refined by adding a new batch of samples, and the search continues.

2.3 Risk Aware Path Planning

Traditionally, the goal of path-planning problems is to minimize the length of the resulting trajectory. However, quite often, there are other important metrics. With respect to small aircraft in urban scenarios, the risk induced by an in-flight failure is a more important metric because an uncontrollable aircraft poses a much more serious problem than a short delay a safer route might require.

In [32], a risk-minimizing A*-based algorithm is proposed. The risk is evaluated based on a static risk map, which is built based on the risk based on the probabilities of hitting a person and the probability of a failure happening, and the probable crash landing locations. The work proposes an off-line path

planner based on the A* algorithm, which plans a risk-minimizing trajectory based on this map, and proposes a post-optimization procedure, which searches for improvements to the path based on line of sight.

Additionally, a dynamic risk evaluation is also presented. If the in-flight estimated risk differs from the risk map the path was based on, an on-line planner is used to compensate for this change. The off-line planner cannot be used because of its high computational demands. Therefore, a simpler on-line planner is also proposed, called the borderland algorithm. The on-line planner attempts to reduce the risk of the path by circumnavigating the higher risk areas.

An approach utilizing the RRT* algorithm is presented in [33]. The algorithm uses a risk-based cost function, which evaluates the risk to people on the ground based on a number of different factors. The area endangered by the crashing aircraft, called causality area, is determined from the size and approach angle of the aircraft and from the average size of a person. The fatality rate is then estimated from the impact energy and sheltering factor of the surroundings. Those values are then combined to determine the expected number of casualties, which is then used as the cost function of the RRT* algorithm. This algorithm is further iterated on in [34], which replaces the map-based risk estimate by assessing the risk of each node directly during the planning process.

A similar algorithm is proposed in [35]. In addition to the minimization of the risk induced by an in-flight loss of control, the proposed method is able to completely eliminate the risk induced by loss of thrust by maintaining altitude such that the aircraft would be able to glide to a safe emergency landing site. The emergency landing trajectory guarantee is achieved by only adding nodes that are above the safe altitude level. The risk-aware planner itself is based on the RRT* algorithm as well, using the risk of the path from the root of the tree to the given node as the cost of the node.

2.4 Emergency Landing Planning

The planner proposed in this work minimizes the risk associated with an in-flight failure. However, if an emergency landing is possible, the risk depends on the emergency landing trajectory, and in order to evaluate the risk, it is necessary to study the emergency landing planning techniques. This then allows the planner to include the risk associated with the emergency landing as a part of the risk function used for the planning.

The topic of emergency landing trajectory planners has been thoroughly discussed in the literature, especially with the advent of smaller unmanned aerial vehicles and their increasing potential for civilian employment during the last few years. The most commonly discussed emergency is an engine failure leading to loss of thrust. This is the most common technical failure, and if handled well, a safe emergency landing is possible. The emergency landing is a complex topic with many different aspects, including but not limited to path-planning from the location of the failure to the landing site, post-failure vehicle constraints satisfaction, landing site selection, risk minimization, landing maneuver safety, or environmental hazards. The works concerning this topic cover the full spectrum of the aspects, and this section provides an overview of those works.

In [36], the problem of emergency landing following a total loss of thrust is addressed in two parts. The first step is to select a reachable runway on which to perform the emergency landing. The second step is the path-planning itself. The selection of the landing site is limited to a database of airport runways, and the reachability of the runway is estimated based on the current altitude and wind conditions, but the terrain is not taken into account in any way. If no reachable runways are available, some of the constraints are relaxed, and the search is attempted again. If multiple are available, the authors propose

a ranking based on the runway utility. The utility is computed as a weighted sum of several factors based on the properties of the runway and other relevant conditions: the runway width and length, the distance from the aircraft, headwind velocity, the surface of the runway, facility score, and more. Afterward, a Dubins path to the selected runway is computed. If the altitude of the aircraft is too high, a sequence of waypoints is constructed, which guides the aircraft using easy-to-follow maneuvers.

In [37], a simulation of forced landing to evaluate emergency path planners was proposed, together with a comparatively straightforward planner. The reachability of a potential landing site was determined based on the altitude and position of the aircraft in relation to the landing site. The reachable area was modeled as a cone shape below the aircraft, which was then deformed based on the wind.

In [38], the problem of finding the least risky emergency landing trajectory for a damaged aircraft is presented. The authors identified several characteristics of the problem: the initial state, the control envelope of the damaged aircraft, the potential landing sites with their characteristics, and the obstacles. The problem is addressed from the perspective of longer-range path-planning, in that the majority of the planned trajectory is expected to be sufficiently high, that terrain and urban areas are not a concern, but severe weather conditions might play a role. Additionally, the dynamics of the vehicle are not taken into account, and things like a limited turning radius would be negligible in this scope. The resulting path is then a sequence of points that the aircraft is expected to be able to navigate, and any potential restrictions on maneuverability are addressed by manipulating the obstacles. The risk is quantified as expected loss of life and is assessed as the sum of risks during different stages of the forced landing trajectory. The En Route risk assumes risk induced by the pilot losing control over the plane and considers only the crew and passengers as being in danger. The second part of the risk assessment is associated with approaching the landing site over a populated area and takes into account the level of damage to the aircraft, visibility, altitude, etc. The last part of the risk is determined based on the properties of the airport, such as the characteristics of the runway and the available facilities. The planner itself is based on the A* algorithm executed on a modified visibility graph-based roadmap.

In [39], the authors propose to use trochoidal curves instead of the commonly used Dubins curves, which allow the planner to take potential environmental effects into account. Thus, the planner can compensate for the wind conditions if the wind has constant direction and force. However, trochoidal curves do not have any straightforward extension into 3D. Therefore, the descent rate has to be addressed separately by estimating the glide ratio and time necessary to cover each segment of the path. The planner considers only a single predefined landing site, but computationally it is feasible to run several instances at the same time. The proposed planner has been tested and performed well in the Robot Operating System environment, as well as on two different real-world aircraft.

The any-time emergency landing planner proposed in [40] offers in-flight computation of a safe emergency landing trajectory in case of complete loss of thrust. The planner is able to take into account terrain and obstacles, which the more straightforward planners are unable to do. Following the loss of thrust, the aircraft can only glide, and as a consequence, it is constantly losing altitude. Therefore, the authors propose to optimize the safe altitude of the configuration using an RRT*-based tree. Optimizing the altitude reduces the dimensionality and, thus, the overall computational complexity of the algorithm. The safe altitude of a node is determined based on the altitude loss of the maneuver from the parent to the node, increased by any obstacles that must be flown over. This means that the safe altitude of the node depends not only on the maneuver itself but also on the safe altitude of the parent. Additionally, it is assumed that the aircraft can lose altitude at will. Thus, any path leading to a safe landing site is considered safe, regardless of the altitude of arrival, as long as the altitude is higher than the landing site itself. The algorithm runs during the whole flight and builds an RRT*-based tree from the potential landing sites to the aircraft, which makes it possible to plan the trajectory for a moving

airplane. New nodes are continuously added to the tree using informed sampling, and unreachable ones are pruned in order to decrease the memory requirements of the algorithm so that it is viable for it to run during the whole flight. The authors also provide an aerodynamic point of view on the loss of thrust, from which the damaged aircraft model is derived.

The emergency landing planner proposed in [40] is further developed in [41]. The emergency landing trajectories are computed in advance. This produces a dense roadmap, which can be queried to obtain the safe altitude for any given configuration. The planner is used in an offline surveillance planner with a safe emergency landing guarantee in case of total loss of thrust. The surveillance trajectory has to visit a set of given locations and return to the original configuration, and it uses a similar safe emergency landing planner as proposed in [40]. A safe altitude is the altitude level from which there is a possible emergency landing trajectory which leads to a safe landing site, such as a runway. This is then used to compute trajectories connecting the individual locations that are safe with respect to total loss of thrust, in that in the event of loss of thrust, the aircraft can always follow a path to a safe landing site.

The work [42] handles the task from the point of view of the safety of the landing itself. The main focus is that the proposed waypoint-based method creates a trajectory that leads the aircraft to the runway with a safe approach speed and pitch angle. The authors propose an analytical velocity prediction method that estimates the speed of the aircraft in a given path segment. This velocity predictor, together with the restrictions on the approach speed, altitude, and direction, is then used to create a non-linear system of equations. The solution of this system is then a forced landing trajectory that maximizes the safety of the landing maneuver itself.

2.5 Set Cover Problem

A sub-problem in this work is the selection of considered emergency landing locations, such that the coverage of the configuration space is maximized, the risk is minimized, and the number of selected landing locations is low. If an airport is available, it is obviously the best choice because there are dedicated runways and potentially even the possibility of repairing or refueling the aircraft. However, in many cases, the airport might be unreachable, and thus another landing location has to be selected.

The problem of selecting the best emergency landing locations to consider can be seen as a variant of the maximum coverage problem, as each considered emergency landing covers a part of the configuration space, and we want to cover as much of the space as possible. This section describes the properties of the problem, its approximation algorithms, and its relation to other covering problems. The maximum coverage problem is a variant of the set cover problem, and therefore the set cover problem needs to be addressed first. The analogy of the studied landing site selection to the maximum coverage problem is discussed once the maximum coverage problem is presented.

The set cover problem [43] is the problem of finding a set of subsets that covers the whole universum. Formally, given a universum $\mathcal{U}$, and a collection $\mathcal{S} = \{\mathcal{S}_1, ..., \mathcal{S}_m\}$ of m subsets of $\mathcal{U}$, the goal is to find a minimal set cover $\mathcal{A} \subset \mathcal{S}$, for which it holds, that $\bigcup \mathcal{A}_i = \mathcal{U}$. This problem has several variants, such as the weighted set cover problem, where each of the subsets has a weight associated, and the goal is to find the set cover with the minimum weight. The set cover problem has also been shown to be NP-complete [43].

In [44], it has been shown that the greedy heuristic for the set cover problem is an approximating algorithm. In every iteration, the greedy algorithm selects the set that has the most elements that have not been covered yet. In the case of the weighted variant of the problem, the weight is distributed

across the newly added elements. The upper bound on the performance ratio has been shown to be $H(m) = \sum_{j=1}^{m} \frac{1}{j}$, where m is the cardinality of the universum $\mathcal{U}$, i.e. the result of the greedy algorithm is at most $H(m)$ times the value of the optimal solution. The proof has been later refined in [45], by showing the exact performance ratio to be $\ln m - \ln \ln m + \Theta(1)$.

The maximum coverage problem [46] is closely related to the set cover problem. In this problem, instead of trying to find the minimum set cover, the goal is to select at most $k \leq m$ of the subsets from $\mathcal{S}$, such that the cardinality of their union is maximized.

The problem of selecting the optimal potential landing sites is very similar. The terrain corresponds to the universum, and each potential landing site corresponds to one of the subsets. Each potential landing location is reachable from any configuration in an area above it. In higher altitudes, the area is bigger, as the aircraft can glide further. Thus, the area can be estimated as a cone based on the gliding properties of the aircraft. Thus, the area covered by the landing location is estimated as a radius around it, as it corresponds to a projection of the cone to the terrain from a certain height. The cost of each of those landing locations is proportional to the risk associated with an emergency landing at that location. Thus, the problem is to select at most k landing locations (subsets), such that they cover as much of the terrain (universum) as possible, and their risks (costs) are minimized.

The maximum coverage problem is motivated by the real-world implications of optimizing the locations of facilities providing some kind of service to customers, such as shops or ambulance depots. The customers are then the elements of $\mathcal{U}$, and a facility that covers all customers within a certain radius corresponds to a set $\mathcal{S}_i$. Selecting k sets from $\mathcal{S}$ then corresponds to selecting k locations to build the facility on. From the point of view of the set cover problem, the max cover problem is almost the same problem, but instead of minimizing the number of subsets used, it aims to maximize the covering with a given number of subsets. Similarly to the set cover problem, the maximum coverage problem has been shown to be NP-complete. In [47], a greedy heuristic was compared to a linear programming approach and to an improved greedy heuristic. Similarly to the set cover problem, the greedy algorithm has been shown to be an approximating algorithm [46] that yields satisfactory results when applied in practice [48].

Chapter 3
Problem Statement

This work discusses Risk-aware Trajectory Planning, which, for any two configurations, would find a trajectory minimizing the risk associated with in-flight failures. Two types of failures are considered: fatal failure leading to uncontrollable fall and a partial failure leading to a partial loss of thrust, potentially combined with inhibited maneuverability. The latter failure can still lead to a potentially risk-free emergency landing because the aircraft might still be capable of a gliding flight to a safe emergency landing location. Minimizing the risk associated with this kind of failure is the main contribution of this work.

This task can be decomposed into several subtasks: path planning to satisfy the vehicle constraints, emergency landing trajectory planning, and risk-aware trajectory planning. These individual subtasks are discussed in further detail in this chapter, in this order.

3.1 Satisfying the Vehicle Constraints

The nature of the fixed-wing aircraft imposes some constraints on the model, which have to be adhered to. The aircraft in this work is modeled as a Dubins airplane [13], which specifies and formalizes the motion constraints. Let $\boldsymbol{q} = [x, y, z, \psi, \theta]$ be a configuration of the model. It consists of the position $[x, y, z] \in \mathbb{R}^3$, the heading angle ψ, and the pitch angle θ. The configuration space is then $\mathcal{C} = \mathbb{R}^3 \times \mathbb{S}^2$. The trajectory of such an airplane must satisfy a minimum turning radius constraint, and the pitch angle must be within limits imposed by the properties of the aircraft. The state of the airplane modeled like this is then described by the following equations:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix} = v \begin{bmatrix} \cos(\psi)\cos(\theta) \\ \sin(\psi)\cos(\theta) \\ \sin(\theta) \\ u\rho^{-1} \end{bmatrix}, \tag{3.1}$$

where ρ represents the minimal turning radius, u is a steering control input, and v is a constant forward speed. The steering is limited by the maneuvering capabilities of the airplane and corresponds to the turning radius constraints of the airplane:

$$u \in [-1, 1]\,. \tag{3.2}$$

Similarly, as mentioned above, the pitch angle θ must be within a specified range,

$$\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]\,. \tag{3.3}$$

The pitch angle is assumed to change abruptly within the given limits. Compared to the heading angle, the maneuvers required to change the pitch angle are relatively fast. Therefore the flying airplane can change its pitch angle much more abruptly, and this simplification does not compromise the model.

One more constraint is imposed on the model, in that the airplane cannot pass through terrain and obstacles. This constraint is expressed as $\boldsymbol{q} \in \mathcal{C}_{\text{free}}$, where $\mathcal{C}_{\text{free}}$ is the collision-free part of the configuration space $\mathcal{C}$.

3.2 Emergency Landing

A failure that leaves the aircraft capable of flight, albeit with limited maneuverability, is considered a partial failure. An emergency landing trajectory is then a trajectory that the aircraft can execute following a partial failure. Following a failure $\mathcal{F}$, the emergency landing trajectory $\Gamma_{\mathcal{F}} : [0,1] \to \mathcal{C}_{\text{free}}$ is any controlled flight trajectory from an initial configuration $\boldsymbol{q}_{\text{i}} \in \mathcal{C}_{\text{free}}$ to a final configuration $\boldsymbol{q}_{\text{f}} \in \mathcal{T}$, where $\mathcal{T} \in \mathcal{C}$ is the terrain (including obstacles).

In case of a partial failure, the airplane is still capable of flight, but its maneuverability is limited. For example, in case of total loss of thrust, the airplane can glide as long as its speed is maintained, which is achieved by carefully controlling the pitch angle of the airplane. Thus a failure $\mathcal{F}$ is represented as an additional restriction on the control variables of the model, the control input u in (3.2) and the pitch angle limits in (3.3). Each failure is considered such that the restriction on pitch angle is at least $\theta_{\text{max}} < 0$ because such a failure prevents the aircraft from climbing high up in the air and performing an emergency landing somewhere outside the populated area. Multiple failures may be considered simultaneously, and the pool of all considered failures is denoted as $\mathbb{F}$.

During the flight, the aircraft must be above ground level and has to avoid potential obstacles. The terrain and obstacles, denoted as $\mathcal{T}$, are considered as a subset of the configuration space, defined by a function of the x and y coordinates, $\mathcal{T} = \{(x,y,z) \,|Alt\,(x,y) = z\}$, where $Alt : \mathbb{R}^2 \to \mathbb{R}$ is the terrain height on coordinates x, y. This terrain definition assumes that no collision-free configuration is below the terrain level, i.e., that the aircraft cannot fly under obstacles, such as under a bridge. The collision part of the configuration space $\mathcal{C}_{\text{O}}$ is then the subset of $\mathcal{C}$, which is at or below the terrain level or within an obstacle, $\mathcal{C}_{\text{O}} = \{\boldsymbol{q} = (x,y,z,\psi,\theta) \,|\boldsymbol{q} \in \mathcal{C}, Alt\,(x,y) \leq z\}$, and the collision-free part is the complement, $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{O}}$.

Then, $\mathcal{R}_{\mathcal{F}} : \mathcal{T} \times \mathbb{S} \to \mathbb{R}$ is the emergency landing risk function, defined such that the value $\mathcal{R}_{\mathcal{F}}\,(\boldsymbol{p})$ is the risk associated with an emergency landing on configuration $\boldsymbol{p}$ following a partial failure $\mathcal{F}$. Similarly to the Dubins airplane model, the pitch angle θ of the incoming aircraft is neglected, and only the heading angle ψ is considered. This value is defined for any configuration on the terrain level and for any considered failure, but the problem does not depend on its values.

The problem is to find a collision-free emergency landing path $\Gamma_{\mathcal{F}} : [0,1] \to \mathcal{C}_{\text{free}}$ from a given admissible configuration $\boldsymbol{q}_{\text{i}}$, which satisfies the gliding constraints of the aircraft model restricted by failure $\mathcal{F}$, and leads to an emergency landing location $\boldsymbol{q}_{\text{f}}$, that minimizes the associated risk:

Problem 1. Multi-failure Risk-aware Trajectory Planning for Urban Air MobilityEmergency Landing Path Planning Problem

$$\begin{aligned} &\min_{\Gamma_{\mathcal{F}}, \boldsymbol{q}_{\text{f}} \in \mathcal{T}} \mathcal{R}_{\mathcal{F}}\,(\boldsymbol{q}_{\text{f}}) \\ &\text{Subject to:} \\ &\qquad \text{Motion constraints (3.1) are satisfied} \\ &\qquad \Gamma_{\mathcal{F}}\,(0) = \boldsymbol{q}_{\text{i}}, \\ &\qquad \Gamma_{\mathcal{F}}\,(1) = \boldsymbol{q}_{\text{f}}. \end{aligned} \tag{3.4}$$

The ground landing risk function $\mathcal{R}_{\mathcal{F}}\,(.)$ is n naturally extended to the emergency landing risk $\mathcal{R}_{\mathcal{F}} : \mathcal{C}_{\text{free}} \to \mathbb{R}$, which covers the whole $\mathcal{C}_{\text{free}}$. Its values are defined as the ground landing risk $\mathcal{R}_{\mathcal{F}}$ value of the landing configuration of the least risky emergency landing trajectory.

3.3 Risk-Aware Path Planning with Emergency Landing Trajectory Guarantee

The overall goal is to find a least-risky path from an arbitrary, feasible initial configuration $\boldsymbol{q}_\mathrm{i}$, to an arbitrary feasible final configuration $\boldsymbol{q}_\mathrm{f}$. The risk of a configuration $\boldsymbol{q} \in \mathcal{C}_\mathrm{free}$ considered here has two components. The first is the risk induced by potential partial failures, $\mathbb{F}$, defined as a sum of the risks associated with the individual failures, $\sum_{\mathcal{F}\in\mathbb{F}} \left(p_\mathcal{F}\mathcal{R}_\mathcal{F}\left(\boldsymbol{q}\right)\right)$. The second component is the risk associated with fatal failure, $\mathcal{R}_\mathrm{q}\left(\boldsymbol{q}\right)$. This function may be any arbitrary risk function associated with the configuration. The complete risk function is a combination of those two values,

$$\mathcal{R}\left(\boldsymbol{q}\right) = \sum_{\mathcal{F}\in\mathbb{F}} \left(p_\mathcal{F}\mathcal{R}_\mathcal{F}\left(\boldsymbol{q}\right)\right) + p_\mathrm{fatal}\mathcal{R}_\mathrm{q}\left(\boldsymbol{q}\right), \tag{3.5}$$

where p_fatal denotes the probability of a fatal failure resulting in an uncontrollable ballistic fall.

For a path $\Gamma : [0,1] \to \mathcal{C}_\mathrm{free}$, the risk is computed as the integral of risks along its path. The optimal path Γ^* is then a path minimizing this risk:

Problem 2. Multi-failure Risk-aware Trajectory Planning for Urban Air MobilityRisk-minimizing path planning

$$\begin{aligned} \Gamma^* &= \arg\min_{\Gamma} \int_0^1 \mathcal{R}\left(\Gamma\left(t\right)\right) \mathrm{d}t \\ &\text{Subject to:} \\ &\Gamma\left(0\right) = q_\mathrm{i}, \\ &\Gamma\left(1\right) = q_\mathrm{f}. \end{aligned} \tag{3.6}$$

For a configuration $\boldsymbol{q} \in \Gamma$, in order to compute the risk $\mathcal{R}_\mathcal{F}\left(\boldsymbol{q}\right)$, there is emergency landing trajectory $\Gamma_\mathcal{F}$, for which $\Gamma_\mathcal{F}\left(1\right) = \boldsymbol{q}_\mathrm{f}$, for some $\boldsymbol{q}_\mathrm{f}$, for which $\mathcal{R}_\mathcal{F}\left(\boldsymbol{q}_\mathrm{f}\right) = \mathcal{R}_\mathcal{F}\left(\boldsymbol{q}\right)$ Consequently, if such a path is found, that means, that at any configuration the emergency landing trajectory in case of any of the considered failures is known.

Chapter 4
Failure Model

The risk-aware planner proposed in this work considers two kinds of failures: fatal failure, which results in an uncontrollable crash, and a partial failure, which severely inhibits the maneuverability of the aircraft, but the aircraft is still capable of controllable flight. In order to minimize the risk induced by partial failures, it is necessary to determine an emergency landing trajectory. Therefore, it is necessary to study the different effects that partial failure may have on the aircraft. Intuitively, loss of thrust leads to the aircraft being able to accelerate during level flight and consequently losing altitude, and broken control surfaces inhibit the ability to turn. This chapter introduces the considered failure models and attempts to shed light on how the specific mechanical failures reflect on the mathematical model.

4.1 Airplane Model

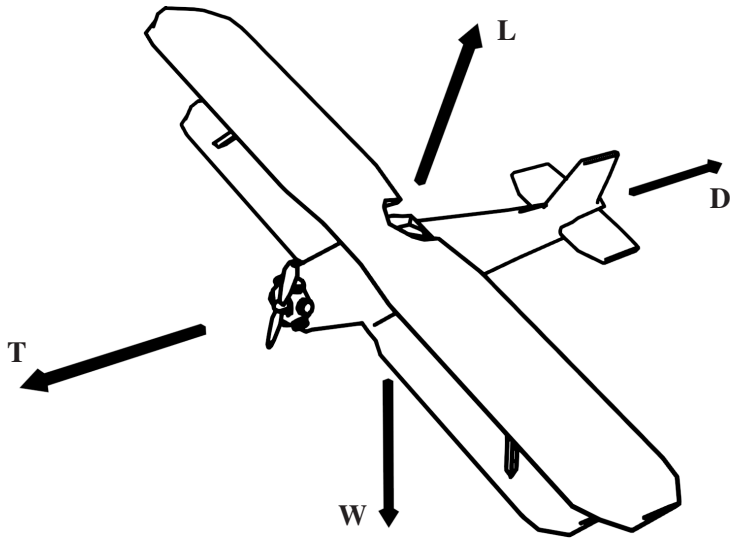


Figure 4.1: Forces acting on the aircraft. Thrust **T** pulls the aircraft forward, Lift **L** acts perpendicular to the wings, Weight **W** pulls the aircraft downwards, and Drag **D** acts in opposition to the velocity.

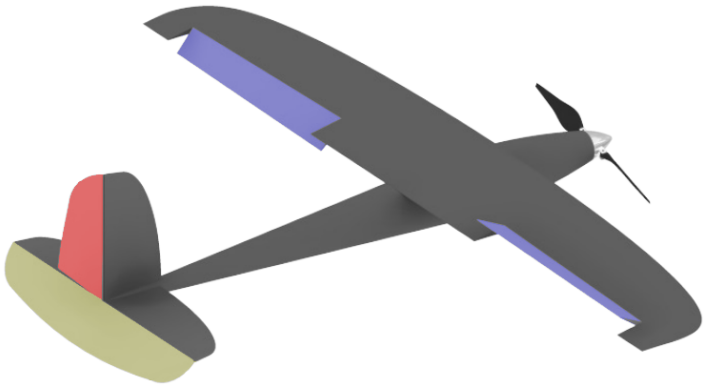
The model of an aircraft is based on [8].There are four main forces acting on the aircraft: Thrust **T**, Drag **D**, Lift **L**, and Weight **W**, as shown in the figure Figure 4.1. Thrust acts in the direction the aircraft is heading, while drag counteracts it, weight pulls the aircraft down, and the lift pulls the aircraft in the direction perpendicular to its wings. When the aircraft is in steady flight, which means, that the aircraft keeps a stable altitude, speed, and heading angle, these four forces are in balance.

The magnitude of the lift and drag is dependent on the properties of the airplane and is given by

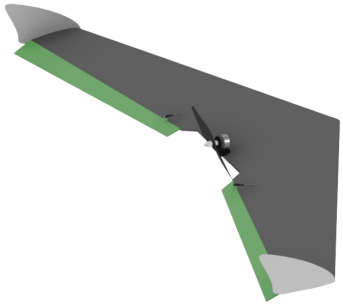
$$L = \|\mathbf{L}\| = \frac{1}{2}\rho C_{\mathrm{L}} S V^2, D = \|\mathbf{D}\| = \frac{1}{2}\rho C_{\mathrm{D}} S V^2 \tag{4.1}$$

for lift and drag, respectively, where ρ is the air density, S is the reference wing area, V is the velocity, C_{L} and C_{D} are the lift and drag coefficients, respectively. Both of the coefficients depend on

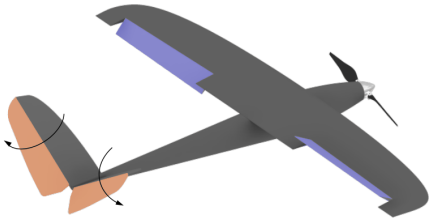
the geometrical properties of the airplane. Importantly, the drag coefficient also depends on the lift coefficient, due to lift-induced drag.



(a) The conventional control surfaces of the aircraft: rudder (red), ailerons (blue), and elevators (yellow)



(b) Control surfaces on a flying wing: The ailerons and elevators are replaced by the elevons (green). Rudder is missing altogether.



(c) Control surfaces on a V-tail: the rudder and the elevators are replaced by a V-shaped tail control surfaces, called ruddervators (orange).

Figure 4.2: The control surfaces of the aircraft, based on [8]. Images are courtesy of [8]

Conventionally, the aircraft has three types of control surfaces: vertical rudder, ailerons on the wings, and elevators on the tail, as depicted in Figure 4.2a. Two other configurations of the aircraft control surfaces are considered in [8]: the flying wing Figure 4.2b and the V-tail Figure 4.2c. In the case of the V-tail, the tail control surfaces, called ruddervators, can move in opposition, which has the same effect as the rudder, or in unison, which has the same effect as elevators. Similarly, the control surfaces of the flying wing, called elevons, can move together and function like elevators, or they can move differentially and function like ailerons. Therefore, any flight pattern achievable with the standard control surface configuration can be achieved by either of the other control surface configurations as well, and vice versa.

Generally, the rudder causes yaw, the ailerons cause roll, and the elevators cause pitch. However, each control surface has additional side effects on the other angles. The rudder and ailerons influence both the roll and yaw moments. That is, both have an effect on the turning capabilities of the airplane, and in order to perform a coordinated turn, both must be employed.

4.2 Considered Partial Failures

The Emergency landing planner creates a trajectory that a damaged aircraft can follow to a selected landing location. Therefore, it is necessary to define how potential partial failures affect the performance of the aircraft. Two different cases are considered, loss of thrust and loss of thrust combined with limited maneuverability. According to [6], the most common mechanical issue is powerplant failure, which leads to loss of thrust. Additionally, failures inhibiting the ability of the aircraft to gain height are deemed more critical to address because they pose a severe restriction on the flight range of the aircraft and thus restrict the potential emergency landing locations to the close vicinity. In terms of the Dubins airplane model, lift reduction will have the same effect as a partial loss of thrust. If the aircraft can gain altitude, a relatively safe solution (with respect to people) that is always available is to fly outside the urban area and crash-land somewhere where there is no risk to people on the ground. Additionally, this selection of considered failures allows for some simplifications and optimizations in the planner.

The loss of thrust model considered in this work has been adopted from [40], which models the equations and the failure models based on [8]. Following loss of thrust, the aircraft has to keep losing altitude in order to maintain the same velocity, which means, that it has to maintain a negative pitch angle. The necessary descent rate is dependent on the parameters of the flight, in that turning requires steeper descent to maintain the same speed.

In order to simplify the complicated aerodynamic model presented in [8], some assumptions about the motion of the aircraft are made. Firstly, the angle of attack is assumed to be zero. Secondly, all of the maneuvers are considered to be smooth and coordinated, with zero sideslip angle. Thus, when the sideslip angle is zero, the angle of attack is zero, and the bank angle is equal to the roll angle, the aircraft travels in the direction it is heading. A steady descending flight is assumed to be optimal for maximizing the glide distance.

The model is then defined as follows [40]:

$$\begin{aligned}
\dot{\boldsymbol{x}} &= V\cos(\psi)\cos(\theta)\,,\\
\dot{\boldsymbol{y}} &= V\sin(\psi)\cos(\theta)\,,\\
\dot{\boldsymbol{z}} &= V\sin(\theta)\,,\\
\dot{\boldsymbol{V}} &= \frac{1}{m}\left(T - D - W\sin(\theta)\right),\\
\dot{\psi} &= \frac{L\sin(\phi)}{mV},
\end{aligned} \tag{4.2}$$

where $W = \|\mathbf{W}\| = mg$ is the magnitude of the weight and ϕ is the roll angle.

The speed is assumed to be constant, and thus $\dot{V} = 0$. In case of a total loss of thrust, the thrust is zero, $T = 0$. Therefore, according to (4.2), it is derived that

$$D = -W\sin(\theta)\,. \tag{4.3}$$

Computing the coefficients from (4.1) allows for the pitch angle to be determined such that within the given segment, the aircraft is able to glide with constant speed. The pitch angle is computed by $\theta = \sin^{-1}\left(\frac{-D}{W}\right)$. Note that with a slight adjustment, it is also possible to simulate partial loss of thrust by adding the thrust component to (4.3): $D = T - W\sin(\theta)$. Note that the lift force is assumed to counteract both the drag and the weight force. Consequently, because during a turn a component of the lift force is directed to the side, the component of the lift force counteracting the drag is lower. Thus, while the aircraft is turning, the pitch angle necessary for a gliding flight is lower.

The second type of failure considered in this work is the loss of thrust combined with limited maneuverability. All of the characteristics of the (partial) loss of thrust remain, but a turning radius limitation is considered as well. The considered turning radius limitation may be symmetrical or asymmetrical. In terms of the Dubins airplane model described in Chapter 3, in (3.1), this corresponds to restriction on the control input u, in (3.2), in that the limits are closer to zero, so the resulting limits are $u \in [u_{\text{min}}, u_{\text{max}}]$, where $-1 \leq u_{\text{min}} \leq u_{\text{max}} \leq 1$.

Unlike the loss of thrust, the limited maneuverability failure is considered purely from the point of view of the planner, based on the simplified Dubins airplane model, rather than based on the dynamics model. In literature, such failures have been discussed, for example, in [38], but they were evaluated only numerically. That is, a full simulation of the aircraft was performed rather than an analysis of the damaged aircraft model. In [49], several potential mechanical failures have been discussed, and the effect on the simplified model has been numerically evaluated with respect to the specific airplane, Airbus A320 in this case. The method was used to determine the effects of individual failures on turning radius (left/right, if the failure caused asymmetry), turning rate (similarly, with potential asymmetry), descent rate, pitch angle, pitch rate, and glide ratio.

In case of a control surface failure, the aircraft might be forced to fly with an induced bank angle. It has been shown that in the case of an induced bank angle, the turning radius becomes asymmetrically restricted. The restriction may be so severe that the aircraft is unable to perform a turn in one direction at all or even that the aircraft has to turn constantly and can only influence the turning rate. A similar issue may also be a consequence of an asymmetrical thrust, but this is only possible when considering aircraft with several engines on the wings.

Both loss of thrust and loss of thrust combined with inhibited maneuverability can be cleanly translated into the Dubins airplane model. The post-failure Dubins airplane model is then formulated by the equations

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix} = v \begin{bmatrix} \cos(\psi)\cos(\theta) \\ \sin(\psi)\cos(\theta) \\ \sin(\theta) \\ u\rho^{-1} \end{bmatrix}, \tag{4.4}$$

where $u \in [-1, 1]$ and $\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]$. The (partial) loss of thrust translates to additional restriction on the pitch angle θ, $\theta \in [\theta_{\text{min}}, \theta^f_{\text{max}}]$, for which it holds that $\theta_{\text{min}} \leq \theta^f_{\text{max}} \leq \theta_{\text{max}}$. Similarly, the inhibited turning radius leads to additional restriction on the control input u, $u \in [u^f_{\text{min}}, u^f_{\text{max}}]$ where $-1 \leq u^f_{\text{min}} \leq u^f_{\text{max}} \leq 1$.

The post-failure limit values are determined based on the physical properties following the given failure. Loss of thrust has been discussed above, and the turning radius limitations can be determined based on simulations, as has been shown in [49].

Chapter 5
Proposed Method

This work handles the problem of risk-aware trajectory planning for unmanned aerial vehicles. The goal is to find a path for an aircraft for any pair of initial and final configurations, such that the risk induced by potential failures would be minimized. This task is motivated by the increasing potential of small unmanned aerial vehicles and by the hazards that are inherently tied to increasing numbers of flying vehicles above urban areas. Even though air traffic is very safe (compared with ,for example, cars), with increasing numbers, accidents are bound to happen, and it is necessary to be prepared for them. The planner proposed in this work offers not only risk minimization but also a fast emergency-landing path selection in case of partial failure.

The proposed planner has two compact, distinct, and somewhat independent steps. First of those is the creation of an emergency landing map, which is discussed in Section 5.1. The second is the risk-aware path planner, which utilizes the emergency landing map to evaluate risks induced by non-fatal in-flight failures. The risk-aware path planner is discussed in Section 5.2.

5.1 Risk-Minimizing Emergency Landing Trajectory Planner

The approach to the emergency trajectory selection presented here is based on the works [40,41]. The planner is targeted for small unmanned aircraft and is easily deployable for urban planning scenarios, where the terrain and small-scale obstacles (such as buildings) play a significant role, and thus it is deemed most appropriate for the problem discussed in this work. The authors used an RRT*-based algorithm to plan emergency-landing trajectories from any location to one of a number of given previously determined safe landing sites. A crucial part of the approach is the assumption that the aircraft can quickly lose altitude if necessary. Therefore, the minimal possible altitude loss may be considered for any maneuver. If the resulting altitude is higher than the landing site is, the altitude may be lost at will. Thus, maximizing the altitude maximizes the flight range and the ability to avoid obstacles, and the excess altitude does not inhibit the flight in any way. As a result, the altitude component of any maneuver may be considered given and used as the cost of the RRT* nodes. This reduces the dimensionality and thus significantly reduces the computational demands. The safe emergency landing sites are then at the root of the tree. However, as a consequence, the planner cannot differentiate landing locations with different risks.

Each edge in the tree represents a path between two configurations, and from any location in the tree, a trajectory to a safe landing site is given by traversal of the edges to the root. A query to this tree then attempts to connect the provided configuration to the tree. If the configuration is successfully inserted, the safe emergency landing trajectory is extracted from the tree. If the connection fails, no known emergency landing trajectory exists, and the configuration is considered unsafe.

In this work, the original RRT*-based algorithm is altered using discretization of the configuration space $\mathcal{C}$, which allows for the use of precomputed trajectories, and for the caching of collision check results, significantly lowering the overall computational demands. With the method proposed here, it is computationally feasible to find emergency trajectories from any point on the grid defined by the discretization, and it is possible to consider any configuration on the grid as a potential emergency landing site. This means that this approach allows for the use of this method not only to determine the

safe altitude but also to determine the best possible emergency landing trajectory, even in case a safe landing area is unreachable.

The overall emergency landing trajectory algorithm is summarized in Algorithm 2, described in detail here (Sections 5.1.1 to 5.1.5). The result of this algorithm is an emergency landing tree, which can be queried to obtain an emergency landing trajectory from any configuration within the collision-free configuration space. The query is described in Section 5.1.5 and Algorithm 4.

The overall aim of the emergency landing tree is to present an efficient way of solving the Emergency landing trajectory Problem 1. This computation needs to be computationally inexpensive because. In case of loss of thrust, every second spent on the computation is a second a damaged aircraft continues flying aimlessly and losing altitude. In addition, if the emergency planner is supposed to be used in the risk-aware trajectory planner, the risk computation will have to be executed thousands of times. Thus any complicated procedures would be computationally infeasible. Therefore, the desired outcome is a structure that would allow for fast queries on any point in the configuration space $\mathcal{C}$.

As mentioned above, a similar problem was addressed in [41]. However, the problem studied therein is slightly different. The method proposed in [41] is only able to plan an emergency landing trajectory that leads to an airport, that is, a safe emergency landing site. The planner proposed in this work extends their work by considering unsafe landing locations, thus enabling the risk-aware planner to use the risk induced by partial failures to evaluate the configurations.

A straightforward implementation of the RRT* algorithm for a problem similar to the one studied in this work would use the risk of the landing site as the cost of the node, and whenever a new node is inserted, set its cost equal to the cost of its parent. However, using the risk as the cost means that the height of the node cannot be used as its cost, unlike in the approach from [41]. At every location, there are potentially multiple altitude levels that each have a different risk associated with them and thus cannot be represented by a single node. The state space would then have to be $\mathbb{R}^3 \times \mathbb{S}$, which in conjunction with the vehicle constraints represented by (3.1), (3.2) and (3.3) becomes computationally very demanding.

The approach presented in this work tackles this problem by introducing a regular discretization of the whole configuration space $\mathcal{C}$, an example of which is visualized in Figure 5.1. This allows for the use of precomputed maneuvers instead of computing every maneuver anew. Furthermore, this discretization also allows for the caching of the collision check results, further decreasing the computational complexity. With this approach, it is computationally feasible to not only use the fully dimensional configuration space $\mathcal{C}$ but also to evaluate all of the discretized nodes in the tree.

5.1.1 Tree Computation

The algorithm takes the collision-free part of the configurations space, $\mathcal{C}_{\text{free}}$, together with the terrain $\mathcal{T}$, and produces a graph G, where every vertex corresponds to a configuration $\boldsymbol{q} \in \mathcal{C}_{\text{free}}$, and an edge between two nodes represents a collision-free path between the two configurations.

The first step of the search tree computation is the discretization of the state space (Algorithm 2, Line 1). The state space is uniformly sampled along the x, y, and z-axis and along the dimension of the heading angle ψ, producing a set of samples $\mathcal{Q}$, forming a grid. For any configuration $\boldsymbol{q} \in \mathcal{Q}$, the risk is initially set to infinity. The sampling rate is further denoted by $\Delta d, \Delta z$, and k, which corresponds to the horizontal spacing, vertical spacing, and heading angle sample count, respectively.

Once the sampling is established, the maneuver pool Λ needs to be computed. Thanks to the discretization, each maneuver may be translated to another configuration and reused. Thus, it is not

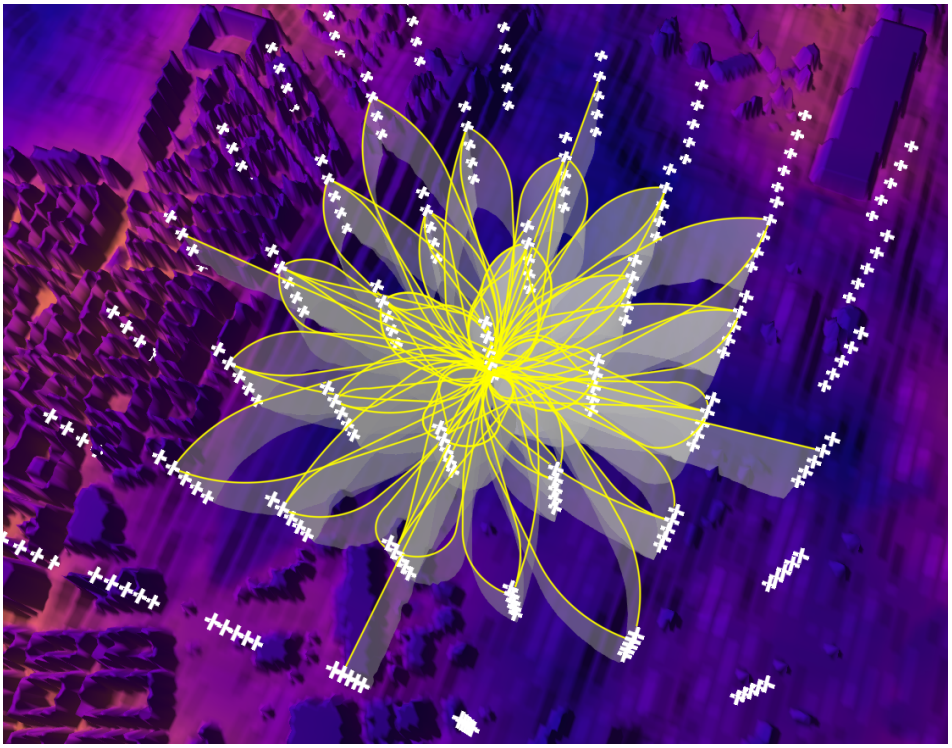


Figure 5.1: The proposed discretization of the configuration, together with an example of the maneuver pool. The visualized discretization has $\Delta d = 200$, $\Delta z = 20$ and $k = 4$. Each arm of a cross represents one of the configurations. Each of the yellow maneuvers represents one of the precomputed maneuvers in the pattern pool. For the purposes of this visualization, the height component is scaled up.

Algorithm 2: Risk Map Creation

Input: $\mathcal{C}_{\text{free}}$ – Collision-free configuration space.
Input: $\mathcal{T}$ – Terrain.
Parameter: m – Number of maneuvers.
Parameter: n – Number of nearest nodes.
Parameter: s – Number of emergency landing sites.
Parameter: $\Delta d, \Delta z, k$ – Sampling parameters.
Parameter: $\mathcal{A}$ – Aircraft model.
Output: Connection Graph G.

1 $\mathcal{Q} \leftarrow \text{UniformSampling}(\mathcal{C}_{\text{free}}, k, \Delta d, \Delta z)$
2 **for** $\boldsymbol{q} \in \mathcal{Q}$ **do**
3 $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}) \leftarrow \infty$
4 $\Lambda \leftarrow \text{ManeuverPool}(\mathcal{A}, m, k, \Delta d, \Delta z)$
5 $\mathcal{Q}_{\text{EL}} \leftarrow \text{SelectLandingSites}(s, \mathcal{T})$
6 $G \leftarrow \{\boldsymbol{V} \leftarrow \mathcal{Q} \cup \mathcal{Q}_{\text{EL}}, \boldsymbol{E} \leftarrow \emptyset\}$
7 **for** $\boldsymbol{\xi} \in \mathcal{Q}_{EL}$ **do**
8 $Q_{\text{n}} \leftarrow \text{Near}(\boldsymbol{\xi}, \mathcal{Q})$
9 **for** $\boldsymbol{q} \in Q_n$ **do**
10 **if** $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}) > \mathcal{R}_{\mathcal{F}}(\boldsymbol{\xi})$ **then**
11 **if** *notAdmissible* $((\boldsymbol{q}, \boldsymbol{\xi}), \mathcal{T})$ **then**
12 continue
13 $\text{Parent}(\boldsymbol{q}) \leftarrow \boldsymbol{\xi}$
14 $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}) \leftarrow \mathcal{R}_{\mathcal{F}}(\boldsymbol{\xi})$
15 **for** $\mathcal{Q}_i \in \mathit{Layers}(\mathcal{Q})$ **do**
16 **for** $\boldsymbol{q} \in \mathcal{Q}_i$ **do**
17 **for** $\Gamma \in \Lambda$ **do**
18 $\Gamma \leftarrow \text{SetTarget}(\Gamma, \boldsymbol{q})$
19 $\boldsymbol{q}_{\text{i}} \leftarrow \Gamma(0)$
20 **if** $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}_i) < \mathcal{R}_{\mathcal{F}}(\boldsymbol{q})$ **then**
21 continue
22 **if** *notAdmissible*$(\Gamma, \mathcal{T})$ **then**
23 continue
24 $\text{Parent}(\boldsymbol{q}_{\text{i}}) \leftarrow \boldsymbol{q}$
25 $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}_{\text{i}}) \leftarrow \mathcal{R}_{\mathcal{F}}(\boldsymbol{q})$
26 $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup (\boldsymbol{q}, \boldsymbol{q}_{\text{i}})$
27 **return** G

necessary to compute new maneuvers and their samples in order to connect a new configuration to the tree. Instead, a pool of potential flight patterns is precomputed at the beginning of the algorithm, and those are all the maneuvers used during the computation. This is done on Line 4, and the procedure is described in further detail in Section 5.1.2.

The next step of the initialization is the selection and connection of the potential emergency landing sites to the tree. The considered landing locations $\mathcal{Q}_{\text{EL}} \subset \mathcal{T}$ are selected on Line 5 based on the risk of an emergency landing in that configuration. In addition, a number of airport runways may be considered, which are added to the selected landing locations, and are considered safe in that

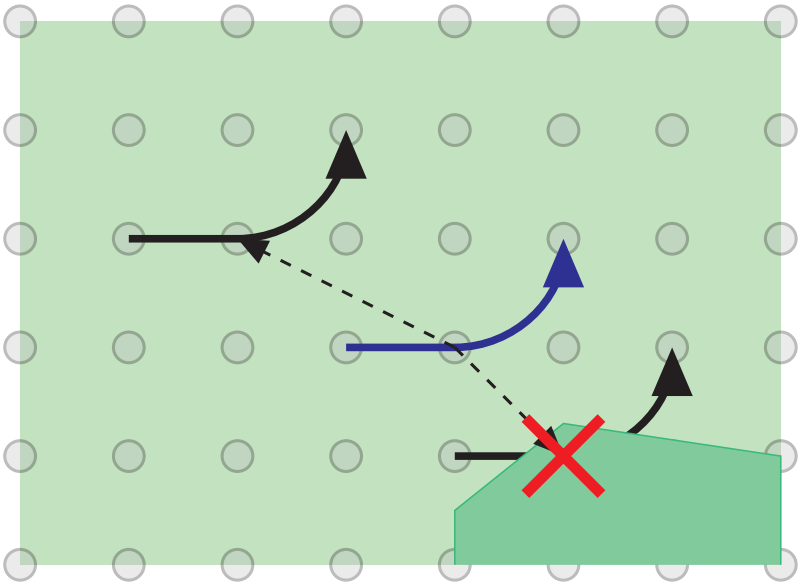


Figure 5.2: A simplified 2D example of the maneuver translation. Grey points represent the sampled configurations, blue arrow is the original maneuver, dashed lines represent the translation of the maneuver. Darker green represents area, where the terrain is in collision with the sampled configurations. As long as the original maneuver, and the translation are aligned with the sampling, so is the translated maneuver.

their risk is lowest among the potential landing locations. The selection process is discussed more in Section 5.1.3.

After the emergency landing locations are selected, it is necessary to connect them to the tree because they are not part of the sampled space. Therefore, they have to be explicitly connected, and the precomputed maneuvers cannot be used here. The connection itself is described on Lines 7 to 14. For any potential landing site configuration $q \in \mathcal{Q}_{\text{EL}}$, we try to connect it to all nearby samples in the tree. For each of those samples, a gliding trajectory is computed from the sampled configuration to the emergency landing configuration. If it is a feasible gliding trajectory between those two configurations, the risk associated with the landing site is propagated to the sample. Otherwise, the connection is impossible, and the trajectory is discarded.

After these steps are completed, the tree is initialized, and the nodes can be expanded using the precomputed maneuvers. Every maneuver $\Gamma \in \Lambda$ has the final configuration at the origin, that is, $\boldsymbol{q}_{\text{f}} = [0, 0, 0, \psi_{\text{i}}]$, and the initial configuration aligned with the sampling: $\boldsymbol{q}_{\text{i}} \in \mathcal{Q}$. If a node being expanded is $\boldsymbol{q} = [x, y, z, \psi] \in \mathcal{Q}$, each of the precomputed maneuvers may be translated by $T_{(x,y,z)}$, and the resulting maneuver $T_{(x,y,z)}(\Gamma)$ connects some configuration $\boldsymbol{q_0}$ to $\boldsymbol{q}$, and thanks to how the maneuvers are computed, it holds that $\boldsymbol{q_0} \in \mathcal{Q}$. If the path $T_{(x,y,z)}(\Gamma)$ is collision-free, then the node $\boldsymbol{q_0}$ is added to the tree, and the whole computation requires only simple integer arithmetic and the collision check, which can be further simplified, as discussed later in Section 5.1.4. A simplified 2D example of the maneuver translation is visualized in Figure 5.2.

The expansion is done in layers of the same height, from the lowest to the highest, as shown in Lines 15 to 26. Since none of the maneuvers can gain height (a restriction imposed by the failure), it is impossible for any maneuver to improve any of the samples in the same or lower height layer. Thus, if the algorithm proceeds from the ground level upward, all possible nodes are expanded, and the risk is (with respect to the maneuver pool) minimized.

For every node with finite risk in the layer being expanded, all the precomputed maneuvers are con-

sidered. Each maneuver is translated so that the $\boldsymbol{q}_{\mathrm{f}}$ is located at the location of the node, and then the initial configuration of the maneuver $\boldsymbol{q}_{\mathrm{i}}$ is checked. If the risk of the node at the initial configuration is greater or equal to the risk already found for the target sample, the maneuver offers no improvement. Otherwise, a collision check for that maneuver is performed. If there is no collision along the maneuver, the risk is propagated to the higher node ($\boldsymbol{q}_{\mathrm{i}}$ of the maneuver).

5.1.2 Pattern Pool generation

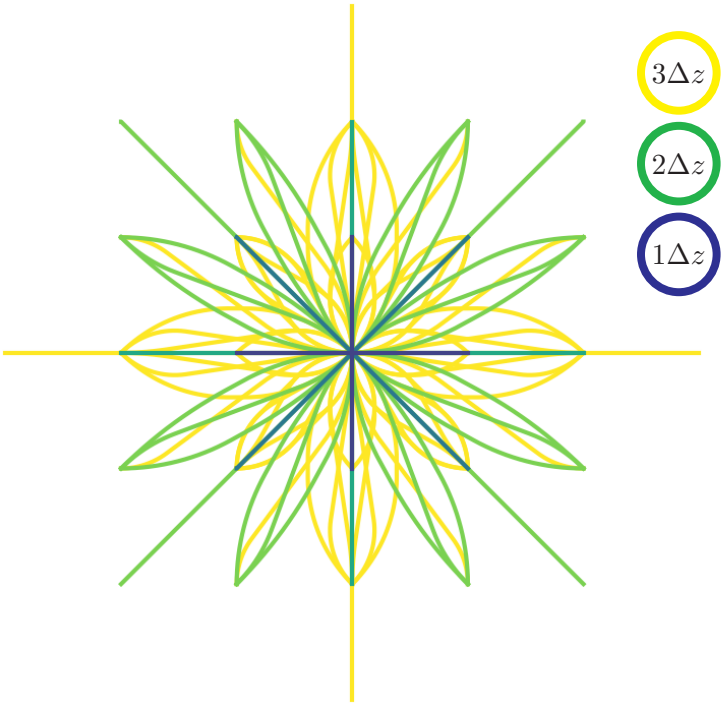


Figure 5.3: An example of the computed maneuver pool. This maneuver pool contains 108 maneuvers, generated with 8 heading samples, maximum distance of $3\Delta d$. Each altitude difference layer is distinguished by different color, lighter colors represent greater altitude level, and every line corresponds to one maneuver. Final configuration of each of the maneuvers is in the center.

The precomputed maneuvers are an essential part of the algorithm, as they determine the optimality of the result, and also because the computational complexity scales linearly with the number of considered maneuvers. Each of those maneuvers represents a flight pattern, which the damaged aircraft can execute. Together, those flight patterns form a pattern pool, from which the maneuvers during the expansion are taken, and no other maneuvers are considered. An example of a pattern pool is shown in Figure 5.3.

Each of the generated flight patterns has to be aligned with the grid. The final configuration $\boldsymbol{q}_{\mathrm{f}}$ is at the origin, while the initial $\boldsymbol{q}_{\mathrm{i}}$ configuration is at a multiple of the sampling size: $x = k_1\Delta d, y = k_2\Delta d$, where $k1, k2 \in \mathbb{Z}$. The heading angle ψ of each of those configurations is also set according to the sampling, $\psi = \frac{k_3}{k}2\pi$, for some $k_3 \in \mathbb{Z}, 0 \leq k_3 < k$. The maneuver connecting the two configurations is computed, and the optimal height of the initial configuration $\boldsymbol{q}_{\mathrm{i}}$ is determined based on the height of the maneuver. However, since the $\boldsymbol{q}_{\mathrm{i}}$ has to be aligned with the sampling, the height also has to be rounded up to a multiple of the sampling rate Δz so that the initial configuration is aligned with the uniform sampling.

In addition to the maneuver with the minimal possible height, all of the maneuvers obtained by scaling the altitude component up to the limit of the airplane are also considered. This allows the paths with a higher-than-minimal descent rate to be constructed, which connect the configurations in higher altitudes.

5.1.3 Emergency Landing Site Selection

Algorithm 3: Emergency Landing Site Selection

Input: $\mathcal{T}$ – Terrain.
Input: n – Number of emergency landing sites.
Output: Selected landing sites $\mathcal{Q}_{\text{EL}}$.

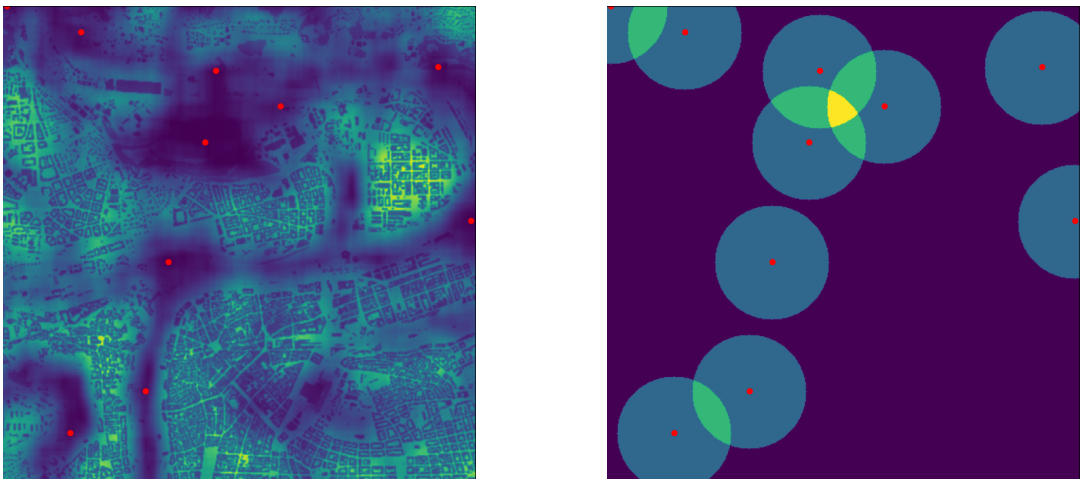
Function `SelectLandingSites`$(\mathcal{T}, n)$**:**
  $\mathcal{Q}_{\text{EL}} \leftarrow \emptyset$
  $\mathcal{Q}_{\text{candidates}} \leftarrow \mathcal{T}$
  for $i = 1...n$ **do**
    $\boldsymbol{q}^* = \text{argmin}_{\boldsymbol{q} \in \mathcal{T}} \mathcal{R}(\boldsymbol{q})$
    $\mathcal{Q}_{\text{EL}} \leftarrow \mathcal{Q}_{\text{EL}} \cup \boldsymbol{q}^*$
    $\mathcal{Q}_{\text{near}} \leftarrow \text{Near}(\boldsymbol{q}^*)$
    $\mathcal{Q}_{\text{candidates}} \leftarrow \mathcal{Q}_{\text{candidates}} \setminus \mathcal{Q}_{\text{near}}$
  return $\mathcal{Q}_{EL}$

The last big subtask of the emergency landing trajectory tree computation left to address is the landing site selection. For the initialization of the tree, it is necessary to select appropriate landing sites so that as much of the configuration space $\mathcal{C}$ is covered and the risk is as low as possible. The emergency landing sites $\mathcal{Q}_{\text{EL}}$ form a subset of the terrain: $\mathcal{Q}_{\text{EL}} \subset \mathcal{T}$. This is similar to the Maximal Covering Location problem [46], in which the goal is to select a limited number of facilities, such that the number of covered customers is maximized. Each potential emergency landing site can be seen as a facility with cost as a function of its risk, and the samples are analogous to the customers.

However, there are two issues. The first issue is that determining which part of the state space is covered would have to solve the full emergency landing tree for that landing site. The second issue is that the set Maximal Covering Location problem is NP-hard.

Since the space covered by the potential landing location is unknown, an estimate is used instead. Any emergency landing will be outclassed at a certain altitude level in that a path exists to an airport runway or to a better emergency landing site. Because of that, the most important configurations to cover are the closest ones to the terrain level. Therefore, instead of trying to approximate the covered space, the whole problem is approximated by its projection to the xy plane.

The greedy algorithm for this variant of the Maximal Cover Location problem is described in Algorithm 3. Firstly, the risk value of each of the terrain configurations is computed. Then, the least risky location is selected, and all locations within a certain radius around it are removed from the selection pool. This step is repeated until enough landing sites have been selected or until the pool is empty. The radius and the target landing site count are hand-assigned parameters of this algorithm. An example of the result of this algorithm is depicted in Figure 5.4, while Figure 5.4b depicts the radii around selected locations that have been removed. As has been shown in [46], a greedy algorithm is an approximation algorithm for Set Cover, Maximal Covering Location, and similar problems, and the algorithm for this specific task does not exhibit any severely sub-optimal results.



(a) 10 selected landing sites

(b) Visualization of the areas blocked to further selection by already selected landing locations

Figure 5.4: Visualization of the selection of potential landing locations, based on the risk map and proximity to already selected landing locations. First image shows 10 selected landing locations, and the risk map, while the second image shows the radii around the already selected landing locations. Red dots represent the selected landing locations. Lighter areas represent areas, where an emergency landing would pose high risk to people, and darker areas represent lower risk.

5.1.4 Collision Check

In order to determine whether a maneuver is feasible, it is necessary to check whether the maneuver is collision-free. The common approach to a collision check of a maneuver is to sample points along the maneuver. Then each of those samples is compared with the altitude level and obstacles at that location to determine whether they are in collision. However, the total number of collision evaluations may end up being quite high. For example, if there are 200 maneuvers in the pool, 50×50 samples in the xy plane, 8 heading angle samples, and 30 height samples, the number of maneuver collision checks performed would be around 1.5 million.

As previously stated in Chapter 3, in this work, the terrain $\mathcal{T}$ is considered to have no overhangs: for any configuration $\boldsymbol{q} \in \mathcal{C}_{\text{free}}$, any configuration $\boldsymbol{q}_{\text{a}}$ directly above it, it holds that $\boldsymbol{q}_{\text{a}} \in \mathcal{C}_{\text{free}}$. Consequently, since we have only a limited pool of maneuvers we have to perform the collision check on, it is possible to cache the results of this operation. For a given maneuver and given location, once a collision-free altitude level is found, any collision checks for that maneuver at a higher altitude must necessarily also be collision-free.

A collision check lookup table is used, which holds the minimum known collision-free height of every maneuver for every xy location and heading angle sample. Since the risk propagation proceeds from the ground level upwards, once a maneuver translated to a location is found to be collision-free, no further collision checks in this location will be necessary.

This is a trade-off between memory and time complexity. The lookup table saves a lot of computation, but its size may be even greater than that of the tree itself, depending on the size of the maneuver pool.

Algorithm 4: Get Emergency Landing Trajectory

Input: q' – Query configuration.
Input: G – Risk map.
Input: $\mathcal{T}$ – Terrain.
Parameter: n – Number of nearest nodes.
Output: Risk $\mathcal{R}_q(q')$.
Output: Emergency landing trajectory Γ.

1 **Function** `GetEmergencyLanding`$(q', G, \mathcal{T})$**:**
2 $Q_n \leftarrow \text{Near}(q', G)$
3 $q^* \leftarrow$ nothing
4 **for** $q \in Q_n$ **do**
5 **if** *isAdmissible*$((q', q), \mathcal{T})$ **and** $\mathcal{R}_q(q) < \mathcal{R}_q(q^*)$ **then**
6 $q^* \leftarrow q$
7 $\Gamma \leftarrow \emptyset$, $q \leftarrow q^*$
8 **while** $q \notin \mathcal{Q}_{EL}$ **do**
9 $\Gamma \leftarrow \Gamma \cup (q, \text{Parent}(q))$
10 $q \leftarrow \text{Parent}(q)$
11 **return** $\mathcal{R}_q(q^*)$, Γ

5.1.5 Emergency Landing Query

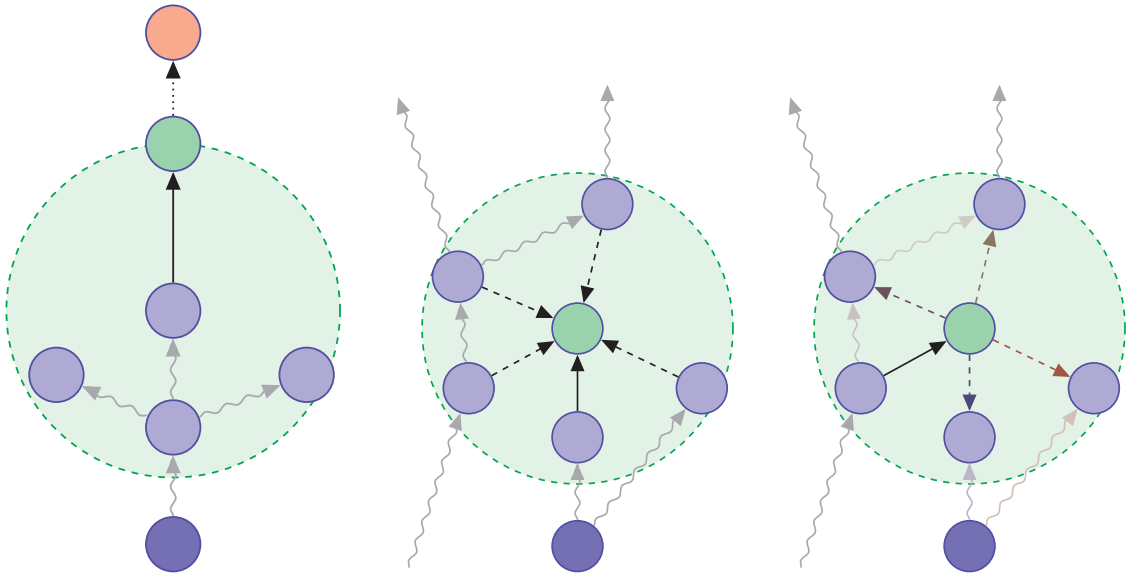
Once the tree computation is done, the structure can be used to determine the risk induced by a non-fatal failure and to plan an emergency landing trajectory in case of such a failure. The procedure is very similar to the query in [41]. The query is described in Algorithm 4.

In order to determine the emergency landing trajectory, it is necessary to connect the configuration to the tree. This is done by attempting a connection to every near node already connected to the tree. The best connection (in terms of the risk) is kept. This part is described on Lines 4 to 6, Algorithm 4. Once such a connection exists, the trajectory is obtained by recursively following the connections to the parent nodes up to the root, as described on Lines 8 to 11, Algorithm 4. The risk of the configuration is then equal to the risk of the node to which it is connected. The risk of that node is, in turn, equal to the risk of the landing location to which the emergency trajectory leads.

5.2 Risk-Aware Planner

The overarching motivation behind the emergency landing path planner is to use it in a Risk-Aware path planner in the risk assessment for partial failures. The proposed planner is an altered variant of the planner from [35], that includes the emergency landing planner in place of the original safe emergency landing planner. This section explains how the planner works and how the proposed emergency planner is included in the planner.

The first step of the risk-aware planning algorithm is to create the emergency landing map introduced earlier in this chapter in Section 5.1. The emergency landing tree is then used to find feasible emergency landing maneuver for any given configuration and to determine the risk associated with the emergency landing. After the emergency landing tree is constructed, the RRT* planner itself is executed. The resulting trajectory then minimizes the risk induced by any in-flight failure, both fatal failure, which results in a ballistic fall, and a partial failure, which leaves the airplane capable of controlled flight with limited maneuverability. The risk associated with a fatal failure may be any risk



(a) Insertion step of RRT* and the steering procedure. The randomly generated node in red.

(b) The first part of the rewiring step

(c) The second part of the rewiring step

Figure 5.5: The sequence of steps executed during insertion of a new node to the RRT* tree. Green node represents the newly added node, dashed arrows represent considered connections, wavy arrows represent the original state of the tree.

function that estimates the risk for a given configuration. The full risk function used in this work is described in Section 5.2.1.

The risk-aware planner incrementally builds an RRT* planning tree in the same way as described in Algorithm 1. A new node is randomly generated, and if it is admissible, it is inserted into the tree by connection to the nearest node already in the tree. A node is considered admissible if it is not in collision. If the randomly generated configuration is too far from the nearest node, the steering procedure generates a new one instead, as described on Line 5 in Algorithm 1. The originally generated node is replaced by a configuration sampled on the connecting path closer to the selected parent node. This configuration and the path connecting it to the parent node are inserted instead of the original randomly sampled configuration. This step is visualized in Figure 5.5a.

Once a node is successfully inserted, the algorithm checks for any potential improvements to the tree it might enable. Two types of improvements might be possible. Firstly, among the near nodes, there might be a better parent node for the newly inserted node available, as shown in Figure 5.5b. This is checked by creating the connection from each of the near nodes and evaluating the risk of this connection. If the risk of the connection plus the risk of the potential parent node is lower than the current risk of the new node, the better parent is assigned. The second potential improvement is that the newly added node offers a better path to a node already in the tree. The rewiring procedure checks whether the newly added node can yield lower risk if assigned as a parent to any of the near nodes. If any such nodes exist, the new node replaces the original parent, and the risk for those nodes is recalculated and propagated to any potential children nodes. A comparison of the connections to the near nodes is depicted in Figure 5.5c.

5.2.1 Risk Estimate

The risk-aware trajectory planner needs some risk metric to evaluate its trajectories. The metric used in this work is similar to the one used in [35], which is the number of people killed. Given a trajectory $\Gamma : [0,1] \rightarrow \mathcal{C}_{\text{free}}$, the risk induced by total loss of control resulting in an uncoordinated crash is formulated as follows:

$$\mathcal{R}(\Gamma) = \int_0^1 \mathcal{R}_0(\Gamma(t))\,\mathrm{d}t, \tag{5.1}$$

where $\mathcal{R}_0(.)$ is some risk function $\mathcal{R}_0 : \mathcal{C} \rightarrow \mathbb{R}$, which quantifies the risk at a given configuration.

The risk is calculated as the number of people killed in case of a failure. It needs to be calculated for every considered failure separately to reflect the probability of such failure occurring. Two types of failures are considered, the risk induced by a fatal failure, resulting in an uncontrollable ballistic fall, and a partial failure, which prevents the aircraft from gaining altitude and inhibits its maneuverability options. The set of considered partial failures is denoted $\mathbb{F}$, and $\mathcal{F}$ denotes a failure (both fatal and partial). Both types of failures are described in further detail in Chapter 4, and the following paragraphs describe how the risk is computed.

For any given configuration $\boldsymbol{q}$, the risk induced by a failure is determined based on the location the aircraft crashes on, the impact energy E, and the impact angle γ. When a failure $\mathcal{F}$ occurs, the risk it causes is then given by

$$\mathcal{R}_{\mathcal{F}}(\boldsymbol{q}) = \int_{\mathcal{T}} p_{\text{imp}}(\boldsymbol{x}|\mathcal{F},\boldsymbol{q})\,\text{Causalities}(\boldsymbol{x},E,\gamma)\,, \tag{5.2}$$

where Causalities is a function determining the number of casualties based on the location, impact energy, and impact angle. The ground risk function Causalities is determined based on [50]. The probability of hitting a person is estimated as $\text{Density}(x,y)\,A_{\text{exp}}$, where Density denotes the population density, and A_{exp} denotes the area exposed to the crash. This is then multiplied by the probability of the impact being fatal, which leads to $\text{Casualties}(\boldsymbol{x},E,\gamma) = \text{Density}(\boldsymbol{x})\,A_{\text{exp}} \cdot p_{\text{fatal}}$, where is the fatality rate p_{fatal} computed from the sheltering factor, and the impact energy $E = \frac{1}{2}m \cdot v_{\text{imp}}^2$.The exposed area is computed from the size of the aircraft, from the average person's height, and from the angle of descent during the crash.

For any $\mathcal{F} \in \mathbb{F}$, the location of the impact is the target location of the selected emergency landing trajectory. Therefore there is no uncertainty, and (5.2) simplifies to $Casualties(\boldsymbol{x},E,\gamma)$, where $\boldsymbol{x}$ is the selected landing location, $E = \frac{1}{2}m \cdot v_{\text{imp}}^2$ is computed from the speed of the aircraft (which is considered constant within the maneuver), and γ is estimated as the optimal gliding angle.

Following a critical failure, the airplane continues on a ballistic trajectory until it hits the ground. The uncontrolled fall is described by the equation $m\dot{\boldsymbol{v}} = m\boldsymbol{g} - \frac{1}{2}c\rho S\|\boldsymbol{v}\|\boldsymbol{v}$, where $m\boldsymbol{g}$ denotes the gravity vector and $\frac{1}{2}c\rho S\|\boldsymbol{v}\|\boldsymbol{v}$ denotes the Newton's drag force. However, the exact location of the impact is impossible to determine because there are too many factors at play, and the equation cannot be exactly evaluated. Therefore, an impact probability map based on the ballistic fall is used instead. The uncontrolled fall formulation has been adopted from [35].

5.2.2 Using the Emergency Landing Planner

The risk-aware planner with safe emergency landing guarantee, as presented in [35], computes trajectories that cannot enter areas that do not have a safe emergency landing site sufficiently close. However, this might be too restrictive. If there are not enough safe emergency landing sites, or if several non-fatal failures are to be considered, the planning space above the safe altitude might become

too constrained. Therefore, it is desirable that the risk-aware planner might consider even trajectories that are not completely safe with respect to partial failure. However, the overall risk of the trajectory must be minimized, even including risk induced by partial failure.

The emergency landing trajectory planner proposed in this work offers a way to evaluate the risk induced by partial failures. For any given configuration, it provides an emergency landing trajectory that is safest within the limits of the computed emergency trajectory tree. If the examined configuration is above the safe altitude, the query will return a safe landing trajectory. If, however, the configuration is below the safe altitude, the query will still return an emergency landing trajectory that minimizes the induced risk. If this emergency landing planner is employed, the safe altitude restriction is removed from the admissibility check, and the risk induced by partial failure is computed together with the risk induced by fatal failure.

The new risk function retains the risk caused by the total failure and adds the risk induced by the partial failure on top of that, as described in (3.5) (repeated here for the sake of completeness),

$$\mathcal{R}\left(\boldsymbol{q}\right) = \sum_{\mathcal{F}\in\mathbb{F}} \left(p_{\mathcal{F}}\mathcal{R}_{\mathcal{F}}\left(\boldsymbol{q}\right)\right) + p_{\text{fatal}}\mathcal{R}_{\text{q}}\left(\boldsymbol{q}\right), \tag{5.3}$$

where $\mathcal{R}_{\text{q}}\left(.\right)$ is the risk of a fatal failure, $\mathcal{R}_{\mathcal{F}}$ is the risk induced by partial failures both defined by (5.2), and $p_{\mathcal{F}}$ is the probability of failure $\mathcal{F}$, estimated based on the publicly available aviation accident data [6].

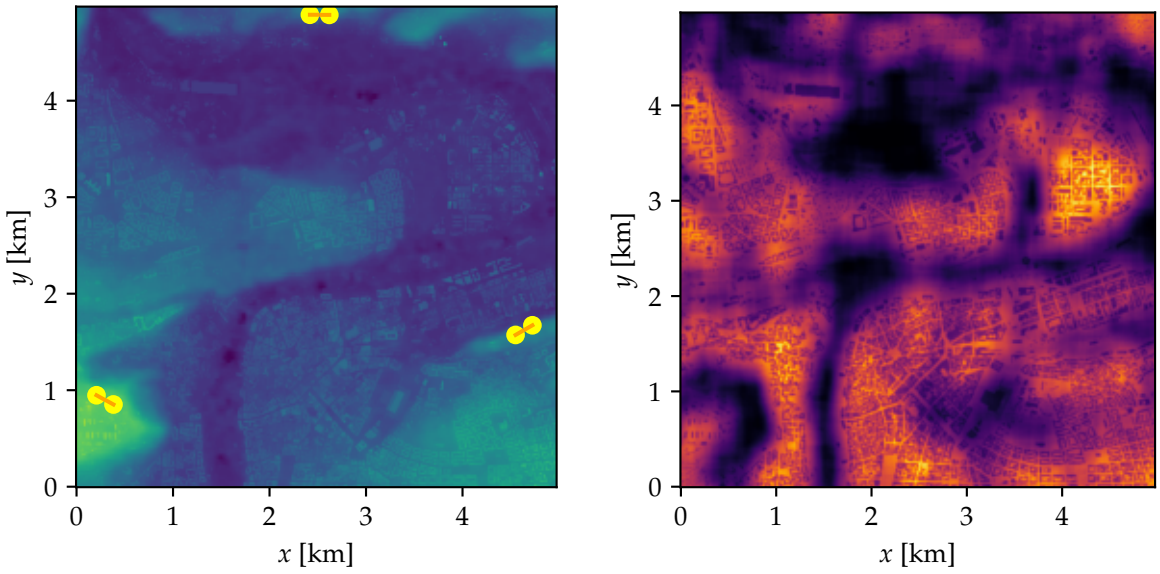
The risk associated with partial failure $\mathcal{R}_{\mathcal{F}}\left(.\right)$ is computed using the emergency landing tree presented in this work. However, the query presented in Algorithm 4 is still quite computationally demanding, and the number of times it needs to be computed is so high that the total computational requirements of the risk planner become too excessive, especially for denser trees. When one node is added to the risk-planning tree, the number of trajectories that need to be evaluated is given by the number of nodes returned as near during the initial parent selection plus the number of nodes considered during the rewiring procedure. For each of those trajectories, the risk assessment has to be called at least Δd intervals, where Δd is the sampling rate. This leads to hundreds of risk evaluations to add a single node. Even though it is feasible, the computational times would be hours instead of minutes. Therefore, a simplified variant of Algorithm 4 is used, which estimates the risk associated with the given configuration by considering the risks of the closest configurations in the emergency landing tree. The simplified query is presented in Algorithm 5, and its accuracy is further discussed in Chapter 6. The query searches all directly neighboring configurations in the tree. The neighboring nodes are considered in all dimensions and up and down, resulting in (up to) 16 different samples. Since all of those nodes are in the tree, their risk is known. The maximum among their risks is considered as the estimated risk value of the node. The Neighbors procedure returns all the direct neighbors in every dimension to each side, which is defined as $\left\{\boldsymbol{q}_{\text{n}} = [x_{\text{n}}, y_{\text{n}}, z_{\text{n}}, \psi_{\text{n}}] \mid \boldsymbol{q}_{\text{n}} \in G, |x - x_{\text{n}}| < \Delta d, |y - y_{\text{n}}| < \Delta d, |z - z_{\text{n}}| < \Delta z, |\psi - \psi_{\text{n}}| < \frac{2\pi}{k}\right\}$, that is, every configuration in the tree, which has each of its components closer than the sampling to the queried configuration.

This allows for a computationally very efficient emergency landing risk estimate that can be used during the planning, and in case of an actual in-flight failure or in advance once the risk-planning is completed, the full query described in Algorithm 4 can be executed only on the configurations, that the aircraft actually visits. The number of queries necessary to cover a planned path of length 5 kilometers, considering sampling of 50 meters (a similar value to the one considered for the insertion computation), is then 100, which is still much less than the insertion of even a single node to the RRT* tree would need.

Algorithm 5: Get Emergency Landing Risk

Input: $\boldsymbol{q} = [x, y, z, \psi]$, – Query configuration.
Input: G – Risk map.
Input: $\mathcal{T}$ – Terrain.
Output: Risk $\mathcal{R}_{\mathcal{F}}(\boldsymbol{q})$.

Function `GetEmergencyLandingRisk` $(\boldsymbol{q}, G, \mathcal{T})$ **:**
    $Q_{\text{n}} \leftarrow$ Neighbours $(\boldsymbol{q})$
    $\mathcal{R} = \max_{\boldsymbol{q} \in Q_n} \mathcal{R}_{\mathcal{F}}(\boldsymbol{q})$
    return $\mathcal{R}$



(a) Altitude map

(b) Risk map

Figure 6.1: Visualization of the used testing data. Darker colors represent lower values. The altitude map includes obstacles and buildings. Yellow dots represent the considered safe emergency landing locations. The risk data is based on sheltering factor and population density.

Chapter 6
Results

This work presents a risk-aware trajectory planner, which minimizes the risk induced by in-flight failures. Two types of failures are considered: a fatal failure, which results in an uncontrollable crash, and a partial failure, which inhibits the maneuverability of the aircraft, but the aircraft is still controllable, and an emergency landing is possible. The planner utilizes an emergency landing tree, which is used to determine an emergency landing trajectory following a partial failure. Such a tree is constructed for every different considered partial failure. During the planning step, the tree is used to estimate the risk associated with each failure, and after the path is constructed, it can be used to determine the optimal emergency landing trajectory at any given point.

The presented method was evaluated on a number of different configurations of the planner. This chapter presents a summary of the achieved results, an analysis of the computational complexity, and visualizations and examples of the computed trajectories. It is separated into two parts. Section 6.1 describes the properties of the emergency landing planner, while Section 6.2 handles the evaluation of the risk-aware planner.

The data used for the computation and risk assessment are taken from publicly available databases. The map is taken from OpenStreetMap [51], terrain data from [52], the population density from [53], and the sheltering factor data is computed using the method proposed in [50]. The map encompasses a $5\,\mathrm{km} \times 5\,\mathrm{km}$ square above the Prague city center. In this area, three safe landing locations are considered, each approachable from two different directions. Areas considered as completely safe during the planning phase (shown as yellow dots in Figure 6.1a) do not correspond to actual airports,

but they are selected in areas with low population density and low risk overall, based on an estimate of locations, which could in the future hold facilities for UAV landing. This selection is arbitrary, and the computational complexity of the planner should not be affected. The altitude map and the computed risk map are shown in Figure 6.1a and Figure 6.1b, respectively. The parameters of the aircraft model are based on the Cessna 172, a common small aircraft (adopted from [40]). The proposed method has been implemented in the Julia programming language [54], and the experiments were performed on a single core of the Intel Xeon Scalable Gold 6146 CPU.

6.1 Emergency Landing Tree

The emergency landing tree proposed in this work utilizes a discretization of the configuration space, which allows it to greatly simplify the individual steps of the RRT* planning algorithm. This part of the chapter discusses the properties of this approach. An analysis of the quality of the resulting emergency landing trajectory tree is provided by means of comparison with the safe altitude computed by the method proposed in [41], further denoted as reference solution. However, the reference solution offers only safe altitude assessment, as opposed to the proposed method, which can plan and evaluate emergency paths in situations where a safe emergency landing is not possible. To the best of the authors' knowledge, there is no more appropriate method that could be used as a reference point.

The evaluation of the emergency landing tree computation has been performed for a failure of total loss of thrust. This leads to pitch angle limitation $\theta_{\max} \approx -4.9^\circ$ for a straight path segment, and $\theta_{\max} \approx --13.1^\circ$ for the minimal turning radius turn.

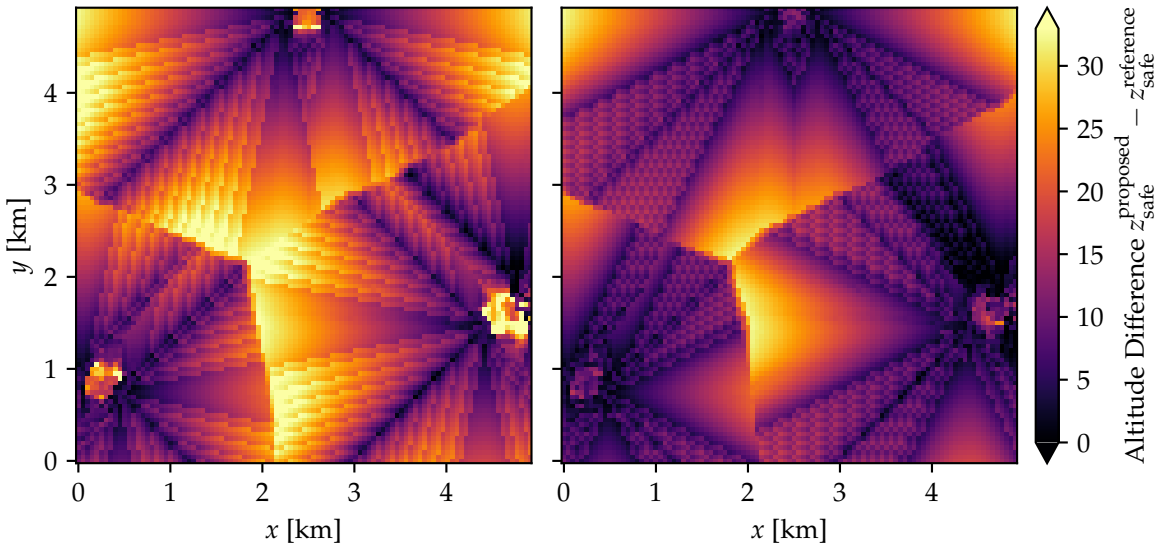
Thanks to the discretization, the whole algorithm is fully deterministic. The influence of the user-selected parameters of the discretization and the algorithm as a whole is presented as well. Lastly, the computational complexity of the algorithm is discussed. Both time complexity and memory requirements are considered, as the emergency landing tree can become quite large.

6.1.1 Quality of the Solution

The quality of the solution is evaluated by comparison with the reference solution [41]. The safe altitude level used for the visualizations is determined by taking the minimum of the safe altitude over all the considered heading angles, both for the reference solution and for the proposed method. For the proposed method, exclusively samples on the discretized grid were considered.

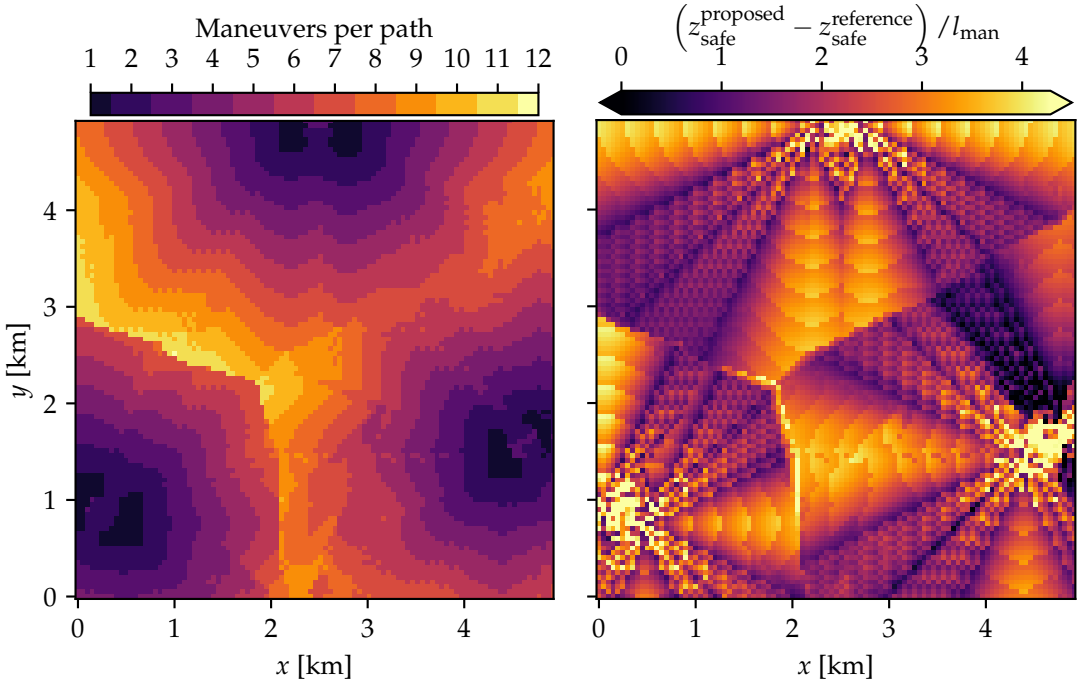
The reference solution provides asymptotically optimal solutions, as it uses the RRT* algorithm, and, in terms of the safe altitude, any maneuver added into the tree is optimal. In comparison, even though the proposed method is also based on the RRT* algorithm, it uses maneuvers that have sub-optimal altitude profile, that is, maneuvers, which could be executed with lower altitude loss. This is a concession necessary for the discretization to work. Consequently, in comparison to the reference solution, the proposed method yields slightly worse results, as is shown in Figure 6.2. However, the proposed method is able to minimize the risk of emergency landing trajectories that are below the safe altitude level, which the reference solution is unable to do.

Figure 6.2 shows the difference between the proposed method and the reference solution. Every value represents the difference between the two altitude levels in meters. Note that any values above 30 are not differentiated. At some values, the proposed method was unreasonably high, which corresponds to edge case locations caused by the safe airport being very close. This is likely because the assumption that the aircraft can lose altitude at will allows the reference solution to find trivial paths. In contrast, the proposed method searches for maneuvers satisfying the altitude component constraints, which may



(a) Heading samples $k = 8$ (b) Heading samples $k = 16$

Figure 6.2: Comparison of the safe altitude levels by the proposed algorithm and by the reference solution. Because the discretization of the proposed method forces sub-optimal maneuvers, in terms of the safe altitude the proposed method is universally slightly worse. The figures depict the results computed with parameters $\Delta d = 50\,\text{m}$, $\Delta z = 5\,\text{m}$, and the pattern pool area of $300\,\text{m} \times 300\,\text{m} \times 30\,\text{m}$. Lower detail is the result with $k = 8$, and higher detail with $k = 16$



(a) Length of the paths (b) The relative optimality gap

Figure 6.3: Optimality with respect to the maneuver length. Each maneuver adds some unnecessary altitude loss, because the altitude component has to be aligned with the sampling. The total unnecessarily lost altitude can be estimated based on Δz and the number of utilized maneuvers, which is shown in Figure 6.3b. Notice, that a vast majority of the values is below Δz. The number of used maneuvers per emergency landing path is displayed in Figure 6.3a, while the relative optimality is displayed in the right figure. Notice, that the relative values are overall very low and radially almost uniform. Computed for the tree shown in Figure 6.2b.

lead to helical paths or similar maneuvers.

The suboptimality is, to some extent, predictable because it is almost linear with respect to the number of used maneuvers and the height of the tiles Δz, as is shown in Figure 6.3difference. Each maneuver introduces a little bit of unnecessary altitude loss. The optimality gap of the resulting path is at least the same as the combined optimality gap of the individual maneuvers. The exception is when there are obstacles that the aircraft must fly over, which thus increase the minimal safe altitude. The unnecessarily lost altitude during a single maneuver is not greater than Δz, because if it was higher, a lower altitude maneuver would also be feasible. However, in the context of the full path, the suboptimality might be worse because the discretization forces the path to follow a sequence of waypoints, which are unlikely to be the optimal path. This is illustrated in Figure 6.3, which displays the length of the emergency landing paths measured in number of used maneuvers and compares the optimality of the path with this metric. Notice that the values in Figure 6.3b are radially almost uniform and that the majority of the relative values are only around three.

The comparison of the safe altitude levels depicted in Figure 6.2 exhibits some other interesting patterns. The whole image is visually divided into three areas, which correspond to the three safe emergency landing sites that are considered in this specific scenario. Each of the emergency landing paths leads to one of those safe landing sites, which creates the gradient pattern.

More interesting patterns emerge within those sectors as well. The suboptimality of the proposed method is caused by the precomputed maneuvers being suboptimal in terms of the altitude lost, which is a concession necessary to employ the discretization. Similarly, the maneuvers connecting the potential landing sites to the discretized grid are (likely) suboptimal. However, some of the maneuvers are better than others, and thus the paths utilizing the better maneuvers are in terms of the altitude closer to the reference solution. This is obviously greatly dependent on the selected attributes, however, not necessarily in a straightforward manner. It is not even safe to say that decreasing the vertical sampling rate Δz will improve the solution because if a maneuver was near-optimal at a certain sampling rate, further decreasing the sampling rate would make it worse (unless the lower sampling rate is a multiple of the higher one).

In the case shown in Figure 6.2b, the most optimal straight maneuvers are the longest possible maneuvers and the diagonal ones and the ones in the direction $\frac{1+2n}{8}\pi$, for some $n \in \mathbb{N}$ (the diagonal maneuvers have slightly worse optimality gap, but are also slightly longer). Any locations reachable by paths that lead in this direction will have a safe altitude closer to the reference solution than others. The bigger the deviation from this direction, the greater the difference to the reference solution. This creates the diagonal patterns visible in Figure 6.2. Additionally, even shorter maneuvers in this direction are worse. This creates the repetitive patterns along the diagonal lines and the cross-hatching-like quality of some of the sectors. Compare this with Figure 6.2a, which shows the same comparison, but for emergency landing tree computed with lower heading angle sample count. Areas, which are near-optimal in the higher-resolution tree, are among the worst in the lower-resolution tree because the maneuvers in their direction are not considered. Note that this is a consequence of the maneuver selection, and different parameters would yield very different patterns.

6.1.2 Multiple Considered Landing Locations

The improvement the proposed method offers in comparison to the reference solution is that it is computationally feasible to consider multiple landing locations with different associated risks. This is important because in most urban areas, dedicated safe landing locations, such as airports, are not usually very common and tend to be further out from the center of the city. If it is required that the

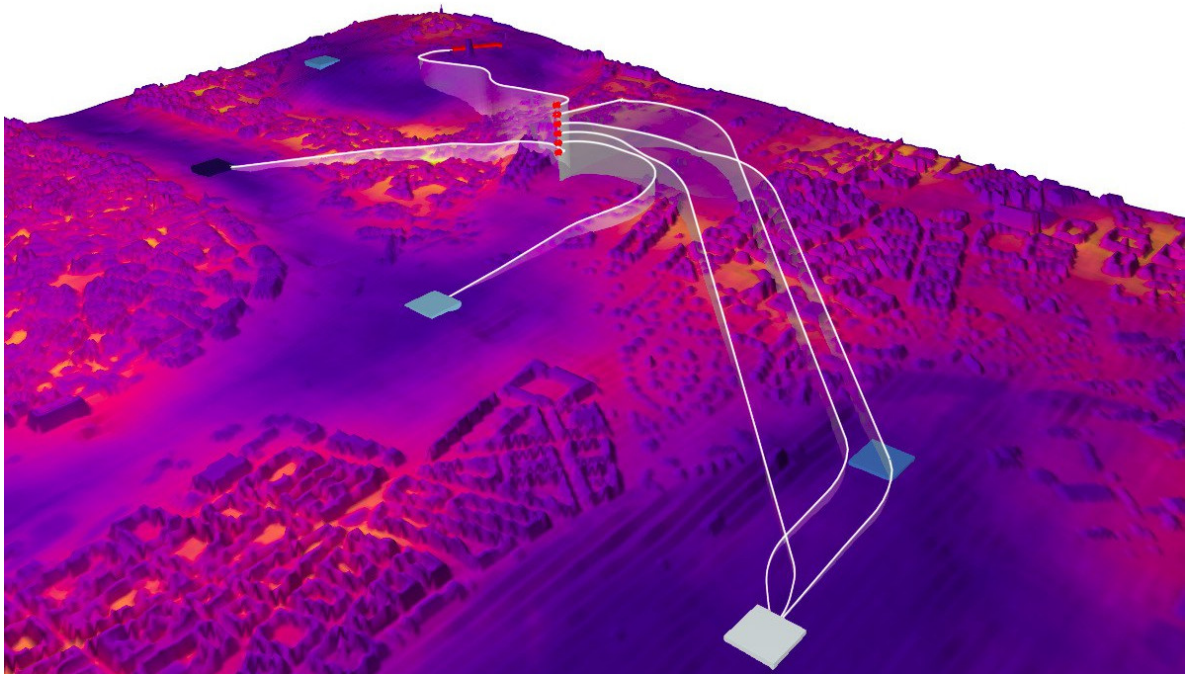


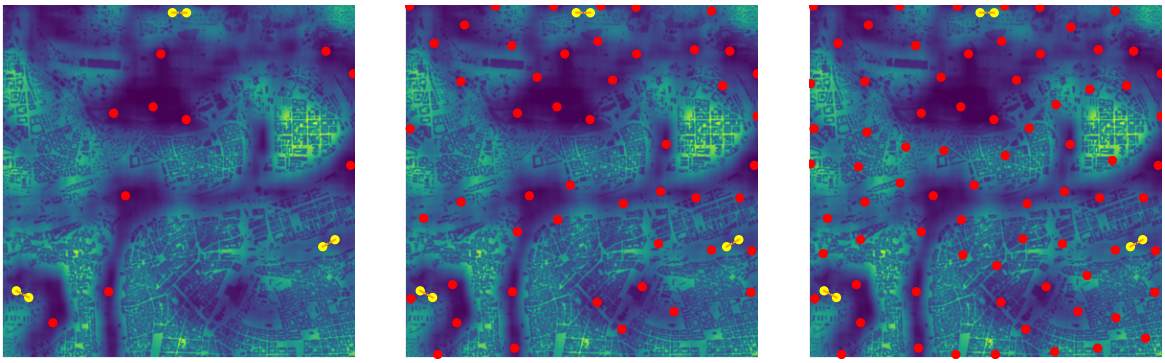
Figure 6.4: Examples of resulting trajectories, returned by the emergency landing query on a sequence of configurations, that differ only in their height. Red line represents a safe landing site, squares represent the considered unsafe landing locations, and the color indicates the associated risk, where darkest is the riskiest.

aircraft is able to reach one of those landing locations, the viable planning space might be in some areas much too restricted.

The consideration of unsafe landing locations allows the configurations below the safe altitude to be connected into the tree. Consequently, the emergency landing trajectory can be computed even for those points. This is visualized in Figure 6.4, where a number of points directly above each other are considered. As the altitude increases, better landing locations become available, up to the point where the highest configuration is connected to a safe landing location. Each of the unsafe landing locations has a risk associated with it, which can then be used to estimate the risk of an emergency landing on that location. The risk is not precise because the impact energy and the impact angle are not known at the time of selection of the landing sites.

In the experiments, the safe landing locations are considered to be runways, which are only approachable from certain directions, while the unsafe ones are considered from all k directions within the discretization. In the evaluated scenarios, there were three runways considered safe landing locations, each approachable from two opposite directions, as visualized by the yellow dots in Figure 6.5. The minimal distance between two selected unsafe landing locations was fixed at 500 meters for each of the scenarios (the distance to the closest safe landing location was disregarded because of the restriction on the direction it is approached from). Examples of the selection based on the number of selected landing locations are shown in Figure 6.5.

If only the three safe landing locations are considered, about $\frac{1}{2}$ of the configurations are above the safe altitude level, as shown in Table 6.1. However, if unsafe landing locations are considered, the part of the configuration space that is connected to the emergency landing tree covers most of the samples above the terrain level. Each configuration connected to the tree has an available emergency landing trajectory, and the risk estimate for this trajectory is known. Increasing the number of selected samples pushes the threshold further down towards the terrain level, as shown in Table 6.1, up to about 80% of



(a) 10 selected landing locations (b) 50 selected landing locations (c) 75 selected landing locations

Figure 6.5: Selected landing locations. The red dots represent the unsafe landing locations, and are considered in all of the directions dictated by the discretization. The yellow dots represent the safe landing locations, and have an associated direction. The landing locations are selected so that the total covered area is maximized. In this setup, the minimal distance between them was fixed for each of the selections, which makes it so that the selection of landing locations is a superset of any selection with less landing locations. The underlying map shows the risk estimate.

Table 6.1: Percentage of connected samples with respect to the number of considered landing locations, computed on trees with $\Delta d = 50\,\mathrm{m}$, $\Delta z = 5\,\mathrm{m}$, $k = 8$, and 3 bi-directional runways. The pattern pool area was selected at $300\,\mathrm{m} \times 300\,\mathrm{m} \times 30\,\mathrm{m}$. The column Dominated denotes how many of the selected landing sites are dominated, and as such do not contribute anything to the solution. Note, that the percentage of safe locations greatly depends on the selected maximum altitude.

Landing Sites [-]	**Terrain Samples** [%]	**Connected** [$\times 10^6$]	**Unconnected** [$\times 10^6$]	**Percentage** [%]	**Dominated** [-]
0		3.8	4.1	47	0
10		5.4	2.5	68	1
25	7.77	6.0	1.9	76	5
50		6.3	1.6	78	8
75		6.2	1.6	79	13

Table 6.2: The properties of the resulting trees, based on the selected parameters.

Parameters			Time Measurement			Tree properties		Memory Requirements	
k	Δd [m]	Δz [m]	Total [h:min]	Connect [h:min]	Expand [h:min]	Samples [$\times 10^6$]	Pool [-]	Nodes [MB]	Total [MB]
8	100	10	0:05	0:02	0:03	1	112	8	27
		5	0:14	0:07	0:06	2	160	16	25
	50	10	0:45	0:13	0:32	4	444	32	107
		5	1:39	0:25	1:14	8	628	64	169
16	100	10	0:11	0:07	0:04	2	440	16	54
		5	0:28	0:15	0:13	4	656	32	88
	50	10	1:42	0:28	1:14	8	1828	64	655
		5	4:28	0:50	3:37	16	2724	128	1008

the configuration space covered.

Figure 6.6 shows a cut of the tree through the xz plane. The colors show, what is the value of the configuration: safe, unsafe, unconnected, or terrain. Figure 6.6b and Figure 6.6c display cuts of two different trees, showing how the number of selected landing sites affects how much of the configuration space is connected to a landing location and thus has known risk associated with it. The safe altitude is the same for all of the computed trees because the considered safe emergency landings remain the same and because the only parameter that changes is the number of unsafe landing locations. Any path to a safe emergency landing exists regardless of any considered unsafe landing locations.

Numerically, this is shown in Table 6.1. Since the landing locations are selected in advance, and their coverage has to be estimated, it may happen that some have very little contribution or no contribution at all. The column Dominated shows how many of the selected landing locations were completely dominated, that is, no configuration was connected to them. This may happen because the terrain and obstacles around them make it hard to reach or because there is a landing location with lower risk close enough (most likely a combination of those factors).

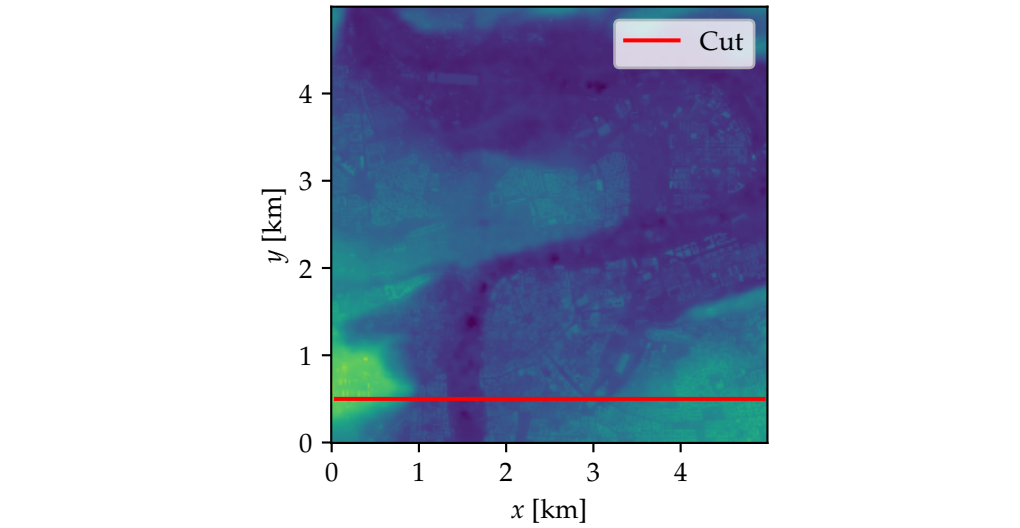
6.1.3 Computational Requirements

The proposed method was evaluated with a number of different parameter configurations. This section discusses the influence of the parameter selection on the time and memory requirements of the algorithm. The results of some of the experiments with respect to the complexity are summarized in Table 6.2.

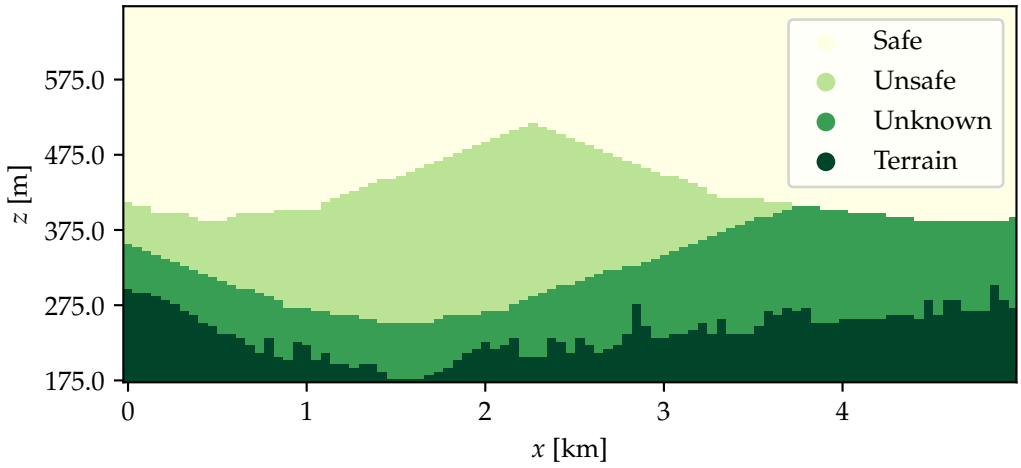
6.1.3.1 Time Complexity

The algorithm is evaluated based on its time requirements and analyzed as to which parts are the most computationally intensive and why. The influence of the individual parameters with respect to the time requirements is discussed in depth based on the measurement data.

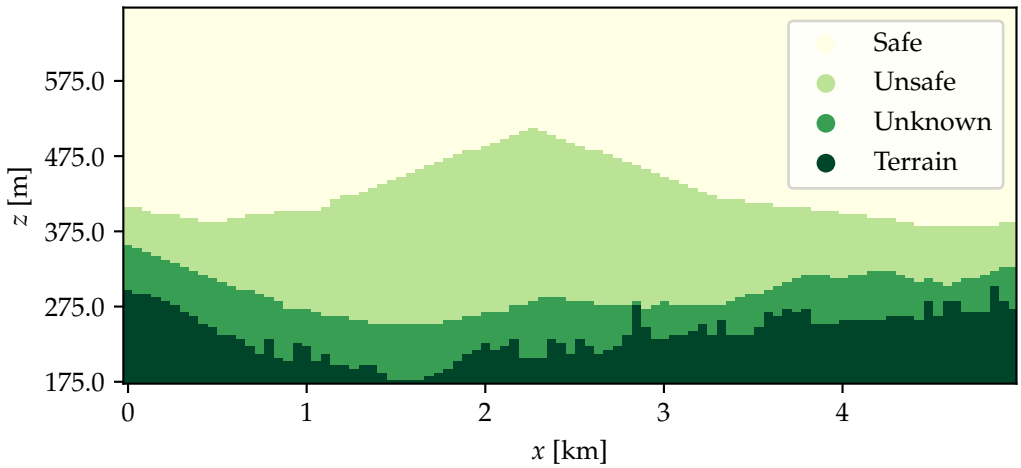
Table 6.2 offers an overview of the time and memory requirements based on the selected parameters, as well as the size of the pattern pool and the number of samples. It shows that there are two parts of the computation that make up most of the run time. First of those is the connection of the considered landing locations into the tree. This includes both the safe landing locations and the unsafe ones. The



(a) Cut location



(b) 5 unsafe landing locations



(c) 75 unsafe landing locations

Figure 6.6: The visualization of the vertical cut of the tree. The y axis position was fixed, and the image shows the xz plane, colored according to the connected emergency landing locations. The safe altitude is colored with the lightest color. Darkest color represents terrain. Unsafe refers to locations, that are connected to an unsafe one, whereas Unknown refers to locations, that are not connected to any emergency landing at all.

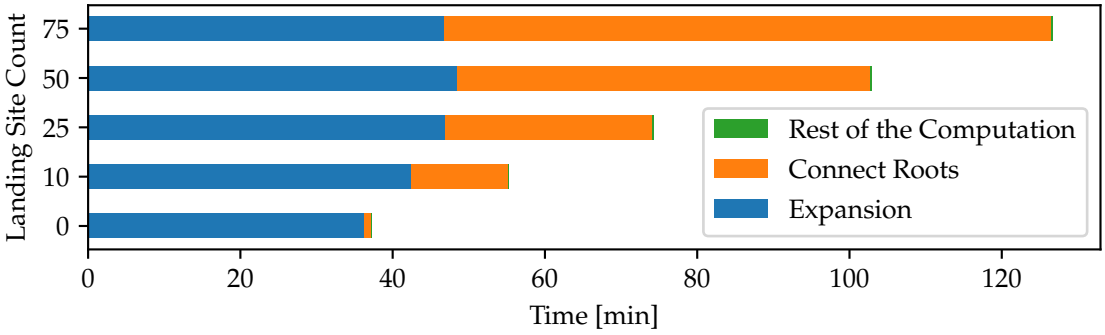


Figure 6.7: Time requirements with respect to the number of selected landing locations, as measured on a tree with $\Delta d = 50\,\mathrm{m}$, $\Delta z = 5\,\mathrm{m}$ and $k = 8$.

second computationally intensive part of the computation is the expansion itself. The other parts of the computation include pattern pool generation, landing site selection, and risk computation, however, in the context of the overall time complexity, all of these are irrelevant.

The time requirements of the connection of landing locations depend on several factors. Most importantly, it depends on the number of selected landing locations. The connection time is mostly linear with respect to the number of selected landing locations, as is shown in Figure 6.7. However, if the number of landing locations is high, it may happen that a some samples could be covered by multiple landing locations. In that case, the connections to the worse of the two landing locations will have been outclassed, even before they are computed. Such a connection would not improve the tree and is skipped entirely, which may slightly improve the time requirements of the connection step.

Another very important factor influencing the complexity of the connections is the Near function (Algorithm 2, Line 8). The Near function searches the same area the pattern pool covers because this is the area that all of the other expansion steps cover. For every near sample, a connection must be computed and checked for collisions. Thus, smaller area and hoarser sampling greatly reduces the time necessary to connect the landing locations.

The second computationally intensive step is the expansion of the tree itself. The most important factors here are the number of expanded nodes and the number of maneuvers that are considered during the expansion. The size of the pool, as stated in Table 6.2 represents the number of nodes needed to expand a location, not a node. Thus, the expansion is mostly dependent of the number of maneuvers in the pool and the number of locations. Importantly, the expansion is almost independent on the number of selected landing locations, as shown in Figure 6.7. The difference shown is partially due to the fact that a higher number of landing locations leads to more established connections in the lower parts of the tree, but the role of variance is hard to estimate.

Surprisingly, the collision check caching has very little effect because the vast majority of the evaluated maneuvers are pruned before the collision check itself even occurs. The collision checking amounts to only about $4\,\%$ of the computational time (with the caching). Even though the caching improves the complexity of the collision check quite significantly, as shown in Table 6.3, the collision check itself is not significant in the scope of the whole computation.

6.1.3.2 Memory Requirements

Since the whole emergency landing tree has to be stored in memory, the proposed method has also quite significant memory requirements. However, several optimizations have been used that significantly reduce the necessary memory. This section shortly discusses this aspect of the proposed method.

Table 6.3: The number and timings of the collision checks, based on a tree constructed with $\Delta d = 50\,\mathrm{m}$, $\Delta z = 5\,\mathrm{m}$, and $k = 8$, and pattern pool area $300\,\mathrm{m} \times 300\,\mathrm{m} \times 30\,\mathrm{m}$

Type	Evaluations [$\times 10^6$]	Single Query Time [ns]	Total Time [s]
Full Check	5.1	20 000	103
Cached Check	5.9	360	2

There are two parts of the emergency landing tree that make up the majority of the memory requirements of the whole structure. The first one is the tree structure itself, that is, the nodes and the connections. The second is the collision check lookup table.

In order to store the emergency landing tree in memory, the nodes in the tree have to somehow be stored. Each of those nodes holds two important pieces of information: its position and the connection to the parent. However, the discretization of the configuration space enables significant simplification. Firstly, since the discretization is regular, the nodes can be stored in a four-dimensional array, which captures the location and orientation of the node in the configuration space. Secondly, since only a relatively small number of maneuvers is used, instead of storing the connection to the parent node, it is possible to store only the index of the maneuver in the precomputed maneuver pool. Thus, the node is well-defined by a single integer and its location in the array. However, since multiple risks are considered, also the rank of its risk is stored. This is not strictly necessary because the risk can be found by traversing the tree up to the root, however, this means that the risk estimate can be directly read from the node itself. In total, for every node, two integers have to be stored. For example, the finest sampling experiment had the sampling rates $\Delta d = 50\,\mathrm{m}$, $\Delta z = 5\,\mathrm{m}$, $k = 16$ on a $5000 \times 5000\,\mathrm{m}$ map, reaching up to $500\,\mathrm{m}$ into the air. This amounts to $100 \times 100 \times 100 \times 16 = 16 \times 10^6$ nodes, and the memory necessary is then $128\,\mathrm{MB}$, assuming 32-bit integers.

A similar estimate can be made for the collision check lookup table, but it depends on the number of maneuvers in the pattern pool Λ, which has no straightforward estimate. Assuming the same discretization parameters as above, the number of stored values is then $100 \times 100 \times 16 \times |\Lambda|$. For the number of patterns $|\Lambda| = 1200$ and 16-bit integers, this is $384\,\mathrm{MB}$. However, the number of precomputed patterns might be as low as 100, and as high as 10 000 or more, which makes the size much harder to estimate. Note that the collision check lookup table could be stored more efficiently at the cost of access complexity. Importantly, the collision check lookup table is only useful for the expansion of the tree. Once the tree is fully expanded, it is unnecessary and may be discarded. The tree can then be saved with very little memory overhead.

6.1.4 Risk Estimate

The emergency landing tree is used to compute emergency landing trajectory in case of a partial failure and to determine the risk induced by such a failure. The connection is established by attempting to connect the point to all the near nodes until a feasible connection is found. However, for the purposes of the risk-aware trajectory planner, the full query is too complex, and therefore the simplified version is used.

Instead of computing the optimal emergency landing trajectory, the simplified query considers all the surrounding nodes. The estimated value is then the worst case across those points. This estimate is computationally very simple.

In Table 6.4, the simplified query is evaluated based on the comparison to the true risk value obtained by the full query. The comparison is performed on a tree with a high number of considered unsafe

Table 6.4: Comparison of the two queries, based on randomly sampled configurations. Computed on a tree with 75 unsafe landing locations considered and $\Delta d = 50\,\mathrm{m}$. Column Connected shows, how much of the random queries had been successfully connected to the tree by the full query, and the columns Safe shows, how much of the configurations were above the safe altitude level. Columns Underestimated and Overestimated show, how often the simplified query underestimates or overestimates the risk value. Each measurement is done on sample size 1000.

Above Terrain [m]	Underestimated [%]	Correct [%]	Overestimated [%]	Connected [%]	Safe [%]
50 m	0.6	74.2	25.2	57.3	0.3
75 m	0.2	64.3	35.5	84.1	1.8
100 m	0	64.6	35.4	96.5	6.6
150 m	0	81.3	18.7	99.0	16.7
200 m	0	85.5	14.5	98.3	33.9

landing locations on different altitude levels. According to the measurements, the simplified query returns, in the majority of the cases, the same result, and there is only a very small number of samples, for which the simplified query underestimated the risk. Furthermore, the accuracy of the simplified query increases as the altitude increases. This is caused by the fact that higher above ground, the number of relevant landing location decreases because the ones with greater risk become outclassed. This also causes the slightly higher accuracy very low to the ground, as much more of the configurations there were impossible to connect altogether.

6.2 Risk-Aware Planning

The risk-aware planner is implemented using the RRT* algorithm with the risk of the trajectory as the cost function. The risk is estimated as the sum of risks induced by the fatal failure and the considered partial failures (5.3). The RRT* algorithm is executed in a four-dimensional configuration space $\mathcal{C} = \mathbb{R}^3 \times \mathbb{S}$, where each configuration consists of the three-dimensional position, $[x, y, z] \in \mathbb{R}^3$, and the heading angle ψ The number of near nodes considered during the parent selection and the rewiring step was set to 10, and the risk estimate was determined based on $d = 10\,\mathrm{m}$ sampling of the path.

The planner was tested on three different pairs of starting and goal configurations. One path was shorter, the configurations about $1\,\mathrm{km}$ apart, and the other two were about $3\,\mathrm{km}$ apart. The configuration pairs are visualized in Figure 6.8. Each experiment was run three times. In addition to the loss of thrust failure, two other failures are considered, in which the loss of power leads to a control surface failure, which in turn inhibits the maneuverability by increasing the left or right turning radius five-fold. The multi-failure scenario is meant to demonstrate the capabilities of the emergency landing planner and is not based on actual technical issues of the Cessna aircraft.

6.2.1 Resulting Trajectories

The inclusion of the emergency landing planner allows the risk-aware planner to propose trajectories that are below the safe altitude level while not disregarding the potential risk this induces. This also means that the planner can find and optimize trajectories to and from locations that are below the safe altitude level.

The utilized RRT* algorithm does not have a hard terminating condition. It can run indefinitely and

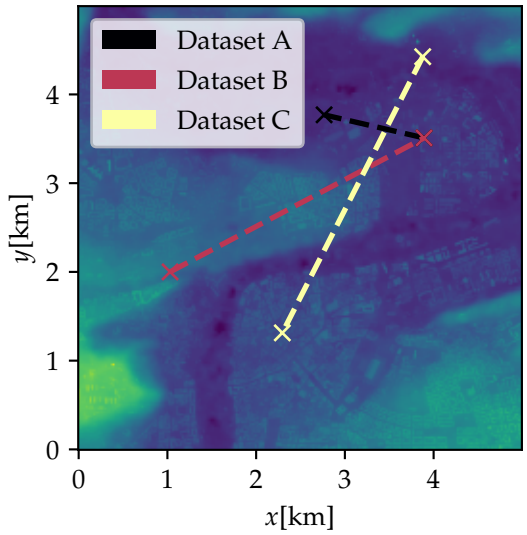


Figure 6.8: Locations of the initial and goal configurations. Each color represents one pair.

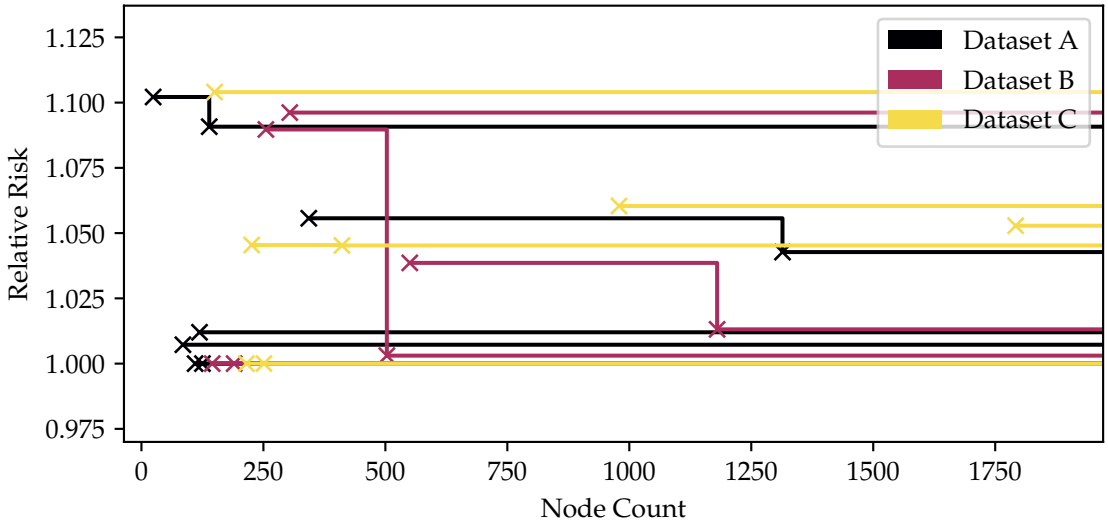


Figure 6.9: The risks of the trajectories found by the risk planner, relative to the best found trajectory for that dataset. Each color represents one dataset, and each line represent a separate run of the risk planner. The results, where only the first found result was considered, are not shown.

keep improving the known solution. This means that the quality of the solution is dependent on how much time the planner is granted. Therefore, for each of the configuration pairs, three experiments were run (each repeated three times). One, in which the first found solution was returned as the result, and two, where the planner ran for a predetermined time, 30 min and 1 h. The summary of results of those experiments is presented in Figure 6.9.

Interestingly, for most of the runs, once a solution was found, it was never improved, and the first found results were generally very close to the best ones (the worst of the results were about 1.1 of the best found result, as seen in Figure 6.9). This is likely caused by the fact that the number of established nodes is relatively low with respect to the dimensionality and size of the configuration space. The risk for the individual runs of the planner was relatively consistent in that no single run resulted in a significantly riskier path than any of the other ones, and only one experiment did not finish in the provided time frame. On the other hand, the time necessary to find that solution varied, in that in some runs, the first solution was found within a few hundred iterations, and in some cases,

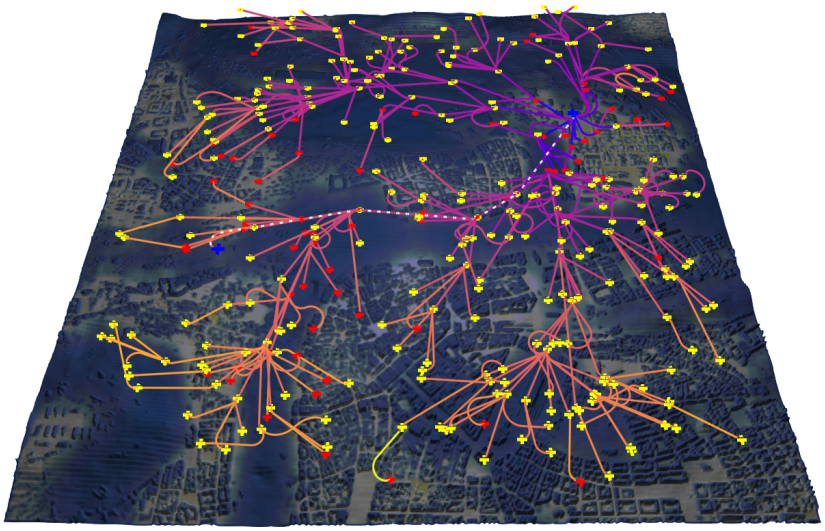


Figure 6.10: An example of a trajectory computed by the risk-aware planner. The start and target configurations (Dataset B) are depicted in blue. Other points are drawn in yellow if they are above the safe altitude or red if they are below the safe altitude. The lines are colored according to the risk to goal, where darkest are least risky. The dashed white path represents the found solution.

Table 6.5: Comparison of the risks of paths found by the proposed risk-aware trajectory planner and the risk of the shortest path, as found by an RRT* algorithm run with length of the path as the cost function. The risk value of the proposed method is taken as an average of the first found results based on nine independent runs of the algorithm.

Data	First Result Average Risk $\times 10^{-5}$	Shortest Path Risk $\times 10^{-5}$
Dataset A	2.7	2.9
Dataset B*	6.3	9.3
Dataset C	6.2	12.2

*one of the nine runs did not finish, therefore the average is taken only over the 8 runs that did finish

it took over a thousand. The time necessary to find a solution depends on the positions of the points, if the goal configuration is far or in a high-risk area (such as low to the ground), the planner will take much longer.

Since, in most cases, the first result was never improved, most of the results of the differently timed experiments are comparable. The resulting risks were also compared to the risk of the shortest path, as found by the RRT* algorithm, with the length of the path as the cost function. The comparison is shown in Table 6.5, where the risk of the proposed method is represented as an average of nine independent runs. In comparison to the shortest path, the proposed method decreases the risk significantly. With the exception of the short scenario, all of the experiments presented paths with much lower estimated risk, and even in the short scenario, the average result of the proposed method achieved a small improvement, even though since the traveled distance is small, any detours increase the total length much more significantly and the shortest path is naturally relatively safe.

An example of a resulting RRT* tree is shown in Figure 6.10. Thanks to the parent selection and steering function, the tree is grown in a way that leads to much denser growth in areas with lower overall risk (darker areas on the underlying map). The example of the grown tree also shows that the consideration of the unsafe configurations significantly broadens the planning space, as a significant portion of the nodes is below the safe altitude level. This is especially true for leaf nodes because if

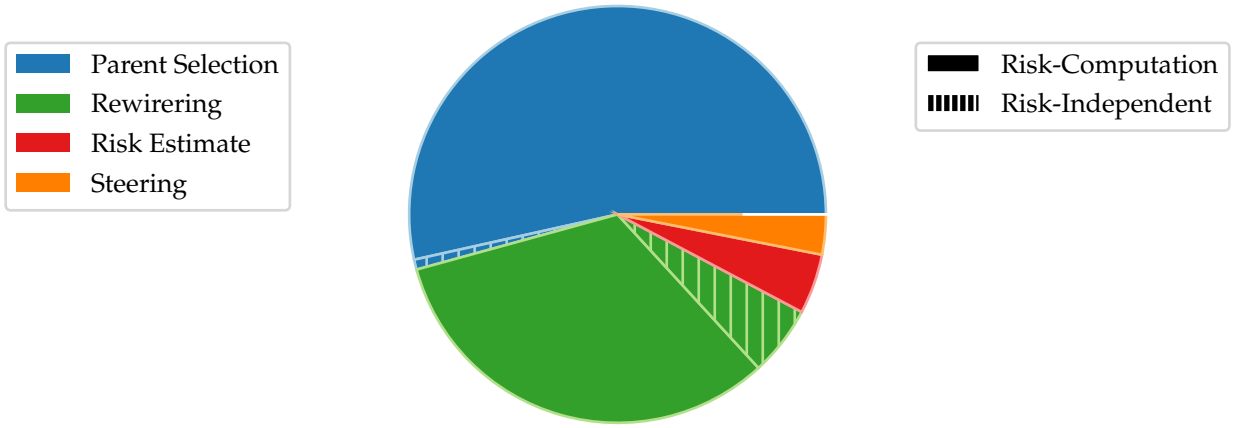


Figure 6.11: Time requirements of the individual steps of inserting a node. Each color corresponds to the individual steps of the process. The bottleneck of each of those steps is the risk-estimate, shown in solid color. The hashed portions represent the rest computation. The risk-estimate is the most demanding part of the computation, and needs to be executed many times during the process of inserting a single node.

the node is below the safe altitude level, its risk is bound to be higher, and thus it is less likely to be selected as a parent node.

The planner could, in theory, be used without considering the fatal failure. However, if only partial failures are considered, any maneuver above the safe altitude has no associated risk. Thus, the planner lacks anything that would guide the exploration, and the result is a chaotic tangle of maneuvers that twist and turn in all directions. The bulk of the tree would be wholly above the safe altitude with zero risk, and lower would be almost exclusively the leaves. Any trajectory that does not descend below the safe altitude is safe, and thus the rewiring procedure cannot improve anything. Consequently, the resulting trajectory would be nonsensical. The optimality would not be lost, but the optimality condition makes little sense.

6.2.2 Computational Requirements

The RRT* algorithm repeatedly creates randomly generated nodes and inserts them into the tree until the goal configuration is inserted. The relative time complexity of the individual steps of this process is displayed in Figure 6.11. As is seen, There are two major steps in the process of inserting a new node: parent selection, which takes up about 50% of the computational time, and the rewiring step, which takes up approximately 33%. The rest of the computational time is spent on inserted node evaluation and steering. However, with respect to the presented risk-aware planner, the bottleneck for all of those steps is the risk computation, which takes up the vast majority of the overall computation time.

The risk function that is used as the cost for the RRT* planner is formulated as the sum of the risk induced by individual partial failures and the risk induced by fatal failure, as defined in (5.3). Both of those parts are computed separately for each of the samples, therefore it is possible to evaluate the computational burden of multiple considered partial failures.

Table 6.6 presents the time complexity of the risk estimate, as measured during multiple experiments of the risk-aware planner with three partial failures considered. The table is separated into two sections. The first shows the properties of the risk computation at a given configuration with respect to the two different types of risks, partial and fatal. The second part shows the time complexity and number of computations of the risk estimate for a maneuver.

Table 6.6: Risk function call parameters based on multiple risk-planner runs.

Risk Type	Calls [$\times 10^3$]	Total Time [h : min]	Average Time [µs]
Risk at Configuration			
Partial	54 000	0:54	60
Fatal	26 000	2:49	400
Maneuver Risk			
Parent Selection	170	2:20	47 600
Rewiring	180	1:24	27 200

The risk of a maneuver is computed as an integral of the risk function over the maneuver, which is estimated as the sum of risks of configurations sampled along the path. For each of the sampled configurations, the risk estimate for each considered failure is computed. According to the measurements, the two different types of failures differ greatly in the time necessary for the risk estimate. The partial failure risk estimate is much faster than the fatal failure risk estimate and increasing the number of considered partial failures (assuming the trees are precomputed) increases the computation time of the risk-aware planner at most by the length of the associated queries during the risk computation. Consequently, it is computationally feasible to consider many different partial failures represented by the proposed emergency landing tree.

In order to add a new node, a parent has to be selected, and if the node is successfully added, the rewiring step is executed. The whole process of adding a new node with the parameters used takes in total about 1 s, but is subject to a surprising amount of variance. During the parent selection, the risk computation has to be executed on every considered connection to a potential parent. This amounts, on average, to slightly less than 10 risk estimate computations (for nearest neighbor search of 10) Similarly, during the rewiring step, a number of maneuvers have to be evaluated, up to twice as much. On average, this led to about 11 more risk estimate computations during the rewiring step. Interestingly, the risk estimate during the rewiring step was, on average, shorter. This is because once a connection exists, any risk computation may be interrupted once it exceeds the risk of that connection. If the connection is improved, the following risk computation will be interrupted sooner. Because the rewiring step comes after parent selection, the bound on the risk is already lower than that during the parent selection, and thus the computation is interrupted sooner and the overall time spent in this step is lower.

Chapter 7
Conclusion

The problem of finding a risk-minimizing trajectory for a fixed-wing vehicle was studied in this work. The promising technical development of small aircraft and the expected advent of Urban Air Mobility bring aerial vehicles much closer to the cities, which also means that an in-flight failure poses a risk not only to the crew and passengers but also to the people on the ground. Therefore, it is desirable that the aircraft path planner minimizes the risk induced by such a failure and that if such a failure does happen, the aircraft is able to recover and plan an emergency landing trajectory if possible.

Firstly, the risk induced by different partial failures has been addressed, where partial failure is any failure that leads to reduced maneuverability but not to the total loss of control. A multi-failure model based on the Dubins Airplane has been proposed, which can describe loss of thrust failures, failures inhibiting maneuverability, and any combination of those. However, only failures preventing the aircraft from gaining or maintaining altitude are considered because those are the most common and critical.

The approach proposed in this work utilizes discretization of the configuration space, which significantly reduces the computational complexity of the emergency landing planning. Thanks to the discretization, only a handful of maneuvers are used throughout the computation. Therefore, they may be precomputed and reused, which greatly reduces the computational burden. Thus, it is computationally feasible to consider multiple emergency landing locations with different risks each, such that the least risky reachable landing location is selected. The result of this algorithm is a risk-minimizing emergency landing tree, which can be queried to estimate the risk associated with a partial failure at any configuration and to determine the best available emergency landing trajectory.

Additionally, a risk-aware path planner has been proposed, which plans paths such that the risk posed by an in-flight failure is minimized. The risk function consists of the risk induced by a fatal failure, which leads to an uncontrolled crash, and the risk induced by partial failures, estimated using the proposed emergency landing tree. The utilization of the proposed emergency landing tree allows the planner to evaluate the risk induced by partial failures even for locations from which no safe emergency landing location is reachable. An emergency landing tree may be constructed for any partial failure, and the computation was tested with two different partial failures, in loss of thrust and loss of thrust combined with inhibited turning ability. However, the partial failure selection needs to be based on the properties of the specific aircraft. The proposed risk-aware planner has been evaluated on an urban scenario based on the Prague city center and has been shown to propose trajectories with lower risk compared to the shortest paths.

In our future work, we would like to iterate on the emergency landing tree computation in order to further reduce the computational requirements, which would allow for finer discretization and, thus, for a higher quality of the resulting emergency landing paths. By nature of the emergency landing tree, on average, one maneuver per location is actually useful, because the number of nodes in each layer is the same. Therefore, careful evaluation of the precomputed maneuvers seems like a promising area to improve the algorithm, as it could reduce the number of maneuvers considered, which is an important factor of the emergency landing tree computation.

References

[1] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30:846 – 894, 2011.

[2] Ian Savage. Comparing the fatality risks in united states transportation across modes and over time. *Research in Transportation Economics*, 43(1):9–22, 2013. The Economics of Transportation Safety.

[3] Shahab Hasan. Urban air mobility (uam) market study. Technical report, National Aeronautics and Space Administration (NASA), 2019.

[4] Study on the social acceptance of urban air mobility in europe. https://www.easa.europa.eu, 2021. Accessed on: 21 May 2023.

[5] National Transportation Safety Board. General aviation accident dashboard 2012-2021. https://www.ntsb.gov/safety/Pages/default.aspx, 2023. Accessed on: 21 May 2023.

[6] J. David Kenny. *26th Joseph T. Nall Report*. Richard G. McSpadden, JR., 2017.

[7] Engine failures and malfunctions in light aeroplanes. https://www.atsb.gov.au/publications/2013/ar-2013-107_research, 2016. Accessed on: 21 May 2023.

[8] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[9] *Pilot's Handbook of Aeronautical Knowledge*. U.S. Department of Transportation, Federal Aviation Administration, Flight Standards Service, 2022.

[10] Lester E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[11] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli. A novel formalisation of the markov-dubins problem. In *European Control Conference (ECC)*, pages 1987–1992, 2020.

[12] S. Hota and D. Ghose. Optimal geometrical path in 3d with curvature constraint. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 113–118, 2010.

[13] H. Chitsaz and S. M. LaValle. Time-optimal paths for a dubins airplane. In *46th IEEE Conference on Decision and Control*, pages 2379–2384, 2007.

[14] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti. Path generation and tracking in 3-d for uavs. *IEEE Transactions on Control Systems Technology*, 17(4):980–988, 2009.

[15] Y. Wang, S. Wang, M. Tan, C. Zhou, and Q. Wei. Real-time dynamic dubins-helix method for 3-d trajectory smoothing. *IEEE Transactions on Control Systems Technology*, 23(2):730–736, 2015.

[16] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing Dubins Airplane Paths on Fixed-Wing UAVs**, pages 1677–1701. Springer Netherlands, Dordrecht, 2015.

[17] P. Váňa, A. Alves Neto, J. Faigl, and D. G. Macharet. Minimal 3d dubins path with bounded curvature and pitch angle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8497–8503, 2020.

[18] A. A. Neto, D. G. Macharet, and M. F. M. Campos. 3d path planning with continuous bounded curvature and pitch angle profiles using 7th order curves. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4923–4928, 2015.

[19] Simon Schopferer, Julian Sören Lorenz, Azarakhsh Keipour, and Sebastian Scherer. Path planning for unmanned fixed-wing aircraft in uncertain wind conditions using trochoids. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512, 2018.

[20] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[21] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[22] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[23] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[24] L.E. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[25] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528 vol.1, 2000.

[26] C.L. Nielsen and L.E. Kavraki. A two level fuzzy prm for manipulation planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Cat. No.00CH37113)*, volume 3, pages 1716–1721 vol.3, 2000.

[27] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[28] Yanbo Li, Zakary Littlefield, and Kostas Bekris. *Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning*, pages 263–282. Springer International Publishing, 2015.

[29] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004, 2014.

[30] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA). Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001, 2000.

[31] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[32] Stefano Primatesta, Giorgio Guglieri, and Alessandro Rizzo. A risk-aware path planning strategy for uavs in urban environments. *Journal of Intelligent & Robotic Systems*, 95, 2019.

[33] Stefano Primatesta, Luca Spanò Cuomo, Giorgio Guglieri, and Alessandro Rizzo. An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments. *Transportation Research Procedia*, 35:44–53, 2018.

[34] Stefano Primatesta, Matteo Scanavino, Giorgio Guglieri, and Alessandro Rizzo. A risk-based path planning strategy to compute optimum risk path for unmanned aircraft systems over populated areas. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 641–650, 2020.

[35] Jakub Sláma, Petr Váňa, and Jan Faigl. Risk-aware trajectory planning in urban environments with safe emergency landing guarantee. In *IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1606–1612, 2021.

[36] Ella Atkins, Igor Portillo, and Matthew Strube. Emergency flight planning applied to total loss of thrust. *Journal of Aircraft*, 43:1205–1216, 2006.

[37] Pillar Eng, Luis Mejias, Rodney Walker, and Daniel Fitzgerald. Simulation of a fixed-wing uav forced landing with dynamic path planning. 2012.

[38] Nicolas Meuleau, Christian Plaunt, and David Smith. Emergency Landing Planning for Damaged Aircraft. *International Conference on Automated Planning and Scheduling*, 2008.

[39] Michael Warren, Luis Mejias, Jonathan Kok, Xilin Yang, Felipe Gonzalez, and Ben Upcroft. An automated emergency landing system for fixed-wing aircraft: Planning and control. *Journal of Field Robotics*, 32(8):1114–1140, 2015.

[40] Petr Váňa, Jakub Sláma, Jan Faigl, and Pavel Pačes. Any-time trajectory planning for safe emergency landing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5691–5696, 2018.

[41] Petr Váňa, Jakub Sláma, and Jan Faigl. Surveillance planning with safe emergency landing guarantee for fixed-wing aircraft. *Robotics and Autonomous Systems*, 133:103644, 2020.

[42] Young-Won Kim, Dong-Yeon Lee, Min-Jea Tahk, and Chang-Hun Lee. A new path planning algorithm for forced landing of uavs in emergency using velocity prediction method. In *2020 28th Mediterranean Conference on Control and Automation (MED)*, pages 62–66, 2020.

[43] Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming*, 1972.

[44] Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[45] Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 435–441, New York, NY, USA, 1996. Association for Computing Machinery.

[46] Chandra Chekuri. Cs 583 : Approximation algorithms : Covering problems. 2018.

[47] Richard L. Church and Charles S. Revelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32:101–118, 1974.

[48] Harummi Sekar Amarilies, A A N Perwira Redi, Ilma Mufidah, and Reny Nadlifatin. Greedy heuristics for the maximum covering location problem: A case study of optimal trashcan location in kampung cipare – tenjo – west java. *IOP Conference Series: Materials Science and Engineering*, 847(1):012007, 2020.

[49] Hassan Haghighi, Daniel Delahaye, and Davood Asadi. Performance-based emergency landing trajectory planning applying meta-heuristic and dubins paths. *Applied Soft Computing*, 117:108453, 2022.

[50] Stefano Primatesta, Alessandro Rizzo, and Anders la Cour-Harbo. Ground risk map for unmanned aircraft in urban environments. *Journal of Intelligent & Robotic Systems*, 97, 2020.

[51] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2023. Accessed on: 25 May 2023.

[52] JPL NASA. Nasa shuttle radar topography mission global 1 arc second. https://opentopography.org/, 2013. Accessed on: 25 May 2023.

[53] Facebook Connectivity Lab and Center for International Earth Science Information Network - CIESIN - Columbia University. High Resolution Settlement Layer (HRSL). Source imagery for HRSL © 2016 DigitalGlobe. https://www.ciesin.columbia.edu/, 2016. Accessed on: 25 May 2023.

[54] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.