

Article

Mission-Driven UAV Path Selection: Post Hoc Cost Evaluation of Deterministic and Sampling Approaches

James R. Kelly ¹ and Umair B. Chaudhry ^{1,2,*}¹ School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK² Department of Computer Science and Information Sciences, Northumbria University, London E17 HT, UK

* Correspondence: u.b.chaudhry@qmul.ac.uk

Highlights

What are the main findings?

- A cost-aware RRT* (CARRT*) yields the lowest mission cost and best path efficiency in low-fuel and time-sensitive scenarios, while grid-based planners (Dijkstra/A*, weighted A*) perform best for high-safety and reward-focused missions.
- Post hoc global cost analysis shows that mission cost is most sensitive to distance and energy weights; safety/reward weights shape outcomes when strong penalty/reward fields are present.

What are the implications of the main findings?

- Planner choice should be aligned with mission priorities; decoupling path generation from prioritisation enables rapid retuning as objectives change without modifying certified planners.
- The framework offers a reproducible benchmark for researchers and operators to compare planners across environments and select algorithms/weights suited to each mission.

Abstract

Unmanned Aerial Vehicles (UAVs) are increasingly deployed in hazardous and dynamic environments, where path planning requires balancing competing objectives beyond simple distance minimisation. Classical planners such as Dijkstra, A*, and RRT* generate paths efficiently but often overlook mission-specific trade-offs involving energy use, risk avoidance, and reward maximisation. This work proposes a unified evaluation framework that integrates grid-based (Dijkstra, A*, weighted A*) and sampling-based (RRT, CARRT*) planners within parameterised environments embedding a range of functions into penalty and reward zones. A global cost function, $J = \alpha L + \beta E + \gamma P - \delta R$, is applied post hoc to decouple path generation from mission prioritisation, enabling rapid reassessment under changing objectives such as low-fuel, high-safety, or speed-priority scenarios. Experiments conducted on an Apple M2 CPU, repeated three times per configuration to ensure statistical robustness, demonstrate that CARRT* achieves the lowest mission costs and highest efficiency for fuel- and time-sensitive missions, while deterministic grid-based planners perform better in safety- and reward-oriented contexts in four environments. These findings indicate that optimal UAV path planning depends not only on algorithmic efficiency but also on aligning planner choice with mission priorities. The framework provides a reproducible methodology for benchmarking and deploying mission-aware path planning strategies in research and operational settings.

Academic Editors: Paraskevi Zacharia, Antreas Kantaros, Elias K. Xidias and Antoni Grau

Received: 15 September 2025

Revised: 27 January 2026

Accepted: 2 February 2026

Published: 22 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: UAV path planning; cost-aware RRT*; A*; Dijkstra algorithm; sampling-based planning; grid-based planning; post hoc cost evaluation

1. Introduction

In the aftermath of the 2023 wildfires in Maui, Hawaii, unmanned drones were able to map fire perimeters in under ten minutes, an operation that would have taken hours using manned aircrafts [1]. Unmanned Aerial Vehicles (UAVs) are now routinely deployed for tasks ranging from infrastructure inspection to emergency response. Their broad utilisation has been seen in surveillance, logistics, environmental protection and precision farming, acting as a catalyst for innovation [2,3]. UAVs have the advantage of reducing operator risk by having the ability to be controlled remotely or autonomously, whilst still being capable of completing tasks with great precision and reliability [4,5]. This is especially useful when operating in areas of high risk, such as disaster relief or environmental sensing. UAVs are also utilised in the real-time monitoring of road traffic, providing wireless coverage, delivery of goods and civil infrastructure inspection [6,7].

Being unmanned, UAVs often make use of pathfinding algorithms to get from a start point to a goal efficiently and safely. The broad range of industries and therefore environments that UAVs perform in present multi-faceted problems when looking at the use of these pathfinding algorithms. All environments present a different problem that algorithms often need to adapt around, and most missions have different priorities that must be catered for. These constantly changing problems, along with a constant desire to improve efficiency, has birthed the long list of algorithms used today. It is often observed that when designing pathfinding systems, studies focus narrowly on minimising distance or time [8–10]. In practice, operators must also consider a variety of mission critical factors, such as penalty flight zones, preferred flight corridors, optimising energy consumption and creating collision-free paths. Our traditional algorithms, A* and RRT*, do not capture these issues so must be built upon [11–13]. In order to optimise algorithm selection, we compare algorithms' skillsets against mission objectives via a post hoc mission cost, decoupling path generation from prioritisation.

Existing studies generally (i) evaluate planners using a fixed cost model making objectives planner bound, which can be inflexible and limits application in fast-paced scenarios; (ii) often focus on distance and time whilst not considering energy cost and penalty/reward areas, limiting adaptation to different mission priorities; (iii) couple planner internals, environments and weights, making it hard to change priorities mid-flight; and (iv) report limited sensitivity analysis of how mission weights drive total cost, in result not giving us an accurate picture of our most critical variables. As a result, there is a need for a reproducible method to compare planners under the same paths whilst varying mission priorities post hoc.

So, how can we compare both deterministic and sampling algorithms accurately and dynamically whilst considering a range of changing mission objectives? The study makes four main contributions to address these gaps and tackle this problem. First, the paper designs a range of parameterised environments, ranging from open to dense urban and mixed terrain to multi-objective that embed logarithmic, exponential, Gaussian, quadratic and step cost fields to emulate realistic reward/penalty landscapes. These environments are the testing ground for our experiment and their variety allows us to explore different algorithmic behaviours. Secondly, RRT* is adapted to consult movement costs during travel through reward and penalty areas, allowing it to be compatible with a cost function. Third, the paper introduces a cost function that unifies both grid-based planners (Dijkstra, A*, weighted A*) and sampling-based planners (RRT, cost-aware RRT* (CARRT*)) with

region-specific reward/penalty multipliers and a single objective $J = \alpha L + \beta E + \gamma P - \delta R$ (see definition defined in Methods), enabling operators to trade off distance, energy, risk and reward without rewriting core algorithms. Fourth, a post hoc global cost analysis is conducted by sweeping α , β , γ , and δ , including sensitivity studies and mission-oriented slices (low-fuel, high-safety, speed-priority, reward-focused, balanced), and compares planners using total weighted cost, path efficiency, zone incursions (a zone incursion was counted whenever a path entered a cell with movement cost exceeding the base cost, with a one-cell buffer to avoid counting borderline cases) and computational effort, yielding guidance on algorithm selection.

This paper is organised as follows: Section 2 reviews the literature review and background on the deterministic and sampling-based planners used throughout the study along with discussion about cost modelling and other algorithms used in industry. Section 3 details the parameterised environments, planners in use and the unified post hoc cost framework. Section 4 presents the analysis and results, including sensitivity studies and mission-oriented comparisons across algorithms and environments. Section 5 concludes with key findings and outlines avenues for future work.

This paper's main contributions are listed below:

- A suite of parameterised, function-based environments (logarithmic, exponential, quadratic, Gaussian, and step cost fields) designed to emulate realistic UAV penalty/reward landscapes and expose behavioural differences between planners.
- A cost-aware variant of RRT* (CARRT*) adapted to be used on the environments that enables sampling-based methods to react intelligently to penalty and reward fields.
- A unified post hoc mission cost framework ($J = \alpha L + \beta E + \gamma P - \delta R$) which decouples path generation from mission prioritisation, making it possible to re-evaluate fixed planner outputs under different mission objectives without modifying planner internals.
- A comprehensive mission-driven evaluation methodology including a full weight sweep, sensitivity analysis, and five mission scenarios (low-fuel, high-safety, speed-priority, reward-focused, balanced), providing actionable guidance for choosing planners under changing mission priorities.

2. Literature Review and Background

Graph-based vs. sampling algorithms

Graph-based planners discretise the configuration space into a finite set of vertices and edges, then apply deterministic graph search algorithms to guarantee optimal, complete paths [11,14–16]. They build and search an explicit graph before planning, which yields exact solutions but can be slow to replan when the environment changes and may not result in the smoothest trajectories.

Sampling-based planners, on the other hand, like RRT and RRT*, incrementally sample random configurations and connect them to form a roadmap or tree, offering probabilistic completeness without precomputing the entire graph [14,15,17]. Their lightweight, incremental nature makes them well suited to high-dimensional or dynamic settings, where real-time replanning and complex constraints are required.

Graph-based planners guarantee optimal, complete paths in static environments by exhaustively constructing and searching the entire graph, but this upfront effort and the resulting exponential state space growth make them impractical for high-dimensional or dynamic problems [8,18]. In contrast, sampling-based methods offer probabilistic completeness converging to a solution as the number of samples increases [14,15]. Furthermore, they support real-time replanning through incremental reuse and extension of existing samples, whereas graph approaches typically incur significant overhead rebuilding or heavily editing their graphs when the environment changes.

Dijkstra:

Dijkstra's algorithm is one of the most widely used methods for finding shortest paths from a single source to all other vertices in a weighted, directed graph with nonnegative edge weights [12,18]. The algorithm is a classical greedy procedure for computing single-source shortest-path distances in graphs. The algorithm initialises the source vertex $d(s) = 0$ and assigns all other vertices $d(v) = \infty$. At each iteration, it extracts the unsettled vertex with the minimum tentative distance, "settles" that vertex, and performs relaxation on its outgoing edges; for each neighbour, it updates the neighbour's tentative distance if traversing the settled vertex yields a shorter path. This process repeats until every vertex has been settled, at which point the recorded distances coincide with the true minimum-cost paths from the source to all other vertices. The algorithm is seen to be used in IP and UAV autonomous planning as a baseline for collision-free path computation in known, discretized workspaces [6] valued for its simplicity. However, it does explore all reachable vertices, even beyond the goal, which can be wasteful compared to goal-directed methods like A* [8,16,18,19].

A*:

A* is a best-first search algorithm that finds least-cost paths from a single start node to one or more goal nodes in a weighted, directed graph by combining Dijkstra's algorithm with greedy best-first search [8,19,20]. It maintains a priority queue (the "open set") of discovered but unexpanded vertices, each keyed by the following:

$$f(v) = g(v) + h(v) \quad (1)$$

where $g(v)$ is the cost-to-come from start to node v , and $h(v)$ is the heuristic estimate from v to goal.

The algorithm initialises the start node with $g = 0$ and $f = h(\text{start})$ assigns all other nodes $g = \infty$ and in each iteration removes the open set node with the lowest f -value. It then "relaxes" each outgoing edge updating a neighbour's g (and thus f) if a shorter path through the settled node is found. It inserts or reprioritises that neighbour in the open set [8,18]. This process repeats until a goal node is settled, guaranteeing optimality when h is admissible and consistent or the open set is empty (no path exists) [8,19,20]. Its heuristic function focuses the search on the goal, rather than blindly expanding outward in all directions. This means that it often computes fewer nodes before finding the optimal route. The algorithm utilises several distance functions depending on the circumstance. These include the Manhattan distance which is consistent on cardinal-only grids. It is known for its speed and simplicity when diagonal steps are not being used. The Euclidean distance (seen in this paper) is as follows:

$$h(v) = \sqrt{(x_v - x_{goal})^2 + (y_v - y_{goal})^2} \quad (2)$$

where (x_v, y_v) and (x_{goal}, y_{goal}) are node and goal coordinates.

This is suitable when diagonal movement or free-form movement truly reflect the geometric distance, and you can afford the extra computation to ensure the distance is as accurate as possible. For 8-connected grids, the Chebyshev distance offers a cheaper-to-compute yet tighter bound option than pure Manhattan on diagonal moves [8,19,20]. While Euclidean heuristics yield the fewest node expansions, Manhattan and Chebyshev are often preferred in time-critical applications due to their simpler arithmetic and lower computational overhead. Euclidean will be used in this project due to the need for diagonal movement.

When running A* in the project, the study will be using an added heuristic weight ($\epsilon = 1.5$). Weighted A* modifies the evaluation function by introducing a weight $\epsilon \geq 1$ on the heuristic term, so that

$$f(v) = g(v) + \varepsilon h(v) \quad (3)$$

where $\varepsilon \geq 1$ is the heuristic inflation factor

By inflating h , the search becomes more aggressively goal-directed, often exploring far fewer nodes than standard A^* at the cost of guaranteed optimality. When h is admissible and consistent, weighted A^* will return a solution whose cost is at most ε times the true optimum [9,10]. In practice, small weights typically yield near-optimal paths with significant speedups, while larger weights further reduce runtime but can lead to increasingly suboptimal routes [8,19]. Recent works continue to explore weighted A^* and real-time heuristic search [16,21].

RRT:

Rapidly exploring random tree (RRT) is a random sampling-based algorithm designed to efficiently explore high-dimensional continuous state spaces. Rather than constructing a full, discrete graph of the environment like Dijkstra or A^* , RRT incrementally grows a tree of collision-free configurations V connected by feasible local motions E . Starting from the initial state q_{start} , it repeatedly samples a random configuration q_{rand} (often uniformly in the free space) with occasional “goal bias” toward q_{goal} , finds the nearest existing tree node q_{near} under some metric d , and “steers” a fixed step δ from q_{near} toward q_{rand} . If the straight-line segment to the new sample q_{new} is collision-free, RRT adds q_{new} to V and the edge (q_{near}, q_{new}) to E . This simple sample–extend–check loop continues until the tree reaches within a user-defined threshold of the goal or until a time or iteration limit is exceeded, at which point a feasible path is extracted by backtracking through the tree edges [22].

RRT uses Euclidean distance to identify its nearest neighbour and uniform random sampling to ensure that the tree rapidly spreads through the free space. RRT is only probabilistically complete; it does not guarantee optimality. Its main power lies in its simplicity and scalability. The simple loop can handle high-dimensional and continuous planning problems that defeat grid-based search options [22].

RRT* and cost-aware RRT*

RRT* was introduced in 2010 as an improved version of RRT. The algorithm maintains probabilistic completeness whilst also adding path optimality [5,11] allowing it to converge to an optimal solution. The algorithm, however, requires more computational power than RRT as it must reselect the parent node and re-route the path.

The RRT* used differs from the standard version by incorporating cost awareness that calculates movement costs (distance multiplied by movement_cost) based on the environment, allowing the algorithm to account for penalty zones and reward regions and helping to create more optimal routes like A^* . Without this feature, the sampling algorithms would have no chance of keeping up with the deterministic algorithms in terms of optimality. Whilst traditional RRT* uses Euclidean distance alone to evaluate path quality, this version (CARRT*) multiplies the Euclidean distance by location-specific movement costs, enabling more realistic mission-aware planning. As a result, the algorithm adapts its growth and rewiring strategy to avoid high-risk areas and exploit beneficial ones. The pseudocode for CARRT* can be seen in Algorithm 1.

Algorithm 1. CARRT* pseudocode

```

1   $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset; \text{parent}(x_{init}) \leftarrow \text{NIL}; \text{cost}(x_{init}) \leftarrow 0$ 
2   $\text{nodes\_explored} \leftarrow 0; \text{final} \leftarrow \text{NIL}$ 
3  for  $i = 1, \dots, n$  do
4     $x_{rand} \leftarrow (\text{Rand}() < p_{goal}) ? x_{goal} : \text{SampleFree}()$ 
5     $x_{near} \leftarrow \text{Nearest}(G = (V, E), x_{rand})$ 
6     $x_{new} \leftarrow \text{Steer}(x_{near}, x_{rand}, \eta)$ 
```

```

7   if ObstacleFree( x_near, x_new ) then
8       X_near ← Near( V, x_new, r_near )           // Euclidean radius
9       // ----- choose parent (classic edge cost = length) -----
10      x_par ← x_near
11      c_min ← cost(x_near) + DIST(x_near, x_new)
12      for each x ∈ X_near do
13          if ObstacleFree( x, x_new ) then
14              c ← cost(x) + DIST( x, x_new )
15              if c < c_min then c_min ← c; x_par ← x end if
16          end if
17      end for
18      V ← V ∪ {x_new}; E ← E ∪ {(x_par, x_new)}
19      parent(x_new) ← x_par; cost(x_new) ← c_min
20      nodes_explored ← nodes_explored + 1
21      // ----- rewire neighbors -----
22      for each x ∈ X_near \ {x_par, x_new} do
23          if ObstacleFree( x_new, x ) then
24              c_rewire ← cost(x_new) + DIST( x_new, x )
25              if c_rewire < cost(x) then
26                  E ← (E \ {(parent(x), x)}) ∪ {( x_new, x )}
27                  parent(x) ← x_new; cost(x) ← c_rewire
28              end if
29          end if
30      end for
31      // ----- snap/upgrade goal if close -----
32      if ||x_new - x_goal|| < η and ObstacleFree( x_new, x_goal ) then
33          if final = NIL then
34              final ← x_goal; parent(final) ← x_new
35              cost(final) ← cost(x_new) + DIST( x_new, final )
36          else
37              c_new ← cost(x_new) + DIST( x_new, x_goal )
38              if c_new < cost(final) then
39                  parent(final) ← x_new; cost(final) ← c_new
40              end if
41          end if
42      end if
43  end if
44 end for
45 if final ≠ NIL then return (ExtractPath(final), cost(final), nodes_explored)
46 else return (FAILURE, ∞, nodes_explored)

```

Similar cost-aware sampling methods have been explored in the recent literature, [5], where RRT*-based planners are adapted using dynamic cost maps. In this paper, this cost-aware RRT* (CARRT*) is used throughout.

Other algorithms used in practice

Many industry-standard algorithms have built on deterministic and sampling formulations to more flexibly incorporate complex cost functions. In graph-based algorithms, extra algorithmic functions are added to account for dynamic situations. ARA* and D* Lite introduce dynamic replanning capabilities. ARA* starts with a highly inflated heuristic ($\epsilon \gg 1$) to rapidly find a suboptimal path and then gradually reduces ϵ toward 1 to tighten optimality bounds, all while reusing prior search effort [22,23]. D* Lite similarly

allows fast repair of existing paths when edge costs change making it well suited to environments with unfolding no-fly zones or time-dependent risk fields.

Other algorithms such as biology-inspired algorithms and machine learning algorithms are utilised in industry. Genetic algorithms (GAs) emulate natural selection by evolving a population of candidate paths through selection, crossover and mutation [24,25]. They excel at complex, multi-objective problems such as cooperative multi-UAV mission planning [22]. However, they often converge prematurely and run slowly, necessitating careful parameter tuning or hybridization to boost performance. Machine learning (ML) methods have enabled UAV planners to adapt trajectories in unpredictable, real-time environments. However, their performance hinges on high-quality data, careful hyperparameter tuning, and robustness to distribution shifts, often requiring domain adaptation or hybridised strategies [17,26].

Cost functions

Making use of a global cost function can be valuable when prioritising different objectives. In UAV path planning, cost functions are typically formulated as weighted sums of metrics such as distance, energy consumption, risks avoided and mission rewards gained to produce a single scalar objective suitable for graph-based or optimisation algorithms [2,18,20]. Different metrics are prioritised for different reasons. It is easy to presume that the minimum path length should be prioritised. As the route that minimises the total distance travelled, reduces flight time and enhances task efficiency under constant-velocity conditions, this path corresponds to the shortest flight duration [19,20]. However, prioritising energy consumption may be more important in situations where UAV fuel is running low. The study can focus on minimising UAV energy usage by considering the number of direction changes per flight. A useful global cost function should provide the ability to optimise energy efficiency during the mission [4,27]. The use of penalty and reward zones in our global cost function to help avoid/prioritise certain areas can also be implemented. Risk areas could consist of enemy airspace, natural disasters or crowded airspace whereas reward areas could consist of high windspeed areas or allies' airspace and need to be considered to various degrees when path planning. This is made possible by incorporating them into our global cost function.

Disclaimer: Each penalty/reward zone in the environments has a separate local cost function. Local cost functions are made up of a variety of mathematical functions which use intensity and a multiplier to shape each reward/penalty zone. The global cost function is not to be confused with our local cost functions. The former is applied post hoc to help analyse algorithms/environments whilst the latter are built into the environments

Deterministic planners excel when objectives align with geometric optimality, while sampling methods offer flexibility in complex spaces. However, prior evaluations typically bind the cost model to the planner and report results for one objective at a time, limiting reuse when priorities change. Soft penalty/reward area and energy turn costs are under-represented and rarely subjected to weight-sensitivity analysis. We address these gaps by decoupling path generation from post hoc evaluation via a cost function, along with introducing parameterised environments with varying penalty/reward structures. Lastly, providing a systematic weight sweep and sensitivity study clarifies which algorithms to prioritise.

While the global cost function is identical across all evaluations, perfect objective parity between deterministic and sampling planners is inherently unattainable. The two families operate under fundamentally different search structures. Graph-based methods expand deterministically on discrete vertices by calculating the least expensive route looking at a base cost, while sampling-based methods explore continuous space probabilistically. Their optimisation dynamics differ in how paths are created along with how costs

are propagated and accumulated. Therefore, our framework focuses on post hoc parity in evaluation rather than enforcing strict equality in internal optimisation objectives.

There are several papers using cost functions to help optimise a specific mission objective [13,28]. Meng et al. utilise a similar cost function structure, allowing for energy use, path distance and other similar factors to be considered when scoring a path. This paper opts to build on this cost function structure by implementing P (penalty zone) and R (reward zone) into the function. This way, penalty and reward zones can be perceived as zones with varying optimality as opposed to no-go obstacle areas allowing different objectives to lean towards staying in these zones for varying amounts of time. Zhang et al. [5] present a path planning method aimed at improving UAV navigation in complex environments, with emphasis on producing feasible routes under challenging conditions. The contribution focuses on proposing and validating a particular path planning algorithm rather than providing a reproducible benchmarking framework for comparing multiple algorithms under a shared mission objective. In particular, it does not explicitly support rapid mission re-prioritisation (e.g., shifting from fuel conservation to safety) by re-evaluating the same generated paths under different weightings, nor is it structured to systematically compare planner families, such as deterministic grid-based search and sampling-based methods, under one consistent evaluation model. As a result, while method papers can demonstrate improved path generation in a target setting, they often leave open the practical decision problem of how to choose between fundamentally different planners when mission objectives change. This is where the use of a “post hoc cost evaluation function” is so beneficial, because there is no need to restructure any of the algorithms in order to work in accordance with a cost function, allowing a post hoc analysis to present us with the most suitable planner to use for our scenario. Thoma et al. [28] provide a strong motivation for explicitly treating the cost model as a first-class design choice in UAV planning, rather than assuming a single fixed weighted-sum objective. They compare multi-objective cost formulations and quantify how those choices affect practical outcomes such as reliability/failure behaviour, distance/time efficiency, and energy-related metrics. Crucially, their study highlights that performance can be highly sensitive to weight selection. This is an encouraging conclusion given that weight selection is at the heart of this paper. However, their contribution remains focused on improving and analysing the objective within a specific planning setup, rather than providing a general benchmarking methodology for comparing path planner families under rapidly changing mission priorities, reinforcing the motivation to create a method using a post hoc, cross-planner evaluation framework, along with there being no reproducible method of using both sampling and deterministic planners to fit specific needs.

Taken together, the recent work demonstrates clear progress in designing richer cost functions to encode mission objectives and proposing specialised planners for challenging environments [5,13,28]. However, the literature still tends to treat these as separate strands: cost function papers often evaluate a single planning setup, spending all their time on perfecting an algorithm as opposed to using the pros of each algorithm, while method papers typically validate one algorithm against limited baselines and fixed objectives. This leaves a practical gap for active decision making: how to pick the most fitting planner when mission priorities shift, without repeatedly redesigning planners or rebuilding environments and whilst using both sampling and deterministic planners. Motivated by these limitations, this paper adopts a post hoc, cross-planner evaluation perspective: paths are generated using standard deterministic and sampling-based planners, then consistently re-scored under a unified cost model that explicitly captures penalties and rewards alongside pathlength and energy. This creates a reproducible basis for mission re-prioritisation, sensitivity testing, and algorithm selection guidance under changing objectives.

3. Materials and Methods

Environments

To provide a good testbed for our reward penalty path planners, the paper designs strong environments to use during testing. The study used Jupyter Notebooks 7.5.2 and Python 3.14 to design 4 increasingly complex environments (see Figure 1) inspired by common strategies in UAV simulation frameworks [29].

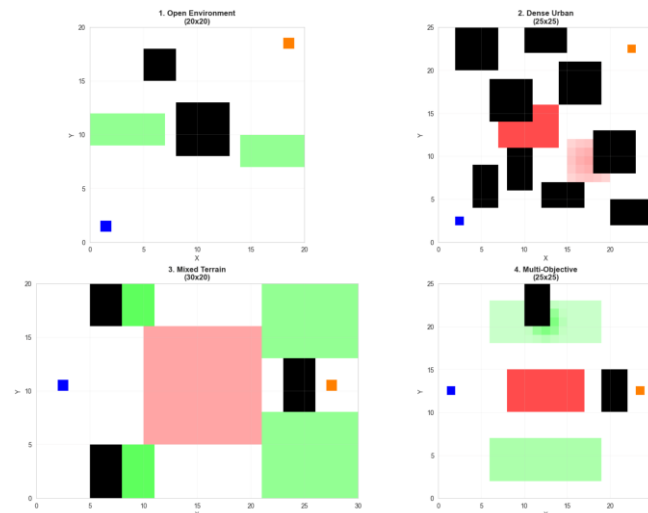


Figure 1. Four environments used in the study. Green represents reward areas whilst red penalty areas with darker colours representing stronger rewards/penalties. Blue represents the start point and orange the end. The same follows for all maps.

All environments make use of different local functions in their penalty and reward areas. The local functions used in each environment are defined throughout the chapter and are used to increase the complexity of penalty/reward areas, providing additional challenges for the algorithms and representing different real-life scenarios. The study used logarithmic, exponential, quadratic, step and Gaussian functions, reflecting how UAVs must navigate through diverse operational contexts with varying cost implications [13,30]. Two parameters, intensity and base multiplier, are used in conjunction with the local functions previously listed to punish or reward routes that stay in these areas for longer or shorter times. Sometimes you want a gentle push away from danger, sometimes a more extreme deterrent. These functions are used to account for these conditions.

The simplest environment is the open environment. This is a basic 20×20 grid with minimal obstacles and two reward zones. The environment tests the algorithms' ability to choose between the logarithmic safe corridor with gradual benefits and the linear energy efficient zone whilst avoiding obstacles. As the local functions are defined, let C_0 denote the base movement cost for an unobstructed cell, and m is the zone's cost multiplier ($m > 1$ for penalty zones, $m < 1$ for reward zones). This local logarithmic function can be written as follows:

$$f_{\log(C_0, m, i)} = \begin{cases} C_0[1 + \ln(1 + (m - 1)i)], & m > 1 \\ C_0 \max[0.1, 1 - \ln(1 + (1 - m)i)], & m \leq 1 \end{cases} \quad (4)$$

where i is an intensity parameter controlling the strength of the logarithmic adjustment ($i \in [0, 2]$).

With a base multiplier of 0.3 and intensity of 1.2, this corridor provides substantial initial cost savings upon first entry, reducing flight costs to approximately 30% of normal, making it an appealing option. However, a logarithmic function ensures that extended time within the corridor yields progressively smaller additional benefits. This mirrors

how benefit saturation works in real-world flight corridors or energy-saving paths in UAV missions, where diminishing returns discourage overly prolonged use [31,32].

This mathematical behaviour mirrors real-world scenarios like air traffic control, preferred routes or established safety corridors, where the greatest advantage comes from simply using the designated path, but staying in the corridor for extended periods does not provide proportional additional value. For pathfinding algorithms, this creates an interesting strategic decision: the corridor is highly attractive for short detours or connecting segments, but algorithms must weigh whether longer routes through the corridor justify the extra distance compared to more direct alternatives.

The paper's second environment is titled "dense urban". It is designed to simulate a city with the abundance of square and rectangle obstacles acting as buildings creating narrow corridors and dead ends, causing the algorithm to manoeuvre its way through shortcuts. The study made use of two different functions for the penalty zones. The first is an exponential function used to simulate a school that can be written as follows:

$$f_{\exp(C_0, m, i)} = \begin{cases} C_0 e^{(m-1)i}, & m > 1 \\ C_0 e^{-(1-m)i}, & m \leq 1 \end{cases} \quad (5)$$

where a larger i sharpens growth/decay.

The exponential curve captures the idea that a school zone should be treated almost like a no-fly area, without making it impossible (you can fly over it but the penalty you incur is very rarely worth it), which mirrors how UAV operators balance legal safety requirements against operational necessities [20,29]. It is likely that planners will either end up over-detouring to avoid any risk or waste time exploring the perimeter trying to find the one "least bad" entry point.

The environment also makes use of a Gaussian function to represent a hospital which can be written as follows:

$$f_{\text{gauss}}(C_0, m, i, d, d_{\max}) = \begin{cases} C_0 \left[1 + (m-1) \exp\left(-\frac{1}{2\left(\frac{3d}{d_{\max}}\right)^2}\right) i \right], & m < 1 \\ C_0 \left[1 - (1-m) \exp\left(-\frac{1}{2\left(\frac{3d}{d_{\max}}\right)^2}\right) i \right], & m \geq 1 \end{cases} \quad (6)$$

where d represents the distance from the zone's centre and $d_{\max}$ presents the maximum distance from the zone centre.

A Gaussian function is used because the core of the site is most sensitive (patients, critical operations) and noise or disturbance must be minimised, but a gentle buffer still exists around it. This reflects the concept of soft safety gradients used in dynamic risk-aware UAV planners [27,29]. Cutting through the edge of this zone would not be the costliest and should be considered where it could potentially make a path less costly.

The third environment, "mixed terrain", has an airspace designed to create a hard on/off barrier that can be passed but only at great expense. It does this by making use of a large central step local cost function:

$$f_{\text{step}}(C_0, m, i) = \begin{cases} C_0(1+i), & (m-1) > \tau, \\ C_0, & \text{otherwise.} \end{cases} \quad (7)$$

where τ is the threshold and is equal to 0.5.

Both north and south of this penalty area there are quadratic corridors which can be written as follows:

$$f_{\text{quad}}(C_0, m, i) = \begin{cases} C_0 [1 + ((m-1)i)^2], & m > 1 \\ C_0 \max\{1 - ((1-m)i)^2, 0.1\}, & m \leq 1 \end{cases} \quad (8)$$

where the floor $0.1 C_0$ avoids zero cost.

When cost drops sharply towards the centre of these reward tunnels, it encourages passing through them. This is consistent with path shaping techniques used in adaptive reward modelling for route optimisation [33,34].

Cells just outside the restricted zone may appear deceptively cheap relative to the core penalty, luring the search into circling the perimeter. The algorithms must weigh a very large shortcut penalty against much longer detours, bearing the question: is it worth enduring a large multiplier of base cost for a shorter path?

The last environment used is the multi-objective environment and it is designed to be the most complicated for the algorithm to deal with. It forces the planner to juggle three very different cost landscapes: a steep exponential “high-risk” trench down the centre that carries huge penalties; a smooth, bell-shaped Gaussian reward bubble to the north that entices detours but softly tapers off; and a logarithmic reward corridor to the south that rewards hugging its path yet contains a quadratic penalty area. Balancing these competing objectives, distance vs. exponential penalties vs. mild quadratic hazards, demands careful open/closed set management to avoid flip-flopping between paths or getting stuck exploring every fringe of the reward bubbles.

Although UAVs operate in three-dimensional space and are subject to motion constraints, this study focuses on global, mission-level route selection rather than low-level trajectory generation. The planners considered output plan-view waypoint sequences that can be executed using standard path-following and smoothing techniques, with vehicle constraints enforced at the tracking or control layer, allowing the findings to be easily applied to UAV use. 2D waypoint-based abstractions are widely adopted in UAV mission planning, particularly in data collection and monitoring applications where objectives and constraints are defined over a ground-referenced workspace and routes are evaluated at the mission level [35,36]. Accordingly, the analysis evaluates mission-aware trade-offs at the route-selection layer, with extension to full 3D planning left for future work.

Running Algorithms

The study opted to test these environments with the outlined algorithms, so the following are carefully implemented and applied: Dijkstra’s, A* (once with heuristic weight of 1 and once with 1.5), RRT and CARRT*. The study did so without the use of a main cost function to help understand their behaviour and see if the environments presented the challenges they were intended to. The study went on to compare algorithms in Python 3.14. For each algorithm, their elapsed time, number of nodes explored, total path cost (summing per-cell movement costs derived from our penalty and reward systems cost local functions), and the reconstructed sequence were recorded in a data class. After execution, a side-by-side table of results is printed to get a basic understanding of the algorithms in the environments (see Table 1).

Table 1. Algorithm results comparison.

Algorithm	Path Cost	Nodes	Time	Path Length
Dijkstra	21.72	517	0.005	24
A* (h-weight = 1)	22.16	134	0.001	24
A* (h-weight = 1.5)	25.90	34	0.000	24
RRT	244.19	150	0.003	25
CARRT*	20.82	2840	1.440	17

The study also opted to visualise each route on a colour-coded map background to help understand how each algorithm negotiates the environments (Figures 2–5).

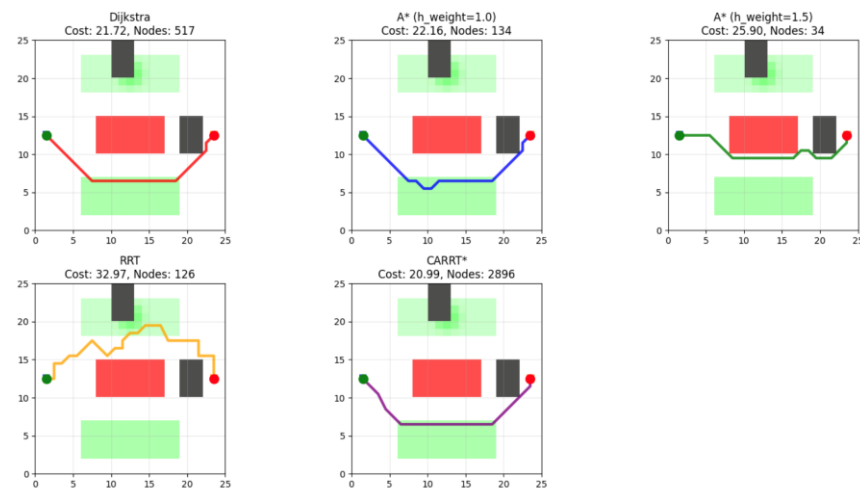


Figure 2. All algorithms on multi-objective environment.

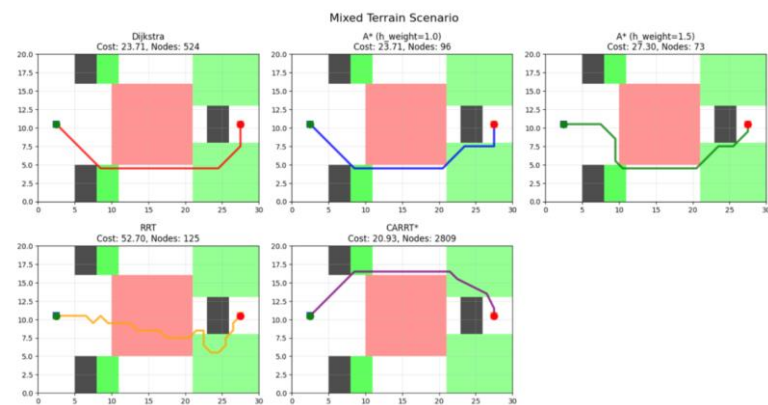


Figure 3. All algorithms on dense urban environment.

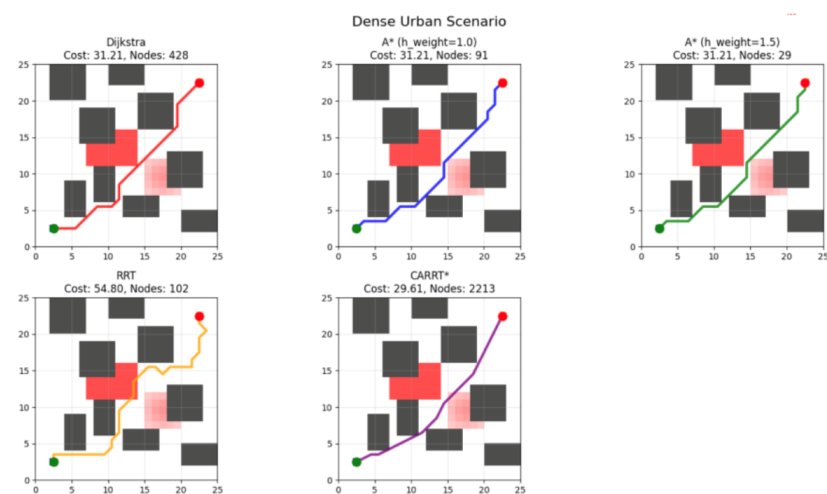


Figure 4. All algorithms on open scenario environment.

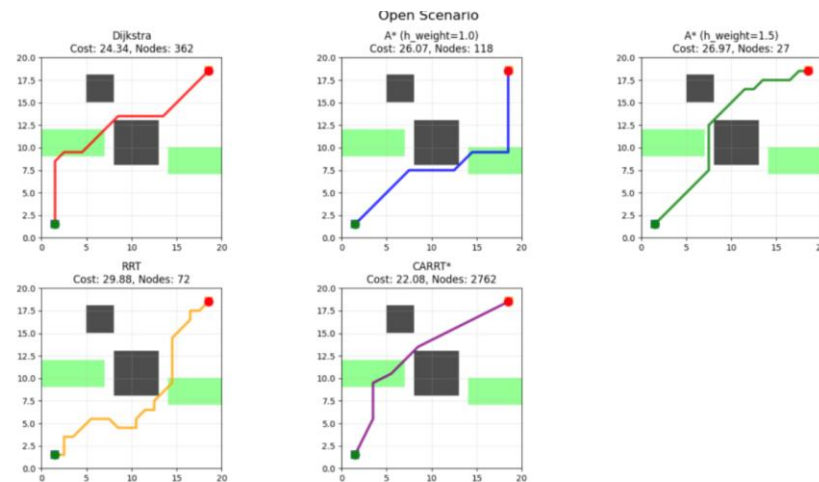


Figure 5. All algorithms on mixed terrain environment.

Disclaimer: Dijkstra/A* (in our main comparisons) plans on base grid cost C_0 and are evaluated post hoc with J; cost-aware RRT* queries c_{move} (a built-in property) during expansion to notice penalty/reward zones and is also evaluated with the same post hoc J for fair comparison.

A summary of all algorithms behaviour on my environments can be seen below (Table 2).

Table 2. Initial takeaways from algorithm performance on the environments.

Environment (Key Costs)	Dijkstra	A* ($\epsilon = 1.0$)	Weighted A* ($\epsilon = 1.5$)	RRT	CARRT*
Open—log reward corridors	Uses corridor but not aggressively. Solid cost with many expansions.	Exploits corridor briefly, balancing distance vs. reward. Far fewer nodes than Dijkstra.	Most goal-directed, misses rewards, fewest nodes, slightly higher cost.	Skims corridor, jagged high-cost path.	Stays in corridor longest to harvest reward, lowest cost, but very high samples/runtime.
Dense urban—exponential penalty, Gaussian soft zone	Avoids exponential penalty zone, cautious routing around blocks; high expansions.	Cuts near Gaussian edge to trade distance vs. soft penalty, low expansions.	Similar route to A* but even fewer nodes, greedy push does not cross exponential wall.	Explores similar route as deterministic planners, clips exponential penalty zone.	Hugs soft edges, strongly avoids exponential, better cost than A*/Dijkstra.
Mixed terrain—central step penalty and quadratic reward tunnels	Wide detour around step, locks onto south reward zones.	Tracks south reward zones efficiently.	Greedy arcs, occasionally clips step or rewards, fewest nodes, higher cost.	Straight through penalty area, huge cost incurred.	Only algorithm capable of utilising north rewards, achieves lowest cost.
Multi-objective—exponential trench, Gaussian reward, log corridor	Large enough nodes for exploration to find south reward zone.	Similar path to Dijkstra.	Misses out on reward zones but does have the shortest path length.	Very high cost, manages to gain some rewards but not lots.	Very similar to Dijkstra again, except with a slightly smoother path.

Global cost function

After experimenting with the algorithms and environments, it was time to design a global cost function. The function's mission weights should be adjustable and used to prioritise and experiment with different mission objectives. First, the study defines adjustable mission weights $(\alpha, \beta, \gamma, \delta)$ allowing chosen parameters to be given different weights.

These are path length, energy, penalties, and rewards like those discussed in [13]. This global cost function will be calculated using the formula:

$$J = \alpha L + \beta E + \gamma P - \delta R \quad (9)$$

where L = total path length (measured using Euclidean distance), E = energy surrogate (defined below), P = accumulated safety penalties (unitless cost above base), R = accumulated rewards (unitless cost below base), and $\alpha, \beta, \gamma, \delta \in \mathbb{R} \geq 0$ are mission weights.

Energy/turn model (per step)

$$step_{energy} = base_{energy} \left[1 + 0.3 \frac{|\theta|}{\pi} \right] \quad (10)$$

where $base_{energy}$ is a constant per move (unitless), θ is the heading change at the step (rad), and π is pi; the total E is $\sum$ over steps of $step_{energy}$.

Penalising paths that use multiple direction changes and penalising sharp directional changes is heavily inspired by [18,26]. Each penalty/reward zone has a separate local cost function. Local cost functions are made up of a variety of mathematical functions which use intensity and a multiplier to shape each reward/penalty zone defined earlier. The global cost function is not to be confused with our local cost functions. The former is applied post hoc to help analyse algorithms/environments whilst the latter are built into the environments.

Running Algorithms and Applying Cost Function

To run the experiment in a safe and efficient manner, the study made use of caching to avoid rebuilding environments every time an algorithm's ran through them. Caching also ensures that every parameter configuration is evaluated consistently on the same environment instances, which is essential for fair comparisons. This is a common practice in reproducible robotics research [18,30].

The paper then proceeds to complete a full grid sweep post hoc, iterating over all combinations of the four mission weight parameters ($\alpha, \beta, \gamma, \delta$), followed by parameter sensitivity analysis allowing parameters to be adjusted accordingly. This adjustment improved the interpretability of results without altering algorithm behaviour or overall conclusions. Predefining this broader range from the outset would have yielded similar trends, differing only in scale rather than direction, as the underlying cost structure and planner performance remained consistent. Expanding the range mid-analysis does not alter the paths; however, it does make the influence of γ easier to see and interpret. Also, allowing L, E, P and R to compete more meaningfully in the global cost function better reflects realistic mission trade-offs. In the code we use $\alpha, \beta \in \{0.5, 1.0, 1.5, 2.0\}$, $\gamma \in \{0.5, 1.0, 2.0, 3.0\}$, and $\delta \in \{0.5, 1.0, 1.5, 2.0\}$, yielding 256 weight configurations applied across four environments and five algorithms (5120 runs). Each configuration was executed three times to ensure robust statistical comparison, particularly for sampling-based planners (RRT, CARRT*). The resulting 15,360 total runs (3×5120) were combined into a single aggregated dataset (CSV) for analysis.

To evaluate how different mission priorities affect the suitability of existing pathfinding algorithms, the global cost function is applied in a post hoc manner making use of the effectiveness of decoupling path generation and cost evaluation by computing candidate UAV routes using traditional methods, then assessing them using a mission-specific cost model. Thoma et al. [28] evaluate fixed planner outputs under varying cost-weight conditions to understand how trade-offs like obstacle proximity and vertical manoeuvring influence total mission cost. This method allowed rapid experimentation without modifying the core logic of each algorithm. It also mirrors real-world scenarios where planners may be fixed due to certification or hardware limitations, but mission objectives change dynamically. By decoupling the cost evaluation from the path generation process, the study was able to systematically explore a wide range of mission configurations that

achieve different mission objectives (e.g., low-fuel route, high-safety route) while maintaining consistency and comparability across all runs. Comparing the total costs of algorithms and environments under these set parameters provides answers to questions like “Which algorithm performs best when prioritising low fuel?” and “Which algorithm is able to prioritise safety whilst being computationally efficient?”. A table summarising the methodology can be seen below (Table 3).

Table 3. Table summarising methodology.

Aspect	Setting/Value
Total runs	Total runs: 15,360 runs = 256 weight combos $\times$ 4 environments $\times$ 5 algorithms $\times$ 3 trials (aggregated)
Runs per env algo-rithm pair	256 (all α , β , γ , δ combinations)
Weight sweep (post hoc JJ)	$\alpha \in \{0.5, 1.0, 1.5, 2.0\}$ $\beta \in \{0.5, 1.0, 1.5, 2.0\}$ $\gamma \in \{0.5, 1.0, 2.0, 3.0\}$ $\delta \in \{0.5, 1.0, 1.5, 2.0\} \rightarrow 256$ combos
Algorithms	Dijkstra; A* ($h = 1.0$); A* ($h = 1.5$); RRT; CARRT*
Planner hyperparameters	A*: Euclidean heuristic; weight = 1.0 or 1.5 RRT: Max iterations = 2000, step size = 2.0, goal bias = 0.1 CARRT*: As RRT + search radius = 3.0 (rewiring, cost-aware)
Base movement cost	$C_0 = 1.0$ (used everywhere unless zone modifies it)
Cost fields and guards	Functions: Linear, exponential, quadratic, logarithmic, step, Gaussian.
Energy model EE	Turn penalty per step = $\text{base_energy}[1+0.3 \theta /\pi]$
Mission cost JJ (post hoc)	$J = \alpha L + \beta E + \gamma P - \delta R$
Environments (grid, start $\rightarrow$ goal)	- Open 20×20 , (1,1) $\rightarrow$ (18,18) - Dense urban 25×25 , (2,2) $\rightarrow$ (22,22) - Mixed terrain 30×20 , (2,10) $\rightarrow$ (27,10) - Multi-objective 25×25 , (1,12) $\rightarrow$ (23,12)

4. Analysis, Results and Discussion

The first part of the analysis was filtering for successful runs and tracking the success rate of runs. The study then automatically computed and logged average overall and per-algorithm/per-environment success rates across the three sweeps. The first of these ranks the best parameter combinations for lowest total cost in an attempt to understand each parameter’s influence (Table 4).

Table 4. Top parameter combinations by mission cost.

Case	α	β	γ	Δ	Cost
1	0.5	0.5	3.0	2.0	25.00
2	0.5	0.5	0.5	1.0	27.68
3	0.5	0.5	0.5	2.0	28.54
4	0.5	0.5	0.5	0.5	28.88
5	0.5	0.5	0.5	1.5	29.58

It is noticeable that path length and energy consumption weights stay at their lowest value, and it is only safety penalties and reward benefits that are changing in the top five combinations. It can be taken away that γ and δ seem to be the most appropriate to tune to acutely alter mission costs. This is to be expected as avoiding bad zones and travelling

through good zones will help keep cost low. α and β staying low is interesting as it shows in our environments that path length and energy clearly have the greater influence on total cost. It is observed that there are no instances in the top five parameter combinations where γ and δ values were the same and α or β changed around them. This means that α and β changes are far more influential in total cost and the planners' routes may not be travelling through enough reward/penalty zones. This is to be always expected as path length and energy always influence total cost, but reward and penalty zones only effect the cost when in their respective zones.

The one-at-a-time sensitivity analysis (Figure 6) revealed that increasing α and β caused sharp increases in the total mission cost of +82% and +74% respectively, despite no change to the underlying path. This is consistent with a post hoc cost framework, where these parameters scale the evaluation of already-generated paths [37]. γ displayed a convex cost curve, indicating that small increases were manageable, but large values resulted in sharp cost escalation; this is likely to do with the local functions in the penalty areas not behaving linearly. δ , on the other hand, decreased the cost steadily (−9% overall), though the effect size was smaller. This is likely due to there not being enticing enough reward zones or only one or two environments having a beneficial enough reward zone to be passed through. The study raised the γ parameter range from 1.5 (0.5–2) to 2.5 (0.5–3) because of its low sensitivity and in future work would raise even higher along with δ to match α and β 's levels of sensitivity. This sensitivity analysis could be used further to tune parameters accordingly.

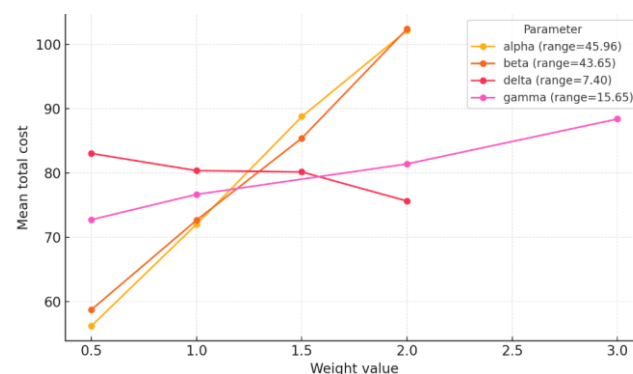


Figure 6. Parameter sensitivity analysis (averaged across three independent sweeps).

The heatmaps offer valuable insight into how combinations with different mission weights affect the evaluated cost of fixed paths [15], where the coordinates denote the total cost at different parameters. In the α – β heatmap, for example, the total mission cost increases smoothly and predictably as both weights rise. α and β serve as near-linear scalars on unchanging path lengths and energy values. More interestingly, the pairwise heatmaps involving γ and δ (Figure 7) reveal how the relative weighting of safety penalties and reward benefits shapes the overall mission evaluation.

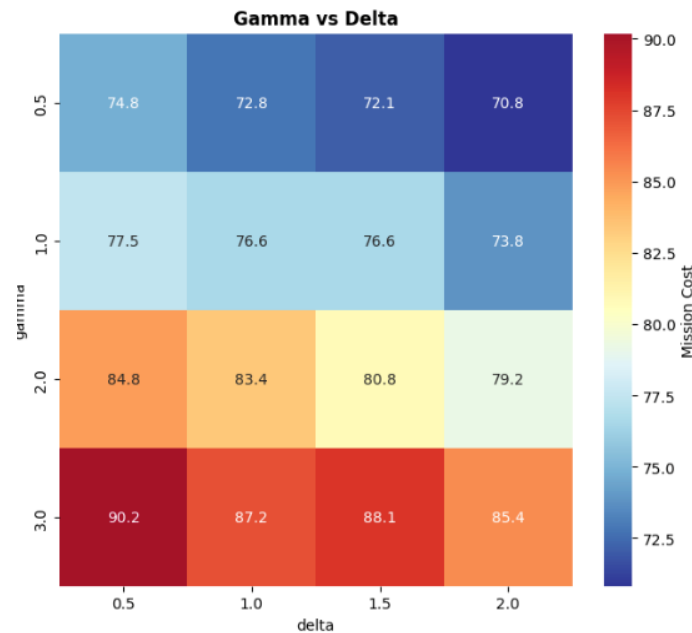


Figure 7. γ vs. δ heatmap (results from CSV averaged across three independent sweeps).

When γ is high and δ is low, costs climb rapidly, particularly for paths that enter or skirt high-penalty zones. Therefore, it is often optimal to minimise one whilst maximising the other to avoid any cancellation effects that could obstruct objectives. While these heatmaps cannot inform us about path selection, they can help identify which mission weights lead to higher or lower perceived mission costs under different scenarios, offering a practical tool for tuning mission parameters in environments where planners cannot be modified.

A useful function of this post hoc analysis is that all algorithms and environments can be compared under specific mission parameters to help understand which algorithms to use for certain mission objectives like a low-fuel route.

It is worth exploring how our cost function interacts with the algorithms in different ways. To do so, the study analysed a correlation heatmap showing, for each algorithm, how strongly each weight moves the total mission cost up or down by stating an r value (Figure 8).

Two behaviours emerge: the deterministic algorithms exhibit strong positive correlations with α and β (≈ 0.56 – 0.74), confirming that bumping up the distance or energy weight simply multiplies the cost of the chosen, often shortest, path route as seen earlier. These deterministic planners are very good at finding this route and it becomes apparent that often the risk incentives to cut through a penalty zone to shorten a path are not worth the cost benefits. Increasing A^* 's heuristic weight has almost no effect on these correlations ($h = 1$ being slightly more energy efficient), showing that the deterministic algorithms are all successfully finding the most optimal path length in the environments, even with a higher heuristic weight, and meaning it usually makes more sense to choose a high heuristic weight as it costs less computationally. In contrast, sampling planners' responses vary. The continual optimisation and altered cost awareness makes our CARRT* much more responsive to the cost terms and smoother path giving it an advantage on the deterministic algorithms when it comes to path weight. RRT, on the other hand, suffers from a "first valid path strategy". Because of this, it often finds short-enough paths quickly but does not refine them, resulting in weak correlations with α and β . δ behaves consistently, retaining a mild negative correlation (≈ -0.01 to -0.10) across every algorithm, so any method that can reward areas achieve the cost benefit. This shows that the environments

have low reward incentives, confirming δ 's intended behaviour, although maybe not having as extreme an effect as intended.

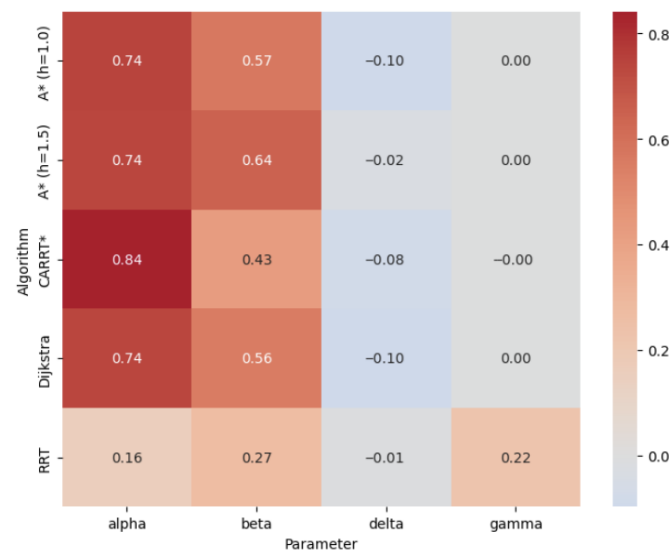


Figure 8. Parameter vs. algorithm heatmap.

The neutrality of the γ parameter with the deterministic algorithms suggests that these algorithms are structurally resilient to even the more complex penalty local functions and that they are enough to deter the algorithms from travelling through the penalty areas in the first place. Consequently, increasing γ has little to no effect on the final path selected, as those penalty areas are already being avoided through default graph traversal behaviour. CARRT* has a small exception as it is smart enough to skim through the outskirts of the Gaussian penalty section in the dense urban environment so to not acquire any large penalties. In contrast, RRT displays a moderate positive correlation with γ , indicating that as safety weighting increases, so does the total cost. This is likely due to RRT's clumsy method of exploring space. When γ is emphasised, the planner is penalised more for entering or skirting unsafe regions.

Among the tested algorithms, our CARRT* consistently achieved the lowest total cost and highest path efficiency, with an average cost nearly 10 units below any other algorithm. However, this comes at the cost of significantly higher computational overhead, as evidenced by both the number of nodes explored (1776) and the average runtime (0.63 s) (Table 5). In contrast, deterministic planners such as A* and Dijkstra completed in under 4 ms, highlighting their computational advantage for real-time applications.

When comparing both A*s, it is noticed that a higher heuristic weight ($h = 1.5$) has significantly fewer nodes explored and a runtime of less than half (0.00032 vs. 0.00076) but a better efficiency score (straight-line distance from start to goal/actual path length). However, due to its lack of nodes explored, it misses out on beneficial reward zones, that $h = 1$ did not miss out on. This gives $h = 1$ a better average cost; this was also noticed by Pohl et al. [9] and remains an active area of research in adaptive heuristic tuning for UAV planners [21]. Dijkstra performs competitively but lacks goal biasing, resulting in more nodes explored and a longer runtime (0.0028).

The basic RRT, while fast (for a sampling-based algorithm) (0.0016 s), suffers from poor path efficiency and high cost variance due to its “first valid path” nature, showing a clear trend in the cost vs. computation trade-off along with a direct relationship between average nodes explored and average runtime. Optimal planners like CARRT* are well suited to high-stakes missions where quality dominates, whereas lighter heuristics may be preferable for time- or resource-constrained contexts.

Table 5. Algorithm comparison table.

Algorithm	Avg Cost	Path Efficiency	Avg Nodes Explored	Avg Runtime
A* (h = 1.0)	61.60	0.822	110	0.00076
A* (h = 1.5)	66.83	0.855	41	0.00032
Dijkstra	60.18	0.837	458	0.002780
RRT	159.18	0.707	110	0.001644
CARRT*	50.34	0.869	1776	0.628612

To further evaluate computational efficiency, a Pareto front analysis was conducted between total mission cost and computation time (see Figure 9)

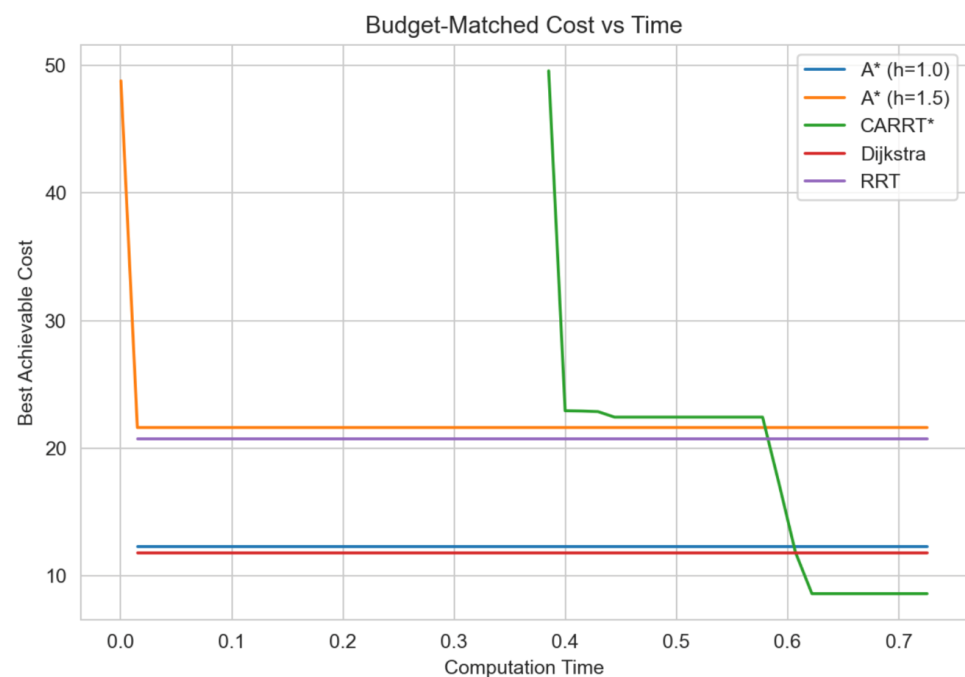
**Figure 9.** Pareto front of mission cost vs. computation time.

Figure 9 allows the paper to identify two specific regions. First, a low-time frontier dominated by Dijkstra and A* (h = 1) which achieves near optimal paths in milliseconds and, secondly, a high time frontier led by CARRT* achieving the lowest achievable cost when the computation time reaches 0.615. Weighted A* (h = 1.5) and RRT were consistently dominated, indicating that their additional runtime does not translate into improved cost performance. This trade-off visualisation reinforces earlier findings that deterministic planners are best for time-critical operations, whereas cost-aware sampling methods become preferable when computation budgets permit extended optimisation.

RRT efficiency and tuning

The observed high variance of RRT is consistent with the lack of optimality in its design. The algorithm stops searching as soon as it reaches its goal, and these early connections often produce jagged routes incurring excessive turn penalties, especially in cluttered environments where lots of obstacle avoidance is needed. Our post hoc energy term also penalises direction changes so even when geometric length is acceptable, turn penalties inflate the mission cost. Parameter tuning of step size and goal bias could be used to reduce variance and improve efficiency; however, it would not be able to remove the structural limitations that the algorithm poses. If the step size is too large, it would result in skipping around obstacles as well as steeper turns. On the other hand, too small would

result in many nodes and a slower algorithm. For goal bias, too high would result in bee-lining into obstacles near the goal, wasting iterations, whilst too small would again result in a slow convergence. In practice, setting the step size to the obstacle scale (≈ 0.5 – 2 cell widths) and using a moderate goal bias (≈ 0.05 – 0.2) will help performance. On top of this, I could look at introducing an iteration budget to reduce variation (at the cost of speed) and spline smoothing to improve energy efficiency. However, these adjustments do not change the algorithm's optimisation properties. In our experiments, RRT remains consistently outperformed by methods that explicitly re-optimize (CARRT*), especially when energy and risk terms are significant in the mission objective. Therefore, it makes sense to tune RRT to prioritise speed as it is the best option when opting for a quick sampling-based algorithm.

Post hoc cost analysis is highly effective for assessing algorithm suitability across diverse mission goals by applying a consistent cost function to precomputed paths [28,29]. This method allowed us to evaluate algorithm performance across five UAV missions using parameter filtering:

- Low-Fuel Route: High β (≥ 1.5), low α (≤ 1.0)—energy efficient.
- High-Safety Route: High γ (≥ 2.0)—safety focused.
- Speed-Priority: High α (≥ 1.5), low β, γ (≤ 1.0)—fast routes.
- Reward Focused: High δ (≥ 1.5)—reward seeking.
- Balanced Mission: All parameters = 1.0 —moderate approach.

The paths evaluated are the ones seen in Figures 2–5. In the low-fuel route scenario, where energy efficiency is prioritised, CARRT* clearly outperforms all other planners, achieving both the best single-run energy cost (9.791) and the lowest average energy cost (14.542) (see Table 6). This is consistent with CARRT*'s known ability to continuously refine solutions emphasising smooth, low-energy paths.

Table 6. Mission objective table.

Scenario	Runs	Best Algorithm	Best Metric	Average Cost
Low-Fuel Route	3840	CARRT*	14.494	78.58
High-Safety Route	7680	A* ($h = 1.0$)	1.000	84.88
Speed-Priority	1920	CARRT*	28.623	75.32
Reward Focused	7680	Dijkstra	3.897	78.35
Balanced Mission	60	CARRT*	40.012	57.94

Disclaimer: The best metric refers to the key performance metric that defines success for the scenario, e.g., for the high-safety route the paper refers to the best safety score. The average cost refers to how costly the average mission is over all runs under these conditions.

In the high-safety route scenario, all deterministic planners achieve perfect safety scores (1.000), with only RRT deviating significantly due to its non-deterministic path planning. However, despite the similar scores, the total costs differ across algorithms, with CARRT* incurring the lowest average cost (50.34) and Dijkstra and A* ranging higher (≈ 60 – 66), suggesting that while deterministic planners are equally effective at avoiding unsafe zones on these environments, CARRT* finds paths that avoid these zones with shorter or more efficient routes. Environmental variation is also notable: all planners achieve full safety in the open environment, but performance degrades slightly in dense urban or multi-objective contexts for RRT, highlighting the difficulty of some environments to navigate.

When prioritising speed (path length, not computation time), CARRT* again leads, achieving the shortest average path length (28.623) and lowest total cost among all planners. A* ($h = 1.5$) follows closely, with slightly longer paths but better performance than its more conservative variant ($h = 1.0$). This demonstrates that heuristic weighting within A* can improve the responsiveness to speed-oriented objectives whilst lower heuristic weights are more likely to go chasing reward zones.

The reward focused mission highlights a different challenge: maximising reward collection (via high δ). Surprisingly, Dijkstra achieves the highest average reward (3.897), narrowly outperforming A* (3.885), despite not being optimised for reward seeking behaviour. This suggests that in many cases, the most efficient or shortest paths inherently pass through reward areas, especially in structured environments like mixed terrain. CARRT* underperforms here, which may be due to its sensitivity to spatial reward layout; if rewards are unevenly or distantly placed, CARRT*'s optimisation may avoid them in favour of overall path smoothness.

Last is the balanced mission, which has all parameters set to the same value. Despite only 20 qualifying runs, CARRT* again emerges as the most cost-effective algorithm (avg. cost = 50.34). The extremely high cost and variance for RRT (avg. cost = 159.18) reinforce its unsuitability for general-purpose missions. This scenario also shows that deterministic planners like A* and Dijkstra fall into a middle ground, producing moderately high costs with relatively predictable performance (Figure 10). Interestingly, the cost across environments rises with terrain complexity, peaking in multi-objective scenarios. This aligns with earlier findings: post hoc evaluation reveals that planner performance is highly context-sensitive, and optimal performance arises when both parameters and algorithms are specifically tailored to the mission's core objective. This framework should therefore be used as a guide to use on other environments as opposed to inflexible facts about planners to help lazily pick from.

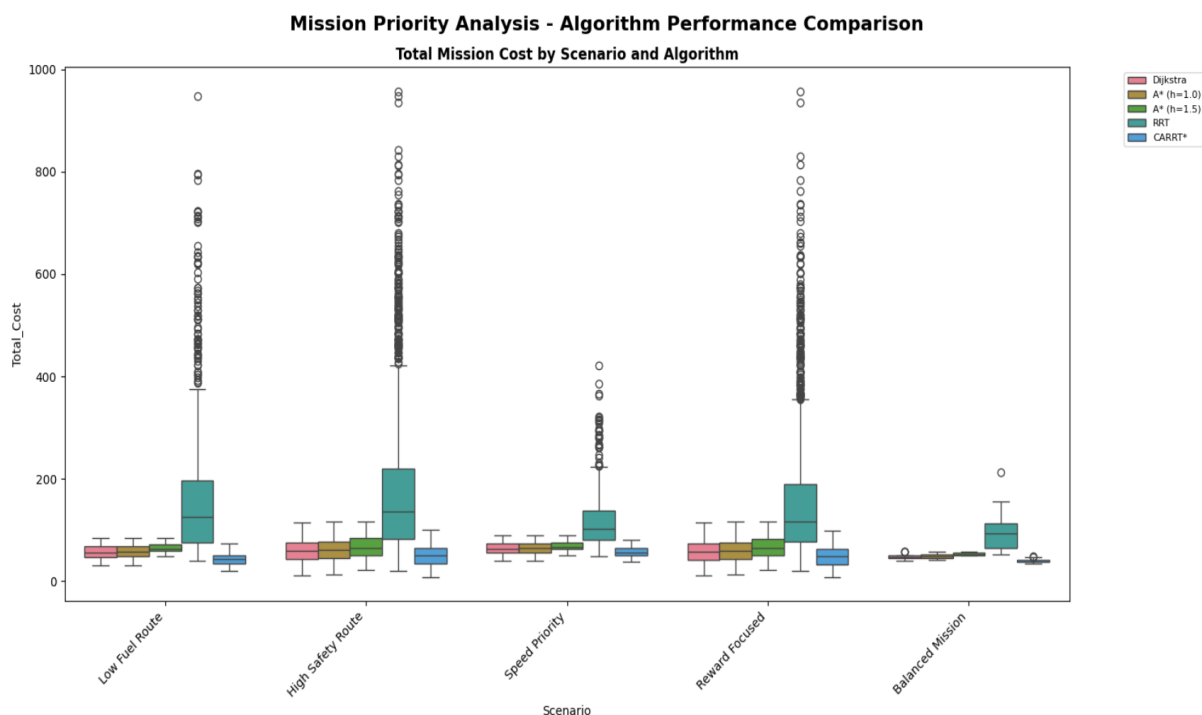


Figure 10. Mission objectives vs. algorithms. The circular markers denote statistical outliers, defined as mission costs lying beyond 1.5 times the interquartile range.

In summary, CARRT* excelled in low-fuel and speed-priority missions; deterministic planners performed similarly in high-safety missions; Dijkstra and A* were unexpectedly strong in reward-focused scenarios; and RRT was generally unsuitable due to its high variance.

In Figure 10 it can clearly be seen that RRT consistently performs the worst with huge outliers and CARRT* the best in total cost, with Dijkstra and A* ($h = 1$) performing well too. In cases where a low computation time is important, deterministic algorithms may be preferred, such as A* ($h = 1.5$). Finally, in conditions where reward areas are heavily favoured, Dijkstra is preferred. Please refer to Table 7 for more details.

All experiments were run three times on an Apple M2 CPU with identical environment and collision-checking parameters. CARRT* explored roughly 3–4× more nodes than Dijkstra or A*, explaining its higher runtime.

Table 7. Summary of mission objective findings.

Mission Objective	Priority	Favoured Algorithm	Key Strength	When to Use
Low-fuel	Energy efficient	CARRT*	Smooth, low-energy trajectories	Missions with limited battery/fuel or endurance-critical flights
High-safety	Obstacle and penalty avoidance	A* ($h = 1.0$), Dijkstra	Collision-free and great penalty zone avoidance	Search and rescue, disaster monitoring, or regulated airspace
Speed-priority	Shortest travel distance	CARRT*	Keeps travel distance low whilst maintaining a smooth path	Time-critical deliveries or rapid reconnaissance
Reward focused	Maximising reward regions	Dijkstra	Naturally explores beneficial corridors	Data collection or resource-mapping missions
Balanced mission	Equal trade-off between all factors	CARRT*	Strong overall performance	General-purpose or mixed-objective operations
Rapid approximation	Fastest computation time	A* ($h = 1.5$)	Quickest initial feasibility check	Rough trajectory generation or preplanning stage

Post hoc method evaluation

The paper's framework generates paths with existing planners and then re-scores them post hoc with a global cost function. This gives very high adaptability in mission prioritisation (weights can be changed instantly without touching certified planner code) but does not by itself change the path online. In contrast, real-time cost-aware planners (e.g., D*/RRT* variants with embedded costs) adapt the path during execution but require modifying/searching with the new cost model and re-verification. The paper argues that this trade-off is beneficial in practice as it provides adaptability via providing us with the framework to select between algorithms based on changing mission priorities. Many real missions experience objective change faster than map change (e.g., fuel drops, deadlines tighten, priorities shift to safety) while the geometric environment is relatively stable. In these cases, rapid re-prioritisation of the same candidate paths is exactly what is needed.

Although the framework only looks at static environments in this paper, it could be adapted to deal with dynamic environments by constantly re-scoring existing candidate paths when the position of penalty/reward zones change. If the new score of the cost func-

tion changes, we can trigger a replan; otherwise, we keep the current path. We keep regulatory constraints (no-fly areas) as hard feasibility constraints inside the planner, so that they are never traded against mission weights, and we reserve the global cost for tuneable preferences like distance, energy, and soft risk/reward. Crashes are seen as the worst possible scenario, and this helps avoid them at all costs.

5. Conclusions

The results show that while deterministic planners deliver consistent performance in structured or low-complexity settings, the CARRT* variant achieves the best overall mission cost and path efficiency across the full range of environments, including those with complex penalty and reward fields. The global cost framework that combines distance, energy, penalties and rewards proved effective for comparing planners under different mission priorities. Sensitivity and minimal-cost analyses further clarified how weight choices drive outcomes, offering a principled basis for setting parameters in future work.

Post hoc cost evaluation, although it does not modify the paths themselves, was crucial for mission-specific analysis, fair algorithm comparison and fast experimentation. Applying a single cost function to precomputed paths made it possible to reassess fixed planner outputs as priorities shifted, mirroring real deployments where planners may be fixed or certified, but mission requirements vary. This provides a practical and reproducible way to select algorithms and configurations that best match the operational objective.

Taken together, the study indicates clear tendencies: CARRT* excels for low-fuel and speed-priority missions; A* and Dijkstra perform strongly in high-safety contexts; Dijkstra can be competitive when rewards dominate; and basic RRT is generally unsuitable due to its high variance and poor path efficiency. Overall, aligning the algorithm choice and cost weighting with mission priorities is critical for achieving optimal UAV path planning performance, and post hoc analysis offers an efficient route to that alignment. This work demonstrates a reproducible framework for UAV path planning evaluation, enabling future researchers and practitioners to align the algorithm choice with mission-specific objectives.

6. Future Work

The study is easily extendable. It would be a useful exercise to run more algorithms through the environments to compare different pathfinding methods, broadening specialist algorithm selection to cater towards specific mission objectives. There are many different variations in RRT* that would be particularly useful to experiment with to cater for sampling efficiency and kino-dynamic and energy-aware RRT* for realistic vehicle limits. In a similar way, new environments of different dimensions or with dynamic elements could easily be added as an extension under the cost function, allowing us to broaden our algorithm selection further as well through dynamic pathfinding.

A big section of the work looked at parameter sensitivity analysis. This is a useful study as it improves the understanding of which parameter settings, relative to environments, have the biggest influence on mission cost and which parameters act similarly. To build on this, in future work these findings could be used to tune parameters, increasing the range of less sensitive parameters to allow γ and δ to influence the total mission cost more heavily, creating more balanced mission objectives. This would allow for more creativity in mission objective options. That being said, more mission objectives could easily be added as an extension of the project, and parameter tuning would just be used to give more punch to less favoured parameters.

Author Contributions: Conceptualization, U.B.C.; Methodology, J.R.K.; Validation, J.R.K.; Formal analysis, J.R.K.; Investigation, J.R.K.; Resources, U.B.C.; Data curation, J.R.K.; Writing—original

draft, J.R.K.; Writing—review and editing, U.B.C.; Visualisation, U.B.C.; Supervision, U.B.C.; Project administration, U.B.C.; Funding acquisition, U.B.C. All authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, U.B.C., upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. NASA Earth Observatory. Wildfire Maps Help Firefighters in Real Time. 2023. Available online: <https://www.earthobservatory.nasa.gov/images/154225/wildfire-maps-help-firefighters-in-real-time> (accessed on 20 August 2025).
2. Shakhathreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634.
3. Gupta, L.; Jain, R.; Vaszkun, G. Survey of important issues in UAV communication networks. *IEEE Commun. Surv. Tutor.* **2021**, *18*, 1123–1152.
4. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA, 2005.
5. Zhang, J.; Li, J.; Yang, H.; Feng, X.; Sun, G. Complex environment path planning for unmanned aerial vehicles. *Sensors* **2021**, *21*, 5250. <https://doi.org/10.3390/s21155250>.
6. Moy, J. RFC 2328: OSPF Version 2; IETF: Wilmington, DE, USA, 1998.
7. Liu, H.; Ma, R. Sky’s-Eye Perspective of UAV Highway Systems: Applications for Traffic Monitoring, Infrastructure Inspection, and Environmental Management. *Appl. Sci.* **2025**, *15*, 11199. <https://doi.org/10.3390/app152011199>.
8. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107.
9. Pohl, I. Heuristic search viewed as path finding in a graph. *Artif. Intell.* **1970**, *1*, 193–204.
10. Gammell, J.D.; Barfoot, T.D.; Srinivasa, S.S. Informed sampling-based planning: A survey. *IEEE Trans. Intell. Veh.* **2021**, *6*, 197–212. <https://doi.org/10.48550/arXiv>.
11. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894.
12. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271.
13. Meng, W.; Zhang, X.; Zhou, L.; Guo, H.; Hu, X. Advances in UAV Path Planning: A Comprehensive Review of Methods, Challenges, and Future Directions. *Drones* **2025**, *9*, 376. <https://doi.org/10.3390/drones9050376>.
14. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
15. Orthey, A.; Chamzas, C.; Kavraki, L.E. Sampling-based motion planning: A comparative review. *Annu. Rev. Control. Robot. Auton. Syst.* **2024**, *7*, 285–310.
16. Sotirchos, G.; Ajanovic, Z. Search-based versus sampling-based robot motion planning: A benchmarking perspective. In *Proceedings of the ICAPS 2024 Workshop on Planning and Robotics (PlanRob)*, Banff, AB, Canada, 3 June 2024; pp. 1–8.
17. McMahon, T.; Du, Y.; Bekris, K.E. A survey on the integration of machine learning with sampling-based motion planning. *Found. Trends® Robot.* **2022**, *9*, 266–327. <https://doi.org/10.1561/23000000063>.
18. Ghambari, S.; Golabi, M.; Jourdan, L.; Lepagnot, J.; Idoumghar, L. UAV path-planning techniques: A survey. *RAIRO—Oper. Res.* **2024**, *58*, 1569–1603.
19. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2010.
20. Sikora, T.; Papic, V. Survey of path planning for aerial drone inspection of multiple moving objects. *Drones* **2024**, *8*, 705.
21. Wang, N.; Li, X.; Zhang, K.; Wang, J.; Xie, D. A Survey on Path Planning for Autonomous Ground Vehicles in Unstructured Environments. *Machines* **2024**, *12*, 31. <https://doi.org/10.3390/machines12010031>.
22. Wu, T.; Zhang, Z.; Jing, F.; Gao, M. A dynamic path planning method for UAVs based on improved Informed-RRT* fused dynamic windows. *Drones* **2024**, *8*, 539. <https://doi.org/10.3390/drones8100539>.
23. Likhachev, M.; Gordon, G.; Thrun, S. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, BC, Canada, 13–18 December 2004.

24. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
25. Mo, F.; Zhang, Z.; Yu, L. An Improved Genetic Algorithm Analysis based on UAV Route Planning. In Proceedings of the 2023 International Conference on Data Science and Network Security (ICDSNS), Tiptur, India, 28–29 July 2023.
26. Abeywickrama, H.V.; Jayawickrama, B.A.; He, Y.; Dutkiewicz, E. Comprehensive energy consumption model for unmanned aerial vehicles based on empirical studies of battery performance. *IEEE Access* **2018**, *6*, 58383–58394.
27. Braßel, H.; Zeh, T.; Linder, M.; Fricke, H. Risk-aware UAV trajectory optimization using open U-Space TLS constraints. *Drones* **2025**, *9*, 666.
28. Thoma, A.; Thomessen, K.; Gardi, A.; Fisher, A.; Braun, C. Prioritising paths: An improved cost function for local path planning for UAVs in medical applications. In Proceedings of the 19th and 20th Australian International Aerospace Congress; Cambridge University Press: Cambridge, MA, USA, 2023. <https://doi.org/10.1017/aer.2023.15>.
29. Scherer, S.; Singh, S.; Chamberlain, L.; Elgersma, M. Flying fast and low among obstacles: Methodology and experiments. *Int. J. Robot. Res.* **2007**, *27*, 549–574.
30. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 28 September–2 October 2004.
31. Bai, H.; Hsu, D.; Lee, W.S. Integrated perception and planning in the continuous space: A POMDP approach. *Int. J. Robot. Res.* **2013**, *33*, 1288–1302.
32. Arnaldo, C.; Suárez, M.Z.; Pérez Moreno, F.; Delgado-Aguilera Jurado, R. Path Planning for Unmanned Aerial Vehicles in Complex Environments. *Drones* **2024**, *8*, 288. <https://doi.org/10.3390/drones8070288>.
33. Richter, C.; Bry, A.; Roy, N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In Proceedings of the International Symposium on Robotics Research (ISRR), Flagstaff, AZ, USA, 9 December 2016.
34. He, J.; Zhao, D.; Liu, T.; Zou, Q.; Xie, J. Research on Adaptive Reward Optimization Method for Robot Navigation in Complex Dynamic Environment. *Comput. Mater. Contin.* **2025**, *84*, 2733–2749. <https://doi.org/10.32604/cmc.2025.065205>.
35. Chaudhry, U.B.; Phillips, C.I. UAV Aided Data Collection for Wildlife Monitoring using Cache-enabled Mobile Ad-hoc Wireless Sensor Nodes. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 6–17. <https://doi.org/10.14569/IJACSA.2021.0121002>.
36. Chaudhry, U.B.; Phillips, C.I. Autonomous UAV Path Optimization Using Genetic and Multiobjective Evolutionary Algorithms for Effective Data Retrieval in Cache-Enabled Mobile Ad-Hoc WSNs. *J. Comput. Netw. Commun.* **2025**, *2025*, 8888509. <https://doi.org/10.1155/jcnc/8888509>.
37. Debnath, D.; Vanegas, F.; Sandino, J.; Hawary, A.F.; Gonzalez, F. A Review of UAV Path-Planning Algorithms and Obstacle Avoidance Methods for Remote Sensing Applications. *Remote Sens.* **2024**, *16*, 4019. <https://doi.org/10.3390/rs16214019>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.