Unmanned Systems, Vol. 10, No. 2 (2022) 129–146

DOI: 10.1142/S2301385022500078

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us

Comparison Between A* and RRT Algorithms for 3D UAV Path Planning

Christian Zammit*, Erik-Jan van Kampen

Control & Simulation, Aerospace Engineering,
Delft University of Technology, Kluyverweg 1, 2629 HS,
Delft, The Netherlands

This paper aims to present a comparative analysis of the two most utilized graph-based and sampling-based algorithms and their variants, in view of 3D UAV path planning in complex indoor environment. The findings of this analysis outline the usability of the methods and can assist future UAV path planning designers to select the best algorithm with the best parameter configuration in relation to the specific application. An extensive literature review of graph-based and sampling-based methods and their variants is first presented. The most utilized algorithms which are the A* for graph-based methods and Rapidly-Exploring Random Tree (RRT) for the sampling-based methods, are defined. A set of variants is also developed to mitigate with inherent shortcomings in the standard algorithms. All algorithms are then tested in the same scenarios and analyzed using the same performance measures. The A* algorithm generates shorter paths with respect to the RRT algorithm. The A* algorithm only explores volumes required for path generation while the RRT algorithms explore the space evenly. The A* algorithm exhibits an oscillatory behavior at different resolutions for the same scenario that is attenuated with the novel A* ripple reduction algorithm. The Multiple RRT generated longer unsmoothed paths in shorter planning times but required more smoothing over RRT. This work is the first attempt to compare graph-based and sampling-based algorithms in 3D path planning of UAVs. Furthermore, this work addresses shortcomings in both A* and RRT standard algorithms by developing a novel A* ripple reduction algorithm, a novel RRT variant and a specifically designed smoothing algorithm.

Keywords: Path planning; obstacle avoidance; 3D; A*; RRT; MRRT.

US

1. Introduction

Since the first industrial revolution, automation has been integrated into everyday life ranging from basic traffic lights to autonomous navigation of submarines. The introduction of these systems into military, industrial, commercial and private uses requires such systems to be robust and reliable to ensure a safe operation. These autonomous systems which can vary in scope, size and shape, are equipped with different sensory systems requiring various levels of intelligence depending upon their application. Advancements in control and navigation technologies and cost-reduction in hardware have made it possible for autonomous systems to operate safely and efficiently in a range of applications. The ever-increasing availability of autonomous systems will require more complex path planning systems so that all these systems can work efficiently without hinderance and possibly enhance the operation of one another.

One aspect of planning in autonomous systems is path planning. Path planning algorithms construct optimal paths to a preset goal point in the presence of time invariant and time variant environmental and vehicular constraints and uncertainties [1]. Path planning initiated as a 2D problem to generate paths for ground vehicles [2–4] or to approximate 3D path planning problems with 2D by assuming that 1 dimension is constant [5–9]. Eventually, with the introduction of complex 3D environmental and model constraints, research shifted its focus to 3D path planning methods

Received 25 February 2021; Accepted 11 July 2021; Published 8 October 2021. This paper was recommended for publication in its revised form by editorial board member, Jinwen Hu.
Email Address: *c.zammit@tudelft.nl*

[9–13]. The path planning system in 2D and 3D environments must deal with a set of constraints and requirements including static [14–16] and moving obstacles [17–19], moving targets [20, 21], weather [22, 23], no go zones [24, 25], communication [26–28], fuel [5, 25] and other model specific constraints [29, 30]. Some of these constraints, such as static obstacles and no go zones, can be accurately defined prior mission initiation. But in real situations, the majority of constraints can only be modeled with uncertainty, primarily weather and dynamic obstacles, although fuel consumption rate and vehicular state also incorporate an element of uncertainty. Besides, in real situations, constraints can change or even new ones pop-up and consequently the path planning system needs to adapt in real-time to ensure safe guidance to the final goal position [24].

To ensure safe and efficient path planning, numerous algorithms were proposed which can be segmented into two main approaches: *Graph-based and Sampling-based algorithms*. Graph-based methods define the environmental space by a set of grid points. Points residing on obstacles are defined as inaccessible grid points. The algorithm searches through the accessible grid points to construct a path from start to goal position if such a path is possible [31]. Graph-based methods guarantee a path from start to goal only for an adequate grid resolution [32]. Sampling-based methods select a random number of points in the environmental space and creates connections between them [32, 33]. Sampling-based algorithms are simple, efficient and probabilistically complete, i.e. they guarantee a solution at infinite time [32].

This paper provides a literature review of the current state-of-the-art path planning algorithms in the graph-based and sampling-based categories. The two most researched path planning methods, namely the A* and the Rapidly-Exploring Random Tree (RRT) from the graph-based and sampling-based categories, respectively, will be described, tested and analyzed in view of 3D UAV path planning. Furthermore, this paper aims to reduce the resultant path length ripple effects at different resolutions for the A* algorithm and the efficiency of the smoothing algorithm since in particular runs the reduction in path length is less than 0.1% even after numerous smoothing iteration attempts.

In this study, the performance parameters are the path length and computational time. Path length determines the time of traversal of a path and consequently the time the vehicle takes to arrive at the goal point. Fuel capacity and fuel efficiency determine the flight range. This is a primary constraint to full autonomy. This implies that path length will determine the effective range of the UAV. Ideally, the path planning time is as low as possible especially in dynamic environments with moving obstacles, since during path planning the environmental situation may significantly vary. Path planning time will impinge on the responsiveness of the path following algorithm to mitigate with both agent and environmental changes. Computational power onboard UAVs may also be very limited and it must be shared with other computationally expensive algorithms and sensing technologies. In this regard, the computational burden of path planning algorithms shall be as low as possible. The main contribution of this paper is an analysis of the A* and RRT algorithms in view of their suitability for 3D UAV path planning.

This paper will be organized as follows. Section 2 defines the standard A* and RRT algorithms, followed by the state-of-the-art graph-based and sampling-based methods in Sec. 3. Section 4 defines the A* ripple reduction (A^*_R) algorithm, RRT variants, the improved smoothing algorithm, vehicle model definition and the testing scenarios with their associated obstacle avoidance constraints. In Sec. 5, the results are analyzed followed by Sec. 6, which summarizes this paper by outlining the main strengths and weaknesses of the A* and RRT algorithms in view of real-time 3D UAV path planning.

2. The A* and RRT Algorithms

2.1. *The A* algorithm*

The A* algorithm constructs an optimal path based on an evaluation function $f(n)$ which calculates the actual cost of the path passing through generic node n, initiating from the starting point x_{init} to the goal node of n in M-dimensional space such that $x_{\text{init}} \in \mathbb{R}^M$ and $x_{\text{goal}} \in \mathbb{R}^M$ [34, 35]. As described by the below formula, $f(n)$ is the sum of the actual costs from x_{init} to a node n ($g(n)$) and from n to the goal point of n ($h(n)$), where $f, g, h :\in \mathbb{R}^M \to \mathbb{R}$:

$$f(n) = g(n) + h(n). \tag{1}$$

As the cost of the evaluation function $f(n)$ is a function of resolution, $g(n)$ and $h(n)$ can only be estimated. The estimated costs are denoted by $\hat{g}(n)$ and $\hat{h}(n)$, respectively, where the estimated evaluation function denoted by $\hat{f}(n)$ equates to

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n). \tag{2}$$

As the resolution $\to \infty$, the evaluation function$\hat{f}(n) \to f(n)$.

Algorithm 2.1 defines and Fig. 1 shows the rationale behind the A* Algorithm [34]. The A* algorithm starts by defining the start and goal nodes, x_{init} and x_{goal}, respectively. Then, $\hat{f}(x_{\text{init}})$ is evaluated for all possible *"Open"* nodes $n_i \in \mathbb{R}^M$ and the node with the smallest cost $c_i \in \mathbb{R}$ is assigned to n. *"Open"* nodes include all nodes n_i except for nodes that

Algorithm 2.1. Pseudo Code of the A* Algorithm

1: Assign x_{init} as an *open* node
2: Calculate $\hat{f}(x_{\text{init}})$ from the start node x_{init} to all *open* nodes n_i
3: Select the node n with the smallest $\hat{f}(x_{\text{init}})$
4: **If** $\hat{f}(x_{\text{init}}) == empty$
5: No path is possible
6: **End**
7: **While** $n \notin x_{\text{goal}}$
8: Mark node n as *closed*
9: Apply the successor operator Γ
10: Re-calculate $\hat{f}$ function for successor nodes of n and mark these nodes as *open*
11: **If** $\hat{f}$ is *empty*
12: No path is possible
13: **Else If** $\hat{f}(n_i)$ is now smaller than when n_i was *closed*
14: Remark successor *closed* nodes of n as *open*
15: **End**
16:**End**
17:Parent node n is *closed* (Path Found) [**34**]

reside on an obstacle and nodes already forming part of the optimal path. These are referred to as *"Closed"* nodes. No path exists if $\hat{f}(x_{\text{init}})$ is *empty* implying that the cost to all *"Open"* nodes is *null*. The A* algorithm continues by generating successive nodes using the successor operator (Γ). Through this operator, a set of successor nodes $n_i \in \mathbb{R}^M$ and their associated cost $c_i \in \mathbb{R}$ for each node n, are calculated based on a predefined heuristic graph movement, such that $\mathbb{R}^M \times \mathbb{R} \rightarrow \mathbb{R}^M$, provided the current node is not an element of x_{goal} otherwise the path from start to goal nodes is found [34].

2.2. *Rapidly-Exploring Random Tree*

The standard RRT algorithm grows a tree from the start point, by randomly planting seeds (nodes) in the configuration space, consequently growing tree branches creating a feasible path [36].

Algorithm 2.2 defines and Fig. 2 illustrates the rationale behind the RRT algorithm. The following will describe the parameters that are used to describe the standard RRT algorithm. $\tau \in (\mathbb{R}^M, \mathbb{R}^M \times \mathbb{R})$ refers to tree nodes and their edges with the first element at the starting node. X_{free} refers to unoccupied points in the environment space. $x_{\text{init}} \in \mathbb{R}^M$: start node, $x_{\text{goal}} \in \mathbb{R}^M$: goal node, $x_{\text{rand}} \in \mathbb{R}^M$: random node selected to construct new branches, $x_{\text{near}} \in \mathbb{R}^M$: nearest node to x_{rand}, $l_{\text{near}} \in \mathbb{R}^M$: nearest point on edge to x_{rand}, $x_{\text{edge}} \in \mathbb{R}^M$: nearest node on edge, $x_{\text{new}} \in \mathbb{R}^M$: new tree node D distance from x_{near} or l_{near} equal to the preset tree branch length and K: predefined maximum number of iterates to construct a tree from start to goal.

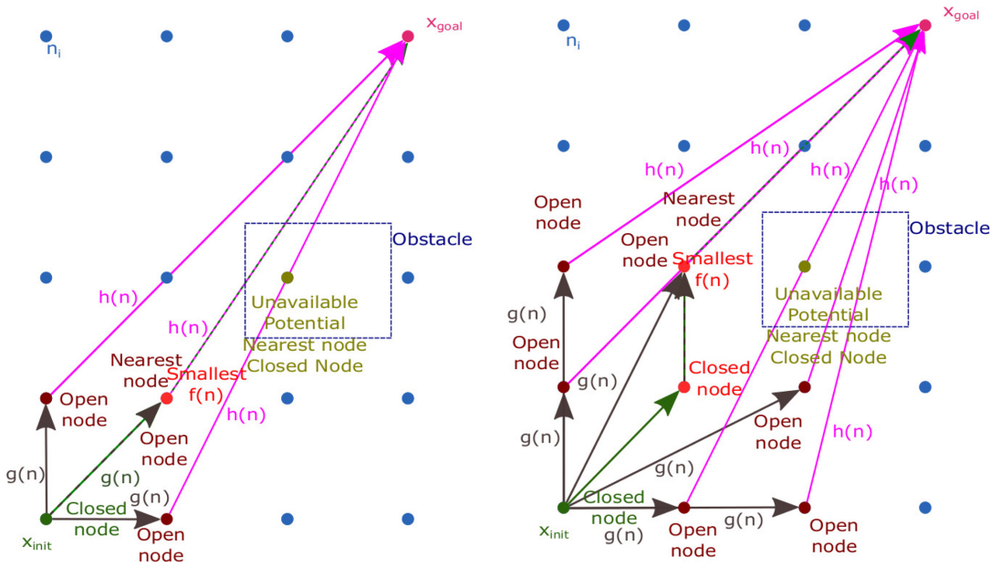


Fig. 1. Two planning iterations of the A* Algorithm: (a) first iteration and (b) second iteration.

Algorithm 2.2. Pseudo-Code of the RRT Algorithm

1: Assign x_{init} to first node of tree τ
2: **While** $i < K$ **AND** $x_{\text{goal}} \notin \tau$
3: Randomly generate $x_{\text{rand}} \in X_{\text{free}}$
4: Find the nearest node x_{near} and edge l_{near} to x_{rand}.
5: **If** Distance (x_{near} to x_{rand}) > Distance (l_{near} to x_{rand}), **then**
6: Calculate x_{new}, a distance D from x_{near} to x_{rand}.
7: **If** line connecting x_{near} to x_{new} does not result into a collision **then**
8: Add new node x_{new} to τ
9: **End**
10: **Else** Re-calculate x_{new}, a distance D from the nearest node on l_{near}, x_{edge} to x_{rand}
11: **If** line connecting x_{edge} to x_{new} does not result into a collision **then**
12: Add new node x_{new} to τ
13: **End**
14: **End**
15: **If** Distance from (x_{new} to x_{goal}) < D **AND** line x_{new} to x_{goal} is free **then**
16: $x_{\text{goal}} \in \tau$
17: **End**
18: $i++$;
19: **End**
20: **If** $x_{\text{goal}} \in \tau$ then path is found
21: **Else** No path found
22: **End**

3. Literature Review

The main scope of this literature review is to identify potential path planning algorithms for 3D UAV path planning of small UAVs operating in indoor environments using limited onboard computational resources. Current literature reviews discuss path planning algorithms either generically or their main scope differs from the scope of this work. In this regard, this section will focus on the implementation, utilization, advantages and disadvantages of graph-based and sampling-based path planning approaches and their variants in line with the scope. This review will conclude with a direct comparison of graph-based and sampling-based approaches in a common scenario.

3.1. *Graph-based approaches*

Graph-based methods have been extensively employed for path planning in different engineering problems due to their relative simplicity. A pioneer on graph-based search algorithms was the Dijkstra algorithm [37]. The A* algorithm, an advancement over the Dijkstra's algorithm, combines the benefit of heuristic methods with the Dijkstra's algorithm [38, 39].

3.1.1. *Dijkstra algorithm*

Dijkstra algorithm searches all possible options to find the minimum cost paths from two points [37]. The continuous Dijkstra, a variant of the original Dijkstra algorithm, finds the shortest path at boundaries through an analogy with

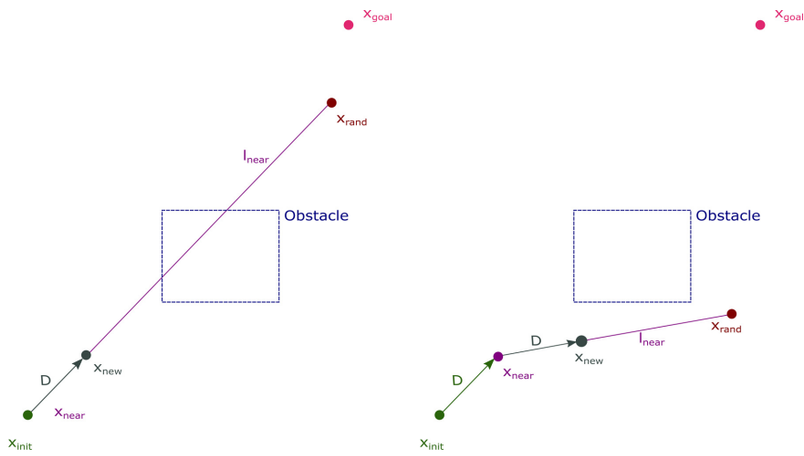


Fig. 2. Two planning iterations of the RRT Algorithm: (a) first iteration and (b) second iteration.

Snell's law of refraction [40]. An improved Dijkstra algorithm uses the Fibonacci technique to sequentially select ideal path planning positions resulting in faster, smaller weight paths [41]. The Dijkstra's algorithm was used for path planning in urban environments [42] in multiagent scenarios [43] and for autonomous driving in the DARPA challenge [44, 45].

3.1.2. *A* algorithm and its variants*

The A* algorithm is computationally efficient and guarantees the shortest possible path [38]. Path planning in both grid and visibility graphs can be computed using the A* algorithm. It was successfully implemented even for multiagent systems in dynamic environments [46]. A number of A* variants were developed to handle dynamic graph changes [39]. One variant, the Dynamic A* (D*) only updates new nodes, reducing computational demand [47]. Similarly, the focused D* reduces computational demand by heuristically searching previous search results [48]. Also, the Lifelong Planning A* (LPA*), an incremental A* variant, makes use of invariant parts from previous searches to reduce search time [49]. An advancement of the LPA is the D* Lite that uses one requirement to compare path planning priorities. This computational demand of D* lite is efficient at least as D* [50]. Another variant, the field D* re-plans using linear-interpolation techniques to efficiently construct smooth paths [51]. The L* algorithm, a faster variant than A*, ensures linear computational complexity and can be applied in global path planning situations [52].

The Theta* is another variant of A*. Two variants of Theta* exists namely: the Basic Theta* and the Angle-Propagation Theta* [53]. The Basic Theta* is relatively computationally efficient although it does not ensure that an optimal path is constructed [53]. Oppositely, the Angle-Propagation Theta* generates slower, more complex and longer paths with respect to the Basic Theta* although it generates better worst-case complexity per vertex [53]. The Phi*, a variant of the standard Theta*, re-plans by searching in all angles using only nodes within line-of-sight of their respective parents [54]. Overall, it requires more calculation time than the basic Theta* [54]. This time can be reduced by using the incremental Phi* which uses a pre-processing stage [54] developed for the differential A* algorithm [55]. Another variant, the lazy Theta*, developed by Nash *et al.* [56] reduces the number of line-of-sight checks effectively reducing planning time. This algorithm was used to operate a UAV in real-time outdoor environments [57].

Another variant, the Block A*, first smooths predefined short paths available in a look-up table and then starts the standard A* search [58]. This approach is consequently faster than both A* and Theta* but owing that path construction is dependent upon the look-up table, this approach does not guarantee a solution [54]. Another fast approach is the Jump Point Search (JPS) which exploits symmetries but neglects neighboring cells and does not search in every direction [59, 60].

The idea of using previous information to reduce computational time is the basis of other A* variants including Differential A* [55] and Fringe-Saving A* [61]. These methods initiate by re-planning iterations in the largest static area by comparing previous planning iterations [62]. Moreover, incremental heuristic algorithms, such as the generalized Adaptive A* [63] and path- and tree-adaptive A* [64], also use knowledge acquired from previous searches to enhance heuristics ultimately reducing computational demand. Such heuristics are a function of cost edges [62].

Anytime and truncated incremental techniques were also integrated with A* and its variants to improve performance. The integration of anytime methods on A* and D* is realized in the Anytime Repairing A* (ARA*), Anytime Tree Restoring A* (ATRA*) and Anytime D* (AD*). These variants construct sub-optimal paths within a predefined short time frame [4]. Similarly, Truncated Incremental Search methods were integrated with LPA* (TLPA*), D* (TD*) and D* Lite (TD* Lite). These techniques utilize information from previous iterations to focus the search expansion based on sub-optimal target bounds [65]. These variants improve performance at the expense of increasing computational demand and therefore these methods suffer in high-dimensional environments [51].

This review shows that the standard A* algorithm has been extensively applied in various fields through the evolution of its variants and its integration with other techniques. Its extensive use is derived from the low computational and optimal path benefits that this algorithm can construct.

3.2. *Sampling-based approaches*

Sampling-based methods are extensively used in various fields due to the their relative low computational requirements and guarantee of convergence [32]. The most commonly used sampling-based methods are the Probabilistic Roadmap Method (PRM) [66] and the RRT [67]. These methods construct cyclic and acyclic random trees, respectively [32].

3.2.1. *Probabilistic Roadmap Method (PRM)*

The PRM algorithm is made up of two phases: the learning and the query phase. In the learning phase, a local planner connects sample points to form a roadmap while the query phase make use of the previously constructed roadmap to

search a feasible path from start to goal point using a graph search algorithm [66].

To improve the performance of the standard PRM method, a set of variants was introduced. The PRM* improves path optimality by making use of a variant distance constraint [68]. To ease computational demand, the Lazy PRM initiates by neglecting all obstacles in the environmental space. Consequently, the algorithm constructs a path from start to goal and then checks for collision. Nodes and edges resulting in a collision are eliminated from the roadmap [69]. Similarly, the Dynamic Roadmap (DRM) method generates a roadmap by mapping the workspace maps to roadmap edges assuming an obstacle-free space as in the Lazy PRM. During online re-planning, edges residing on obstacles are neglected [70]. The cell-based PRM (CPRM) method segments the environmental space into cells, assigning high priority to cells close to both straight path segments and disconnected components. For faster re-planning, searches are biased to solution areas [71]. In this regard, Pomarlan and Sucan[72] allocated higher costs for edges residing near colliding edges diverging the planner away from obstacle zones.

More PRM variants the Reactive Deformation Roadmap (RDR) which generates a roadmap attracted by time-variant sub-targets and repelled by obstacles. Sub-targets and their respective reactive links are added and removed to ensure roadmap connectivity [73]. The Flexible Anytime Dynamic PRM (FADPRM) selects desirable areas by environment segregation. An A* search continuously improves the solution through priority heuristics focused in the vicinity of frontier edges [74]. The elastic roadmap variant connects roadmap points in a dynamic environment through feedback control [75]. The Grid Path Planning Roadmap Planning (GPRM) and the Particle Probabilistic Roadmap (PPRM) are another two variants with the former producing a grid environment and the latter utilizing a probabilistic approach [76].

3.2.2. *Rapidly-Exploring Random Tree (RRT)*

The RRT [67] constructs a unidirectional search tree starting from the start node by randomly generating and connecting states not residing on obstacles until one tree branch extends to the goal node. A distinguished advantage of the RRT with respect to the PRM is the consideration of kinematic and dynamic constraints in the path planning process [67]. Paths constructed using the standard RRT are computationally efficient even in complex situations although they suffer from path optimality [77–79]. The RRT trees grow evenly in free spaces unless restrictions or potentials are included in the random generation of nodes [80]. This allows the RRT algorithm to be used in non-holonomic and restricted environments. In this regard, the standard RRT algorithm was successfully implemented to construct collision-free paths in the presence of static obstacles [16].

The lack optimality is one of the major drawbacks of the standard RRT. To tackle this drawback, the RRT* [81] was developed. RRT* exhibits asymptotic optimality by introducing a minimum length cost, although it lacks in ensuring a local optimum at the start node [68]. During sampling, the RRT* neglects the configuration-cost function and considers only the quality of the path when introducing or removing nodes. Due to its asymptotic optimality, RRT* is slow in achieving an optimal path in complex high-dimensional environments [82]. The RRT*-Smart variant was developed to improve convergence with respect to RRT*, by constructing quasi optimum or optimum paths in a smaller time frame [83]. Another RRT* variant was developed to add probabilistic guarantees of completeness and optimality by constructing tree edges based on vehicle control inputs [84].

The bidirectional or RRT RRT-Connect grows two trees, one starting from the start and the other from the goal node [85]. The artificial field algorithm was integrated with the RRT-Connect algorithm to optimize path length [86] and with the RRT* algorithm to optimize convergence speed [87]. The Execution Extended RRT (ERRT) variant efficiently re-plans to improve path quality by using both cost priority search and previous nodes [88]. Also, Martin *et al.* [89] proposed the integration of evolutionary algorithms with bidirectional RRT creating the bi-directional Rapidly Exploring Evolutionary Trees (RET). By searching in the vicinity of neighboring nodes through balanced Spatial KD-Trees and using *a priori* environmental knowledge, results show an improvement in time variant environments [88, 89]. Gros *et al.* concluded that bidirectional trees are inherently less flexible in cluttered environments than unidirectional search tree algorithms [11].

Another RRT variant neglects the connection to the nearest node rule and connects a new tree node to all obstacle-free node [90]. Re-planning is done using an obstacle-free distance cost function in a predefined time and provided that no collisions exist prior re-planning. Just as for the cell-based PRM, the cell-based RRT uses probabilistic techniques and decomposes the environment for efficient path planning [91]. A set of decomposition granularities was assessed and results show that biasing the search in the vicinity of the goal-biasing improves success rate and reduces path length [91]. By using the previously constructed path segments and applying intelligent decomposition techniques based on the environmental situation, performance was improved [91].

The Transition-based RRT (T-RRT) variant adds nodes to low cost areas to focus the search to these regions using a metropolis-like transition test [78]. By eliminating the path

quality requirement in the construction of edges, path optimality is negatively affected [78]. An advancement over T-RRT, the T-RRT* combines low-cost area biasing of the T-RRT algorithm with the path quality constraint of the RRT* algorithm [78]. In T-RRT* different physical such as road limits and environmental including obstacles restrictions were used to formulate biases for efficient tree growth [2, 92, 93]. Similarly, Shang *et al.* [94] used previous planning knowledge to bias the search in a high-complexity 3D environment.

As with graph-based paradigms, anytime methods integrated with RRT algorithms [95–97] use previous searches to efficiently construct an optimum, converging and feasible path. In this regard, the Anytime T-RRT (AT-RRT) uses the anytime paradigm to T-RRT for path re-planning to converge the path to optimum [78]. An anytime integration to RRT is the RRT-Roadmap (RRM) which compromises between exploration and path quality based on RRT* [98]. This meta-approach amalgamates short-cutting with path hybridization [99]. The Dynamic RRT (DRRT) neglects child nodes of invalid tree nodes and re-plans from goal to current node [100]. Furthermore, the Anytime Dynamic RRT (AD-RRT) exploits the strongholds of both techniques [101]. A further enhancement is the multipartite RRT (MP-RRT) designed for unknown or dynamic environments, samples and re-plans as in ERRT, removing invalid nodes as in DRRT [102].

Due to their benefits, RRT algorithms are also applied in real-time path planning applications. In this regard, the RRT^X a real-time asymptotic optimum re-planning algorithm continuously constructs noncolliding and optimal paths in time-varying environments. The path from start to goal is re-planned by updating search tree when a potential collision is detected [103]. The Partial Motion Planner (PMP) dedicates a fixed time window for both path planning and traversal. Similarly to RRT^X, this real-time path planner constructs current to goal node path segments of predefined length. Then the environmental model is updated and the re-planning process restarts [104]. Another real-time planner, the Closed Loop RRT (CL-RRT), re-plans a tree from the current position in a predetermined time window using a closed-loop feedback method. This algorithm can be applied even for nonlinear controllers and agent models with unstable dynamics. The stabilization controller associated with CL-RRT improves prediction accuracy and rejects disturbances acting on the vehicles [2].

The Greedy Incremental Path-Directed (GRIP) constructs a path prior movement by growing a tree from current point in predetermined time by considering only kinodynamic constraints [105]. During movement, the algorithm uses free nodes found in previous planning stages to update the tree. The search is greedy biased provided that state-space exploration is probabilistically ensured [105]. Similarly, Taheri *et al.* [106] developed a Fuzzy Greedy RRT (FG-RRT) path planning algorithm that controls the tree edge propagation in configuration space, resulting in lower path planning time and complexity with respect to the greedy RRT algorithm.

Reconfigurable Random Forest (RRF) applies the underlying principles of MP-RRT and separates and grows disconnected trees, with the aim of interconnecting them when trees grow [107]. To mitigate with moving obstacles, the Lazy Reconfigurable Forest (LRF) keeps a forest of trees. As opposed to DRRT which validates the entire forest, LRF validates only path edges [108]. Similarly, Multiple RRTs (MRRT) simultaneously grows a set of trees starting from predetermined points including the start and goal points. The ultimate goal is to improve path planning and exploration [80]. The Multiple Incremental RRTs (MIRRT) variant uses incremental paradigms to keep trees constructed in previous iterations that may be used in future path planning [80].

The evolution of PRMs to RRTs to its numerous subcategories that even hybrid with other techniques shows the maturity and flexibility of sampling-based techniques. This literature review showed that these methods were applied for path planning in 2D and 3D in both static and dynamic environments using different vehicle model. Therefore, sampling-based methods can be considered as candidates for feasible path planning using a wide spectrum of vehicular systems.

3.3. *Comparing approaches*

Studies are compared path planning performance of inherent graph-based with sampling-based techniques. In this regard, Tsai *et al.* [109] first implemented the RRT algorithm for 3D path planning of multirotor Umannned Aerial Vehicles (UAVs) and the Dijkstra algorithm for path length reduction. In a second implementation using the same system, 3D path planning was achieved using the RRT-Connect algorithm. Path length was reduced through an A* variant that only made use of the flight path angle without grid discretization. In both cases, Bézier Curves were employed for 3D path smoothing. Tsai *et al.* [109] noted an improvement of computational demand and path efficiency by using RRT-Connect instead of the standard RRT, the amended A* instead of the Dijkstra algorithm and the addition of path smoothing through the Bézier curves [109].

Similarly, Rao and Williams [110] compared the RRT-Connect and A* path planning algorithms for Autonomous Underwater Vehicles (AUVs) to generate feasible energy-optimal 2D paths biased by ocean currents. An Iterative k-Nearest RRT (IkRRT) considered *k*-neighbors to adjust heuristics required to set up the Voronoi-biased

RRT-Connect tree [111]. Rao and Williams [110] concluded that both RRT-Connect and A* algorithms are potential candidates for this application. RRT methods found it difficult to find solutions in view of Voronoi-bias although the planner was able to avoid shallow regions. Oppositely, the A* methods performed well in strong current situations [110].

Graph-based and sampling-based paradigms employ different methodologies to construct a path from start to goal. Both have their inherent advantages and disadvantages which make each method ideal for different applications. Mainly, graph-based methods discretize the environment, reducing computational cost but limiting the possibility of available nodes based on the resolution that is selected. Oppositely, the nondiscretized sampling-nature of the other method, may be computationally expensive in a relatively large obstacle-rich environment but will gradually converge to a solution (if possible) provided there are no time restrictions. In conclusion, it is adequate to analyze both rationales to best select the most promising method for the considered scenarios.

4. Path Planning Algorithm Enhancements and Test Environment

4.1. *Introduction*

The main aim of this part is to describe the A* and RRT variants that are implemented to provide a better comparison between graph-based and sampling-based paradigms. Moreover, to investigate the effects of path smoothing and for fair comparison, the same smoothing algorithm is applied to both A* and RRT variants. Finally, the test environment constructed to assess the performance of each path planning algorithm is described.

4.2. *The A* Ripple Reduction Algorithm* (A_R^*)

The A*'s algorithm graph-based paradigm described in Sec. 2.1 introduces situations where a small increase in resolution results in a much longer or shorter unsmoothed path and vice-versa. This effect is explained in our previous work [112].

To attenuate this, all free graph points are shifted by a random distance varying between zero and half the distance between adjacent grid points. Obstacle planes are not shifted with the same distance. This creates a different grid layout for the same situation. This technique will ultimately average out large variations in path length segments created by the discretision nature of the A* algorithm.

In A*, each situation is repeated for 100 times in each scenario and resolution yielding the same path in all 100 tests. In A_R^*, a different random shift is added for each situation (scenario and resolution) creating 100 different paths. The mean path length for these 100 paths will effectively reduce the path length ripple. Refer [112] for further details.

4.3. *The RRT algorithm without step size constraint*

In this variant, the RRT algorithm's tree branch length defined in Sec. 2.2 is limited by the *step size*. In an alternative approach, the *step size* constraint is neglected and tree branches are connected between the nearest and randomly generated nodes through straight lines only if a collision-free path connecting these two points exists. The path following algorithm shall segment the path into trajectories in view of vehicle's kinematic and dynamic constraints. This algorithm is intended to offer a comparison between the standard RRT and MRRT algorithms.

4.4. *Multiple Rapidly-Exploring Random Tree*

The MRRT algorithm constructs a predetermined number of trees that are simultaneously expanded. Tree seeds are placed at start and goal positions with the others placed at random or preset positions [80]. Trees grow per iteration in the direction of the randomly-generated node as in RRT and ultimately interconnect neglecting *step size* constraints. Similar to the RRT, this algorithm terminates when a single tree connects the start and goal nodes.

4.5. *Smoothing algorithm*

A smoothing algorithm is applied to all the path planning algorithms considered in this study. This post path planning algorithm randomly selects two path points and randomly selects a point from each line connecting the selected node to their respective next path point. Path points in between the latter two points are neglected if a collision-free interconnection is possible. This process is repeated for 1000 times.

Path planning analysis of the A*-based and RRT-based algorithms shows that A* methods require fewer smoothing iterations to reach a point where path length reduction is negligible when compared with RRT-based methods. For fair comparison, the smoothing iteration stopping condition must be set at a point where the prospective improvement of future iterations is negligible for the problem scope. An indication of the prospective improvement of future iterations can be provided from the path length reduction of the

previous set of smoothing iterations. In fact, analysis shows that path length asymptotically approaches a minimum with the number of iterates. This rationale is used in the development of a variant of the smoothing algorithm.

This new smoothing variant is governed by the path length reduction improvement over a preset value of iterations, defined by *excess* and set to 20. The smoothing algorithm is required to compute at least *excess* number of iterations. If the smoothing algorithm has computed more than *excess* and improvement is below a predefined value (set at 1%) the smoothing process is finished otherwise the smoothing algorithm computes a new iteration. The smoothing process also stops when the smoothing iteration reaches the maximum preset value (set at 1000). For further details of this algorithm and discussion about the empirical definition of the described parameters, refer [112].

4.6. *Vehicle model definition*

The main scope of this study is to assess, compare and possibly enhance the performance of graph-based and sampling-based algorithms in static 3D environments. A generic approach is considered in both environmental and vehicular model definitions. As both methods and their variants were applied to wide spectrum of vehicles, we considered the simplest and most generic type of model, namely the point model, to attenuate or possibly eliminate the vehicle model's kinematic and dynamic effects and their estimations on the path planning performance for both algorithms.

The point model considered assumed that

(1) The vehicle can rotate any angle in all three dimensions on itself without changing position.
(2) The vehicle has no orientation as it is assumed to be symmetric in all dimensions.

The path planning algorithm generates a set of points that, when interconnected, connects the start and goal positions. The interconnection of these points can be done by any line, curve or hyperbola depending upon the kinematic and dynamic capabilities of the vehicle. The vehicle-dependent trajectories shall then be constructed by an associated path following algorithm using path points generated from the path planning algorithm. Ultimately, the path following algorithm will be integrated with the path planning algorithm, as in [113]. For the scope of analysis, we assumed that path segments connecting consecutive points are linked by straight lines.

Furthermore, in path construction, it is assumed that the point model can reside a buffer distance from the nearest point on obstacle planes without colliding with them. If a prospective path point or a point on the line intersecting two consecutive path points resides within a smaller distance than the described buffer distance, the path planner will invalidate the last created node and the path planner re-plans. This obstacle avoidance rationale is applied to all algorithms including the post processing smoothing algorithms. This distance is empirically defined as half the distance between grid positions in A*. To offer a fair comparison, the same distance is considered for RRT-based methods. Depending upon the kinematic and dynamic constraints of the considered vehicular system, this buffer distance can be increased or decreased appropriately. Also, a generic unit cube is considered in the definition of the environment. This allows the tests to be scaled to any environment and consequently adapt the buffer distance between vehicular path and obstacles.

4.7. *Scenarios*

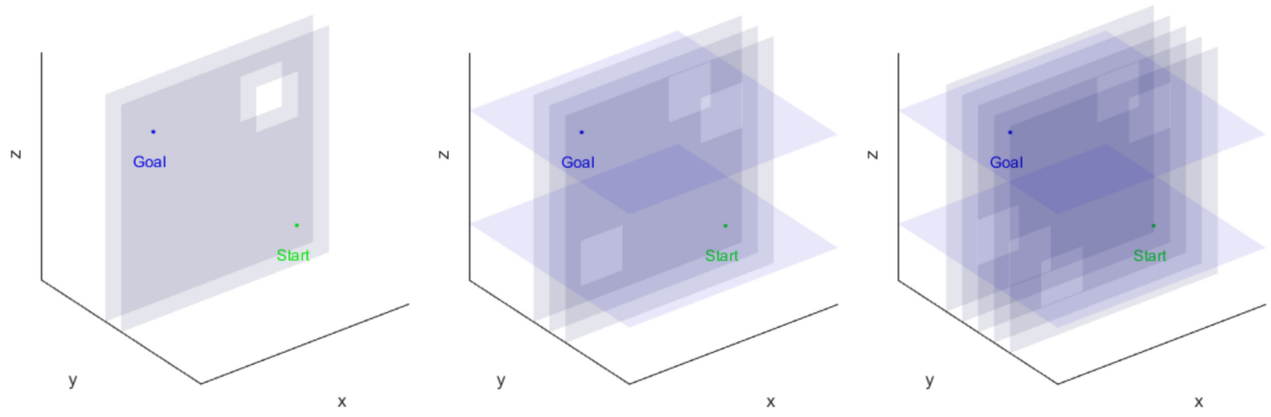
A unity cube with generic units is considered as the environmental space. The center of the cube is at the origin with limits from -0.5 to 0.5 in all three dimensions. The normalization of the environmental space allows the testing environment to be exported to real-life environmental spaces. The test scenarios that will be considered in this study where developed by Clifton *et al.* [114]. These are available online in [115]. The start and goal points are set at [0, -0.5, 0] and [0, 0.5, 0], respectively, for all tests. The following obstacle scenarios, illustrated in Fig. 3, are considered:

(1) Scenario 1: Two obstacle planes in the *X*–*Z* axis with 0.2×0.2 square openings;
(2) Scenario 2: Three obstacle planes in the *X*–*Z* axis with 0.2×0.2 square openings and two obstacle planes in the *X*–*Y* planes with no openings; and
(3) Scenario 3: Five obstacle planes in the *X*–*Z* axis with 0.2×0.2 square openings and two obstacle planes in the *X*–*Y* planes with no openings.

The considered control parameters are the resolution for A*, the step size per iterate for RRT and the number of seeds per axis for MRRT. The resolution is varied between 11 and 28 in steps of 2 for A* and consequently the step size per iterate for RRT is varied between $\frac{1}{10}$ and $\frac{1}{28}$ in steps of $\frac{1}{10+(2i)}$. The number of seeds per axis for MRRT is varied between 2 and 20 in steps of 2.

The same random sequence generator is considered for RRT for unbiased comparison. The following formula, developed in [115], is used to evaluate the maximum tree number (N) as a function of the number of seeds per axis (K):

$$N = 3(K^3) + 2. \tag{3}$$



(a) (b) (c)

Fig. 3. The different scenarios: (a) scenario 1 (b) scenario 2 and (c) scenario 3.

The minimum number of trees is 2, one at the start and another at the goal nodes. The multiplication and power factors of 3 are a result of the three-dimensional environment the path planning algorithms will operate in.

Each test for all algorithms is repeated for 100 times. Path length before and after smoothing and computational time are the performance measures that are considered. Owing to the generic environment, the subsequent results can be extrapolated to assess real applications.[a]

5. Results

5.1. *Introduction*

This section aims to identify the best configurations for the realization of 3D path planning. Therefore, this section will present, analyze, discuss and compare the path planning performance of the A* and RRT algorithms, their variants and the smoothing algorithm. This will help identify the strongholds and shortcomings of each configuration in 3D path planning in both generic and specific environments with associated mission constraints.

5.2. *A* algorithm*

The A* path planner constructs a path from start to goal points in all scenarios and resolutions. This may imply that using the A* algorithm a solution will always be guaranteed. But as remarked in the literature [38], this is not the case as a solution depends also upon the selected resolution. From Fig. 4(a), it can be deduced that path length prior smoothing is a function of scenario complexity. In this regard, the path lengths for Scenarios 2 and 3 are 50% and 100% longer with respect to Scenario 1, since the path planner must pass through three and five windows in opposite obstacle plane extremes instead of two windows, respectively, as can be deduced from Fig. 3.

The direct relationship between path length and scenario complexity is further confirmed in the path length after smoothing results illustrated in Fig. 4(b). The ripple effect in path length is reduced at smoothing stage since path segment length and grid position restrictions are eliminated. A comparison between the mean smoothed and unsmoothed path lengths shows a 12%, 16% and 17% decrease for Scenarios 1–3, respectively, with respect to the unsmoothed path lengths. Tests show that increasing the number of smoothing iterates beyond 1000 increases computational time proportionally without any significant improvement in path length. Moreover, through smoothing iterate increase, the number of path points increases consequently increasing path navigation complexity and ultimately reducing navigation performance.

From Fig. 4(c), it can be concluded that path planning time is a function of resolution and scenario difficulty, with resolution having the major dependence. Resolution is directly proportional with the number of grid points and therefore the more time is required by the A* path planner to generate a path. Similarly, scenario complexity increases path planning time as more options need to be considered for a noncolliding path to be computed. The path construction time is much larger than the path smoothing time and therefore the latter effect on computational time is negligible. Scenario complexity also affects path smoothing time, as can be deduced from Fig. 4(d).

[a] An Intel Xeon ES-1650, 3.2 GHz is used for testing.

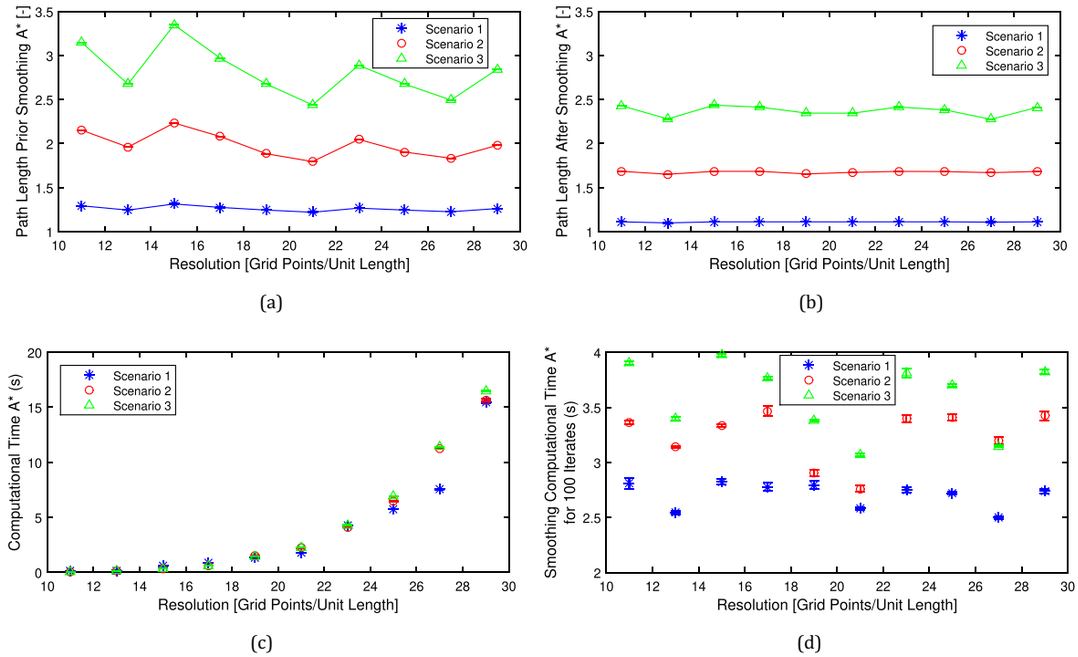


Fig. 4. Results for A* with 95% confidence interval (CI): (a) nonsmoothed path length, (b) smoothed path length, (c) planning time and (d) cumulative smoothing time for 100 iterations.

5.3. *A* ripple reduction algorithm* (A_R^*)

As illustrated in Fig. 4(a), the variance limits at 95% confidence interval for the unsmoothed path length is 0. This results since each test for the same resolution and scenario is repeated for 100 times using the same start and goal positions. Therefore, all 100 runs will yield the same path. Oppositely, with the introduction of A_R^*, each run is different leading to a nonzero 95% confidence interval, as illustrated in Fig. 5. An asymptotic relationship between path length and resolution can be deduced from Fig. 5(a). From a theoretical perspective, as the resolution increases, the distance between graph nodes reduces, increasing the number of graph points in the same environmental space and therefore more optimized paths can be constructed. This confirms the conclusion drawn from the tests that path length and resolution have an inversely proportional relationship.

An asymptotic behavior in confidence interval with increase in resolution is also exhibited for the smoothed path length (Fig. 5(b)) since with increase in resolution the path length variance in possible path options reduces. Similar to path generation time results of A*, the confidence interval for path generation time increases exponentially as for A*. In terms of confidence interval, the resolution has a minimal effect on smoothing time. The smoothing algorithm is not restricted by grid positions and therefore is independent of resolution.

A path length ripple reduction of 46%, 46% and 48% in terms of standard deviation for Scenarios 1–3, respectively, is introduced by the A_R^* algorithm with respect to the standard A* algorithm. This result is confirmed using a different random seed generator. This result confirms that path length ripple exhibited by the A* algorithm is the result of a plane residing exactly on a plane of grid positions with the only available grid points residing at the respective obstacle windows. Using the A_R^* algorithm that normalizes such iterations, a fairer A* performance responses results.

With the addition of the smoothing algorithm, the path length ripple is further reduced by 3 to 4 times for all scenarios but never reaches 0. Results show that the ripple reduction in amplitude due to the smoothing algorithm is higher for the standard A* with respect to A_R^*, since in A* the ripple is higher. This behavior shows that the smoothing algorithm's path length reduction reduces as the path reaches its optimal length.

A marginal mean increase of 1%, 2% and 2% for Scenarios 1–3, respectively, results in unsmoothed path length for A_R^* with respect to A* algorithms. This increase is a consequence of obstacle modeling that defines a buffer of half the distance between nodes. Without this buffer, planes not residing exactly on nodes will be neglected, but this

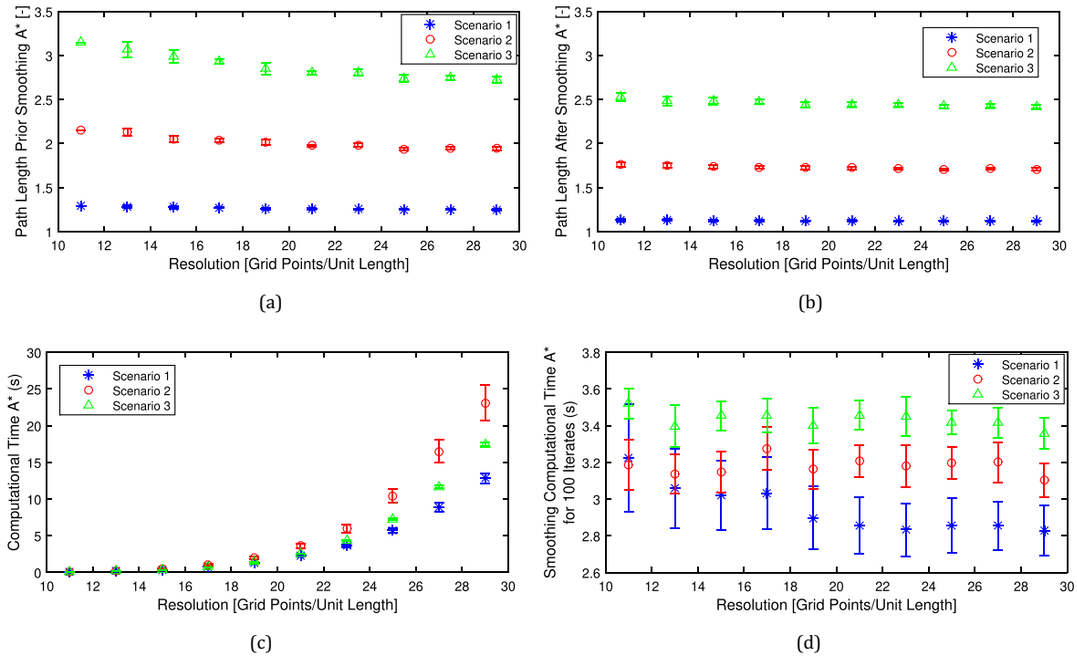


Fig. 5. Results for A* Ripple Reduction Algorithm (A_R^*) with 95% CI: (a) nonsmoothed path length, (b) smoothed path length, (c) planning time and (d) cumulative smoothing time for 100 iterations.

buffer can create situations where one obstacle plane is modeled by two obstacle planes.

The mean smoothed path length is 1%, 3% and 4% increase for Scenarios 1–3, respectively, for the A_R^* with respect to A*. This increase correlates with the path length increase in the unsmoothed path length of both A* and A_R^* algorithms. The path planning time is 5% shorter and 49% and 4% longer for the A_R^* algorithm with respect to the A* algorithm. The major increase in path planning time for Scenario 2 is a consequence of the obstacle grid point estimation as explained earlier in [112]. The path smoothing time is 9% longer, 2% shorter and 5% shorter for the A_R^* algorithm with respect to the A* algorithm. This marginal difference is a consequence of the environmental shifting and the associate possibility of different paths. Further details are provided in [112].

In conclusion, it can be stated that the A_R^* algorithm reduces path length ripple with respect to the A* algorithm. This reduction is not achieved at the expense of longer path length or computational time. These results are confirmed through tests carried on different complexity 3D scenarios.

5.4. *Rapidly-Exploring Random Trees* (*RRT*)

Figure 6 shows that the RRT algorithm finds a path solution in all scenarios and for all considered step sizes. This supports the notion that the RRT algorithm is probabilistically complete [32]. Figure 6(a) shows that the path length prior smoothing is mainly independent of the step size for all scenarios. The step size determines the tree branch length. Therefore the zig-zagging amplitude is directly proportional with the step size. The sampling-based nature of the RRT algorithm in constructing a path using standard length tree branches show the algorithm's lack of optimality emphasized in the literature [32]. The smoothing algorithm reduces the mean path length by 46%, 50% and 49% for scenarios 1–3, respectively.

Overall, the A* algorithm constructed shorter unsmoothed paths with respect to RRT for all scenarios. Although both A* and RRT algorithms are restricted in movement, A* can only pass through predefined grid positions while RRT can use any nonobstacle space, provided that this is distant by the step size constraint from the parent node. Therefore, path length results cannot be compared directly. Similarly for both algorithms, the path construction time exhibits an inverse relationship with resolution or step size since the number of path segments for equal volume increases with increase in resolution and decrease in step size. Although the smoothing time for both algorithms consumes only a small fraction of the total path generation time with resolution and step size constraints neglected, scenario complexity also increases smoothing time.

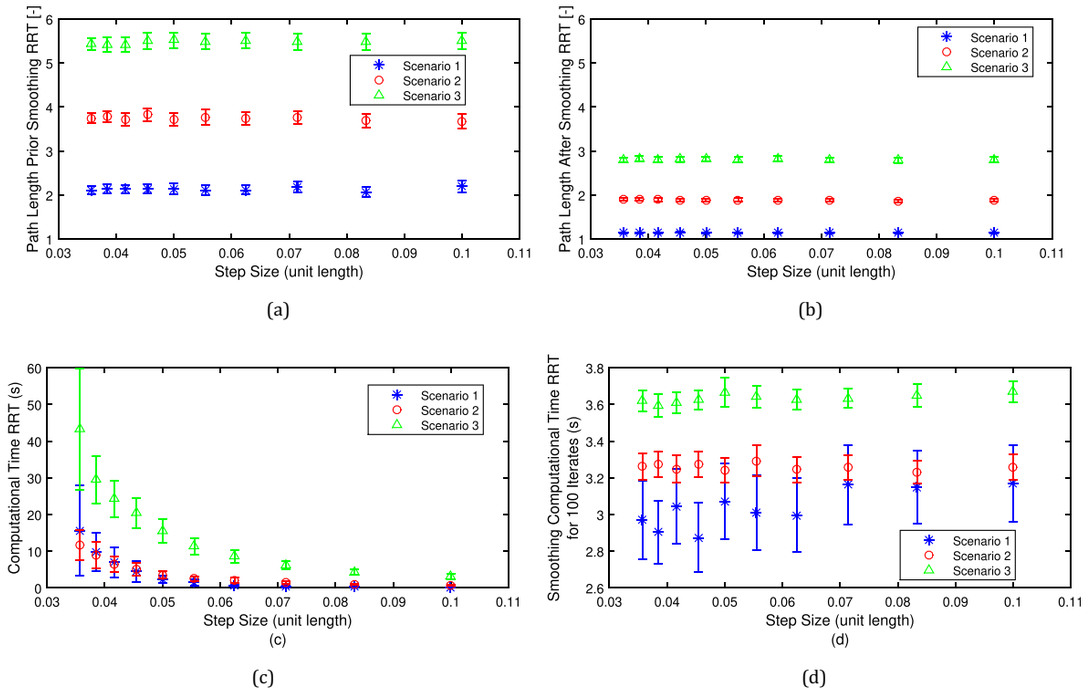


Fig. 6. Results for RRT with 95% CI: (a) nonsmoothed path length, (b) smoothed path length, (c) planning time and (d) cumulative smoothing time for 100 iterations.

5.5. *Rapidly-exploring random trees without step size*

Table 1 tabulates the mean results for all three scenarios for 100 iterations with no limits on tree branch length. The unsmoothed path length is 17%, 11% and 4% longer for Scenarios 1–3, respectively, for the unconstrained RRT variant with respect to the standard RRT. The difference reduced significantly to 1%, 0.4% and $< 0.1\%$, for Scenarios 1–3, respectively, when the smoothed path lengths of the variant and standard RRT are compared, with the latter remaining the shorter. Therefore, it can be concluded that path length is deteriorated by removing constraints, with the major deterioration resulting in low-obstacle density scenarios as long zig-zagging path segments are constructed which would have been attenuated if tree branch lengths are constrained.

Table 1. Path planning performance for RRT with no step size constraint.

Parameter	Scenario 1	Scenario 2	Scenario 3
Nonsmoothed path length [—]	1.83	3.38	5.27
Smoothed path length [—]	1.13	1.88	2.81
Path planning time (s)	0.01	0.67	1.64
Average smoothing time per iterate (ms)	35.97	46.57	51.69

Path planning time for this RRT variant is reduced by an average 450, 7 and 10 with respect to the standard RRT for Scenarios 1–3, respectively. This result shows that the tree branch length constraint limits the propagation to the goal position especially in simple scenarios where only a small amount of manoeuvres are acquired to reach the goal. An increase of 18%, 43% and 42% in smoothing time is exhibited for the RRT without step size constraint with respect to the standard RRT. A slight increase in smoothing time is exhibited for the unconstraint RRT in comparison with the constraint RRT mainly due to the zig-zagging in the low obstacle density paths. Path planning time for A* is also larger than for this RRT variant since the latter is not limited by grid positions. The difference is more emphasized at low resolutions in simple scenarios. In addition, this RRT variant's smoothing time is less than half of A* for the same scenarios.

5.6. *Multiple Rapidly-Exploring Random Trees* (*MRRT*)

Figure 7 shows that a noncolliding path from start to goal is constructed for all scenarios. Results show that scenario difficulty is the main factor affecting path length while this

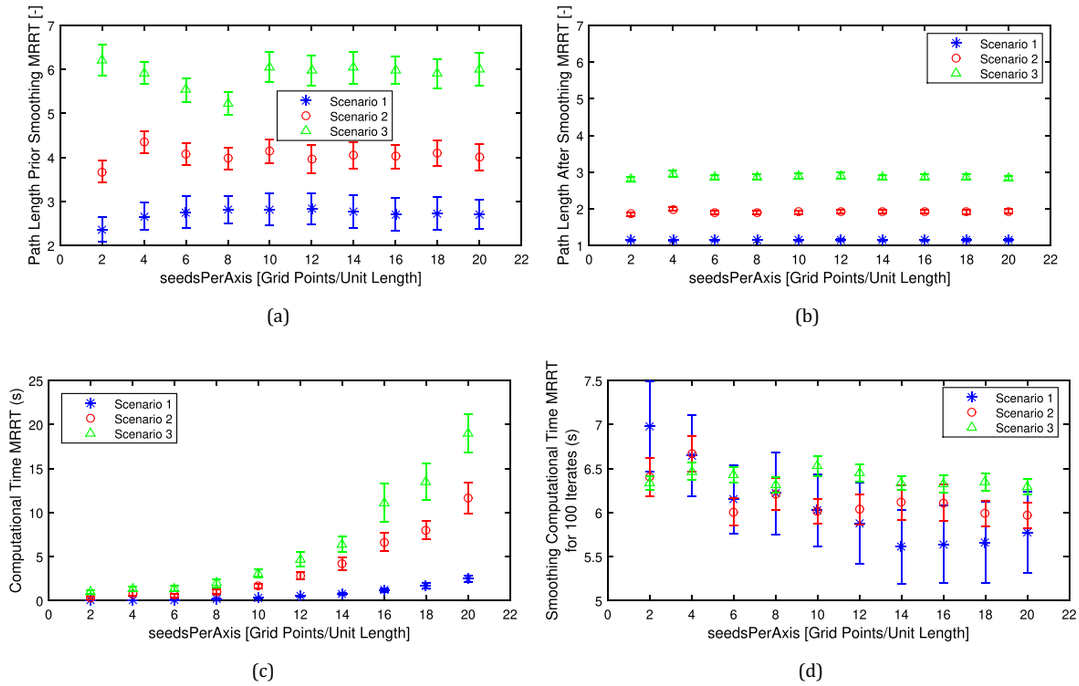


Fig. 7. Results for MRRT with 95% CI: (a) nonsmoothed path length, (b) smoothed path length, (c) planning time and (d) cumulative smoothing time for 100 iterations.

parameter is not affected by the number of seeds per axis. MRRT constructs longer unsmoothed paths when compared with the unconstrained RRT for all scenarios. In MRRT, trees can be interconnected with no length restrictions. These lack-of-length restrictions result in zig-zagging paths prior smoothing, increasing in amplitude at low complexity scenarios, similar to the unconstrained RRT results.

The smoothed path length for MRRT remains longer than that for RRT for each respective scenario, although the difference in mean path length between the two methods is reduced through the smoothing algorithm. For A*, the mean unsmoothed path length is less than half with respect to MRRT for all scenarios. As in other algorithms discussed in this study, the path planning time for MRRT depends on scenario difficulty. Furthermore, the planning time is directly proportional with seeds-per-axis value.

Overall, the MRRT algorithm constructed a path in less time than both A* and RRT with the major difference exhibited in low-complexity scenarios. This results since the MRRT tree propagation and interconnection is unrestricted as opposed to both A* and RRT which are restricted in path points and tree branch length, respectively. The unconstrained RRT constructed the path in multiple less times as opposed to MRRT. To construct a path, the MRRT requires more nodes than both RRTs, resulting in a longer smoothing time. In comparison with A*, the MRRT required less smoothing time owing that A* constructs almost optimal paths and therefore the smoothing algorithm requires more time to find path segments which can improve the path.

5.7. *New smoothing algorithm*

The new smoothing algorithm is computationally more efficient than the older smoothing algorithm version. For the A* algorithm, results show that the lower the scenario complexity, the more smoothing iterations are required since the stopping condition ($<1\%$ reduction in last 20 iterations) is reached. This results since in low-obstacle density scenarios, the environmental space is primarily obstacle free as opposed to high obstacle density scenarios where path improvement is more restricted. Also, grid point restrictions are neglected by the smoothing algorithms, therefore resolution constrained are eliminated. The new smoothing algorithm is efficient resulting in 46%, 50% and 49% for Scenarios 1–3, respectively. For complex situations, such as Scenario 3, the minimum of 20 iterations is too low as sometimes, owing to scenario complexity, it is too difficult to find a set of points in 20 iterations on the path segments to construct a shorter overall path. For more results and further analysis please refer to [112].

The new smoothing algorithm requires more time for RRT with respect to A* for all step sizes/resolutions and scenarios under review. Step size constrains minimally affect the number of smoothing iterations since the stopping condition is reached. This results for the same reason explained for A*. Smoothing results for RRT in simple scenarios show a reduction of between 13–19 times with a minor increase in smoothed path length. Results show that in complex scenarios the minimum of 20 smoothing iterations is too low and shall be increased as it difficult to find shorter noncolliding path segments, explained for A*. The smoothed paths resulting from the A* algorithm outperforms those from the RRT algorithm in terms of both length and smoothed time since the unsmoothed path generated by the A* is better than that for RRT. The elimination of the tree branch length in RRT has no effect on smoothed path length and time.

For MRRT, scenario complexity increases the number of smoothing iterates since the stopping condition is reached. This is attributed to the lack of path optimality constructed by the RRT-based algorithms. MRRT's seeds per axis parameter has no effect on the smoothing algorithm's stopping condition since it neglects nodal and path segment length constraints. Through the new smoothing algorithm, smoothing time has decreased by 22, 17 and 24 times for Scenarios 1–3, respectively, with respect to the original smoothing algorithm.

In conclusion, results show that a path length reduction of 20–50% can be achieved at a small percentage of the path construction time. Furthermore, this analysis highlights the direct relationship between smoothing iterations and path length. This can guide in custom tuning the amount of path length reduction based on the application requirements and computational power availability.

6. Conclusion

This work analyses the path planning performance of the A* and RRT based algorithms with an associated smoothing algorithm in different complexity 3D environments. In line with the literature, the A* algorithm constructed more optimal paths than RRT. The A* algorithm searches volumes in the line of sight of the goal while the RRT algorithms searches evenly throughout the environment. The paths constructed by the A* algorithm are shorter, even after smoothing, and constructed in less time than RRT. The A* algorithm exhibits path length ripples as resolution is varied for the same scenario. Through A* ripple reduction algorithm, a 46–48% path ripple reduction is recorded for all situations considered with respect to the A* algorithm. This improvement is achieved without increasing path length and planning time. The unconstrained RRT variant reduced path planning time especially in low obstacle density scenarios. Moreover, the evenly-distributed MRRT generated longer unsmoothed paths in shorter planning times but required more smoothing over RRT for all considered scenarios. The new smoothing algorithm, developed to eliminate unnecessary smoothing iterates, shows a 90% reduction in smoothing time for all algorithms. This smoothing time reduction is achieved with a path length increase of less than 10% for A* and between 25% and 45% for all RRT algorithms.

In addition to path length optimality and noncolliding paths, the path planning time is also an important factor especially in time-varying environments. In such situations, the path must be continuously checked, optimized and if a potential collision is possible, re-planned using only available and limited onboard resources. This work highlights A*'s potential as an online 3D path planning algorithm in dynamic environments owing to its optimality and low computational demand. A*'s environmental discretization nature allows it to be tuned based on mission requirements and agent onboard resources. RRT-based methods can construct efficient paths in both evenly and focused exploration situations. Their performance in these situations can outperform A*'s, in terms of path length and time, if tree propagation nodes are empirically selected based on obstacles' current and prospective future states. Finally, the new smoothing algorithm further improves path construction in all algorithms by taking $< 1\%$ of the total path planning time.

Furthermore, the inclusion of both kinematic and dynamic agent models, sensor limitations, time-invariant and time-varying obstacle positions, speed and orientation in addition to uncertainty in agent, sensor and environment will further assess the A*-based and RRT-based algorithms' applicability in static and dynamic 3D environments for different vehicular systems including UAVs and AUVs.

References

[1] C. Zammit and E. J. van Kampen, Comparison between A* and RRT algorithms for UAV path planning, in *Proc. AIAA Guidance, Navigation and Control Conf. AIAA SciTech Forum*, Kissimmee, FL, 2018, pp. 1–23.

[2] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How and G. Fiore, Real-time motion planning with applications to autonomous urban driving, *IEEE Trans. Control Syst. Technol.* **17**(5) (2009) 1105–1118.

[3] A. Nash and S. Koenig, Any-angle path planning, *Artif. Intell. Mag.* **34** (4) (2013) 9–31.

[4] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, Anytime search in dynamic graph, *Artif. Intell.* **172**(14) (2008) 1613–1643.

[5] P. B. Sujit and D. Ghose, Search by UAVs with flight time constraints using game theoretical models, in *AIAA Guidance, Navigation and Control Conf.*, San Francisco, CA, 2005, pp. 1–11.

[6] J.-W. Lee, B. Walker and K. Cohen, Path planning of unmanned aerial vehicles in a dynamic environment, in *Infotech@Aerospace*, St. Louis, Missuori, 2011, pp. 1–19.

[7] L. Blackmore, Robust path planning and feedback design under stochastic uncertainty, in *AIAA Guidance, Navigation and Control Conf. Exhibit*, Honolulu, HI, 2008, pp. 1–12.

[8] A. R. Babaei and M. Mortazavi, Fast trajectory planning based on inflight way points for unmanned aerial vehicles, *Aircr. Eng. Aerosp. Technol.* **82**(2) (2010) 107–115.

[9] M. D. Zollar, R. G. Cobb and D. J. Grymin, Optimal SUAS path planning in three-dimensional constrained environments, *Unmanned Syst.* **7 (2)** (2019) 105–118.

[10] A. Chakrabarty and J. W. Langelaan, Energy maps for long-range path planning for small-and micro-UAVs, in *AIAA Guidance, Navigation and Control Conf.*, Chicago, IL, 2009, pp. 1–13.

[11] M. Gros, A. Schöttl and W. Fichter, Spline and OBB-based path-planning for small UAVs with the finite receding-horizon incremental-sampling tree algorithm, in *AIAA Guidance, Navigation and Control Conf.*, Boston, MA, 2013, pp. 1–17.

[12] J. N. Amin, J. D. Boskovic and R. K. Mehra, A fast and efficient approach to path planning for unmanned vehicles, in *AIAA Guidance, Navigation and Control Conf.*, Keystone, CO, 2006, pp. 1–9.

[13] Y. Huang, J. Chen, H. Wang and G. Su, A method of 3D path planning for solar-powered UAV with fixed target and solar tracking, *J. Aerosp. Sci. Technol.* **92** (2019) 831–838.

[14] J. L. Foo, J. Knutzon, V. Kalivarapu, J. Oliver and E. Winer, Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization, *J. Aerosp. Comput. Inf. Commun.* **6**(4) (2009) 271–290.

[15] R. Dai and J. Cochran, Path planning and state estimation for unmanned aerial vehicles in hostile environments, *J. Guid. Control Dyn.* **33**(2) (2010) 595–601.

[16] Y. Dong, Y. Zhang and J. Ai, Experimental test of unmanned ground vehicle delivering goods using RRT path planning algorithm, *Unmanned Syst.* **5**(1) (2017) 45–57.

[17] K. P. Bollino and L. R. Lewis, Collision-free multi-UAV optimal path planning and cooperative control for tactical applications, in *AIAA Guidance, Navigation and Control Conf.*, Honolulu, HI, 2008, pp. 1–18.

[18] X. Sun, C. Cai and X. Shen, A new cloud model based human-machine cooperative path planning method, *J. Intell. Robot. Syst.* **79** (2014) 3–19.

[19] E. L. D. Angelis, F. Giulietti, G. Pipeleers, G. Rossetti and R. Van Parys, Optimal autonomous multirotor motion planning in an obstructed environment, *J. Aerosp. Sci. Technol.* **87** (2019) 379–388.

[20] Z. Zhen, Y. Chen, L. Wen and B. Han, An intelligent cooperative mission planning scheme of UAV swarm in uncertain dynamic environment, *J. Aerosp. Sci. Technol.* **100** (2020) 1–16.

[21] C. Hu, Z. Zhang, N. Yang, Y.-S. Shin and A. Tsourdos, Fuzzy multiobjective cooperative surveillance of multiple UAVs based on distributed predictive control for unknown ground moving target in urban environment, *J. Aerosp. Sci. Technol.* **84** (2019) 329–338.

[22] S. Park, H.-L. Choi, N. Roy and J. How, Learning covariance dynamics for path planning of UAV sensors in a large-scale dynamic environment, in *AIAA Guidance, Navigation and Control Conf.*, Chicago, IL, 2009, pp. 1–18.

[23] C. Crispin and A. Sobester, An intelligent, heuristic path planner for multiple agent unmanned air systems, in *AIAA Information Technology at Aerospace*, Kissemmee, FL, 2015, pp. 1–13.

[24] J. D. Boskovic, N. Knoebel, N. Moshtagh and G. L. Larson, Collaborative mission planning & autonomous control technology (CoMPACT) system employing swarms of UAVs, in *AIAA Guidance, Navigation and Control Conf.*, Chicago, IL, 2009, pp. 1–24.

[25] P. P. Wu, D. Campbell and T. Merz, Multi-objective four-dimensional vehicle motion planning in large dynamic environments, *IEEE Trans. Syst. Man Cybern. B Cybern.* **41**(3) (2011) 621–634.

[26] S. Luke, K. Sullivan, L. Panait and G. C. Balan, Tunably decentralized algorithms for cooperative target observation, in *4th Int. Joint Conf. Autonomous Agents and Multiagent Systems*, Utrecht, The Netherlands, 2005, pp. 1–24.

[27] P. B. Sujit and D. Ghose, Two-agent cooperative search using game models with endurance-time constraints, *Eng. Optim.* **42**(7) (2010) 617–639.

[28] C. Sabo, D. Kingston and K. Cohen, A formulation and heuristic approach to task allocation and routing of UAVs under limited communication, *Unmanned Syst.* **2**(1) (2014) 1–17.

[29] L. Babel, Flight path optimization with application to in-flight replanning to changing destinations, *Aircr. Eng. Aerosp. Technol.* **90** (8) (2018) 1192–1202.

[30] A. Ryan and J. K. Hedrick, A mode-switching path planner for UAV-assisted search and rescue, in *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, 2005, pp. 1471–1476.

[31] D. González, J. Pérez, V. Milanés and F. Nashashibi, A review of motion planning techniques for automated vehicles, *IEEE Trans. Intell. Transp. Syst.* **17**(4) (2016) 1135–1145.

[32] S. Ghandi and E. Masehian, Review and taxonomies of assembly and disassembly path planning problems and approaches, *CAD Comput. Aided Des.* **67–68** (2015) 58–86.

[33] A. Short, Z. Pan, Z. Larkin and S. van Duin, Recent progress on sampling based dynamic motion planning algorithms, in *IEEE Int. Conf. Advanced Intelligent Mechatronics (AIM)*, Alberta, Canada, 2016, pp. 1305–1311.

[34] P. E. Hart, N. J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4**(3) (1968) 100–107.

[35] F. H. Tseng, T. T. Liang, C. H. Lee, L. D. Chou and H. Chao, A star search algorithm for civil UAV path planning with 3G communication, in *10th Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, Kitakyushu, Japan, 2014, pp. 942–945.

[36] S. M. Lavalle and J. J. Kuffner, Randomized kinodynamic planning, *Int. J. Robot. Res.* **20**(3) (2001) 378–400.

[37] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.* **1** (1959) 269–271.

[38] M. B. Sathyaraj, L. C. Jain, A. Finn and S. Drake, A multiple UAVs path planning algorithms: A comparative study, *Fuzzy Optim. Decis. Mak.* **7**(3) (2008) 257–267.

[39] M. Lan, S. Lai, T. H. Lee and B. M. Chen, A survey of motion and task planning techniques for unmanned multicopter systems, *Unmanned Syst.* **9**(2) (2021) 165–198.

[40] J. Mitchell and C. Papadimitriou, The weighted region problem: Finding shortest paths through a weighted planar subdivision, *J. Assoc. Comput. Mach.* **38**(1) (1991) 18–73.

[41] S.-L. Guo, J. Duan, Y. Zhu, X.-C. Li and T.-W. Chen, Improved Dijkstra algorithm based on fibonacci heap for solving the shortest path problem with specified nodes, in *Proc. Int. Conf. Computer Science and Artificial Intelligence*, ed. W.-J. Chang (Guilin, China, 2017), pp. 52–61.

[42] Q. Li, Z. Zeng, B. Yang and T. Zhang, Hierarchical route planning based on taxi GPS-trajectories, in *17th Int. Conf. Geoinformatics*, Fairfax, VA, 2009, pp. 1–5.

[43] R. Kala and K. Warwick, Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization, *J. Intell. Robot. Syst.* **72**(3–4) (2013) 559–590.

[44] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer and B. Satterfield, Little Ben: The Ben Franklin racing team' s entry in the 2007 darpa urban challenge, *J. Field Robot.* **25**(9) (2008) 598–614.

[45] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Rein-holtz, D. Hong, A. Wicks, T. Alberi and D. Anderson, Odin: Team victortango' s entry in the darpa urban challenge, *J. Field Robot.* **25**(8) (2008) 467–492.

[46] J. Gibson, T. Schuler, L. McGuire, D. M. Lofaro and D. Sofge, Swarm and multi-agent time-based A* path planning for lighter-than–air systems, *Unmanned Syst.* **8**(3) (2020) 253–260.

[47] A. Stentz, Optimal and efficient path planning for unknown and dynamic environments, Carnegie Mellon Robotics Institute Technical Report, CMU-RI-TR-93-20. (1993).

[48] A. Stentz, The focused D* algorithm for real-time replanning, in *Proc. Int. Joint Conf. Artificial Intelligence*, Quebec, Canada, 1995, pp. 1652–1659.

[49] S. Koenig and M. Likhachev, Incremental A*, in *Proc. Neural Information Processing Systems*, Vancouver, Canada, 2001, pp. 1–8.

[50] S. Koenig and M. Likhachev, D* lite, in *Proc. AIAA Conf. Artificial Intelligence*, Indianapolis, IN, 2002, pp. 476–483.

[51] D. Ferguson and A. Stentz, Using interpolation to improve path planning: The field D* algorithm, *J. Field Robot.* **23**(2) (2006) 79–101.

[52] A. Niewola and A. Podsdkowski, L* Algorithm — A linear computational complexity graph searching algorithm for path planning, *J. Intell. Robot. Syst.* **91** (2017) 425–444.

[53] K. Daniel, A. Nash, S. Koenig and A. Felner, Theta*: Any-angle path planning on grids, *J. Artif. Intell.* **39** (2010) 533–579.

[54] A. Nash, S. Koenig and M. Likhachev, Incremental Phi*: Incremental any-angle path planning on grids, in *Proc. 21st Int. Joint Conf. Artificial Intelligence (IJCAI)*, Pasadena, CA, 2009, pp. 1824–1830.

[55] K. Trovato and L. Dorst, Differential A*, *IEEE Trans. Knowl. Data Eng.* **14**(6) (2002) 1218–1229.

[56] A. Nash, S. Koenig and C. Tovey, Lazy Theta*: Any-angle path planning and path length analysis in 3D, in *Proc. 3rd Annual Symp. Combinatorial Search Intelligence (SOCS)*, Atlanta, GA, 2010, pp. 153–154.

[57] K. Trovato, R. Marn, M. Popovi, I. Maza and A. Viguria, Efficient lazy Theta* Path planning over a sparse grid to explore large 3D volumes with a multirotor UAV, *Sensors* **19**(1) (2019) 174.

[58] P. Yap, N. Burch, R. Holte and J. Schaeffer, Block A*: Database-driven search with applications in any-angle path-planning, in *Proc. Conf. Artificial Intelligence (AAAI)*, San Francisco, CA, 2011, pp. 120–125.

[59] D. Harabor and A. Grastien, Online graph pruning for path finding on grid maps, in *Proc. Conf. Artificial Intelligence (AAAI)*, San Francisco, CA, 2011, pp. 114–119.

[60] F. Duchoň, A. Babineca, M. Kajana, P. Beňo, M. Floreka, T. Ficoa and L. Jurišica, Path planning with modified A star algorithm for a mobile robot František, in *Modelling of Mechanical and Mechatronic Systems (MMaMS)*, Procedia Engineering Vol. 96, 2014, pp. 59–69.

[61] X. Sun, W. Yeoh and S. Koenig, Dynamic fringe-saving A*, in *Proc. 8th Int. Conf. Autonomous Agents and Multiagent systems (AAMAS)*, Budapest, Hungary, 2009, pp. 891–898.

[62] K. Gochev, A. Safonova and M. Likhachev, Anytime tree-restoring A* graph search, in *Proc. 7th Annual Symp. Combinatorial Search SoCS*, Prague, Czech Republic, 2014, pp. 80–88.

[63] X. Sun, S. Koenig and W. Yeoh, Generalized adaptive A*, in *Proc. 7th Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, Estoril, Portugal, 2008, pp. 469–476.

[64] C. Hernández, X. Sun, S. Koenig and P. Meseguer, Tree adaptive A*, in *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, Taipei, Taiwan, 2011, pp. 123–130.

[65] S. Ainea and M. Likhachevb, Truncated incremental search, *Artif. Intell.* **234** (2016) 49–77.

[66] L. E. Kavraki, P. Švestka, J. C. Latombe and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robot. Autom.* **12**(14) (1996) 566–580.

[67] S. M. LaValle, Rapidly-exploring random trees: a new tool for path planning, Iowa State University Technical Report, 98–11 (1998).

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

[69] R. Bohlin and L. E. Kavraki, Path planning using lazy PRM, in *Proc. IEEE Int. Conf. Robotics and Automation*, Vol. 1 (2000), pp. 521–528.

[70] K. P. Leven and S. Hutchinson, Toward real-time path planning in changing environments, in *Algorithmic and Computational Robotics: New Directions: 4th Int. Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, 2000, pp. 363–376.

[71] K. Klasing, D. Wollherr and M. Buss, Cell-based probabilistic roadmaps (CPRM) for efficient path planning in large environments, in *Proc. Int. Conf. Advanced Robotics*, Daegu, South Korea, 2007.

[72] M. Pomarlan and I. A. Sucan, Motion planning for manipulators in dynamically changing environments using real-time mapping of free workspace, in *IEEE 14th Int. Symp. Computational Intelligence and Informatics (CINTI)*, Budapest, Hungary, 2013, pp. 483–487.

[73] A. Gayle, M. Sud, C. Lin and D. Manocha, Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments, *Int. Conf. Intelligent Robots and Systems (IROS)*, San Diego, CA, 2007, pp. 3777–3783.

[74] K. Belghith, F. Kabanza, L. Hartman and R. Nkambou, Any-time dynamic path-planning with flexible probabilistic roadmaps, *IEEE Int. Conf. Robotics and Automation (ICRA)*, Orlando, FL, 2013, pp. 2372–2377.

[75] B. Y. Yang and O. Brock, Elastic roadmaps motion generation for autonomous mobile manipulation, *Auton. Robots* **28**(1) (2010) 113–130.

[76] A. Dias, T. Fernandes, J. Almeida, A. Martins and E. Silva, 3D path planning methods for unmanned aerial vehicles in search and rescue scenarios, in *Proc. 20th Int. Conf. CLAWAR*, Porto, Portugal, 2017, pp. 213–220.

[77] S. M. LaValle and J. J. Kuffner, Randomized kinodynamic planning, in *Proc. IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999, pp. 473–479.

[78] D. Devaurs, T. Siméon, and J. Cortés, Optimal path planning in complex cost spaces with sampling-based algorithms, *IEEE Trans. Autom. Sci. Eng.* **13**(2) (2015) 415–424.

[79] R. Geraerts and M. Overmars, Creating high-quality paths for motion planning, *Int. J. Robot. Res.* **26**(8) (2007) 845–863.

[80] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos and L. Petrou, A review of global path planning methods for occupancy grid maps regardless of obstacle density, *J. Intell. Robot. Syst.* **84** (2016) 829–858.

[81] S. Karaman and E. Frazzoli, Optimal kinodynamic motion planning using incremental sampling-based methods, in *Proc. 49th IEEE Conf. Decision and Control*, Atlanta, GA, 2010, pp. 7681–7687.

[82] D. Devaurs, T. Siméon and J. Cortés, Enhancing the transition-based RRT to deal with complex cost spaces, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Orlando, FL, 2013, pp. 4561–4568.

[83] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan and M. S. Muhammad, DRRT*-SMART: A rapid convergence implementation of RRT*, *J. Adv. Robot. Syst.* **10**(7) (2013) 1–12.

[84] Z. Zhang, R. Du and R. V. Cowlagi, Randomized sampling-based trajectory optimization for UAVs to satisfy linear temporal logic specifications, *J. Aerosp. Sci. Technol.* **96** (2020) 1–9.

[85] J. J. Kuffner and S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 521–528.

[86] D. Zhang, Y. Xu and X. Yao, An improved path planning algorithm for unmanned aerial vehicle based on RRT-connect, in *Proc. 37th Chinese Control Conf. (CCC)*, Wuhan, China, 2018, pp. 4854–4858.

[87] P. Pharpatara, B. Herisse and Y. Bestaoui, 3D trajectory planning of aerial vehicles using RRT*, *IEEE Trans. Control Syst. Technol.* **25**(3) (2017) 1116–1123.
[88] J. Bruce and M. Veloso, Real-time randomized path planning for robot navigation, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Lausanne, Switzerland, 2002, pp. 2383–2388.
[89] S. R. Martin, S. E. Wright and J. W. Sheppard, Offline and online evolutionary bi-directional RRT algorithms for efficient re-planning in dynamic environments, in *Proc. IEEE Int. Conf. Automation Science and Engineering*, Scottsdale, AZ, 2007, pp. 1131–1136.
[90] E. Frazzoli, M. A. Dahleh and E. Feron, Real-time motion planning for agile autonomous vehicles, *J. Guid. Control Dyn.* **25**(1) (2002) 116–129.
[91] J. Guitton, J. L. Farges and R. Chatila, Cell-RRT: Decomposing the environment for better plan, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, St. Louis, MO, pp. 5776–5781.
[92] D. Braid, A. Broggi and G. Schmiedel, The TerraMax autonomous vehicle, *J. Field Robot.* **23**(5) (2016) 693–708.
[93] R.-H. Ryu, D. Ogay, D. Bulavintsev, H. Kim and J.-S. Park, Development and experiences of an autonomous vehicle for high-speed navigation and obstacle avoidance, *Front. Intell. Auton. Syst.* **466** (2013) 105–116.
[94] W. Shang, J. Liu, R. Ning and M. Liu, Computational path planner for product assembly in complex environments, *Chin. J. Mech. Eng.* **26** (2) (2013) 282–292.
[95] D. Ferguson and A. Stentz, Anytime RRTs, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Beijing, China, 2006, pp. 5369–5375.
[96] Q. Zhu, Y. Wu, G. Wu and X. Wang, An improved anytime RRTs algorithm, in *Int. Conf. Artificial Intelligence and Computational Intelligence*, Shanghai, China, 2009, pp. 268–272.
[97] Y. Abbasi-Yadkori, J. Modayil and C. Szepesvári, Extending rapidly-exploring random trees for asymptotically optimal anytime motion planning, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Shanghai, China, 2010, pp. 127–132.
[98] R. Alterovitz, S. Patil and A. Derbakova, Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 3706–3712.
[99] R. Luna, I. Şucan, M. Moll and L. Kavraki, Anytime solution optimization for sampling-based motion planning, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Orlando, FL, 2013, pp. 5068–5074.
[100] D. Ferguson, N. Karla and A. Stentz, Replanning with RRT, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Orlando, FL, 2006, pp. 1243–1248.
[101] D. Ferguson and A. Stentz, Anytime, dynamic planning in high-dimensional search spaces, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, 2007, pp. 1310–1315.
[102] M. Zucker, J. Kuffner and M. Branicky, Multipartite RRTs for rapid replanning in dynamic environments, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, 2007, pp. 1603–1609.
[103] M. Otte and E. Frazzoli, RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning, *Int. J. Robot. Res.* **29**(7) (2015) 797–822.
[104] R. Benenson, S. Petti, T. Fraichard and M. Parent, Integrating perception and planning for autonomous navigation of urban vehicles, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Beijing, China, 2006, pp. 98–104.
[105] K. E. Bekris and L. E. Kavraki, Greedy but safe replanning under kinodynamic constraints, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, 2007, pp. 704–710.
[106] E. Taheri, M. H. Ferdowsi and M. Danesh, Fuzzy Greedy RRT path planning algorithm in a complex configuration space, *Int. J. Control Autom. Syst.* **16**(6) (2018) 3026–3035.
[107] T. Y. Li and Y. C. Shie, An incremental learning approach to motion planning with roadmap management, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Washington, DC, 2002, pp. 3411–3416.
[108] R. Gayle, K. R. Klingler and P. G. Xavier, Lazy reconfiguration forest (LRF)-an approach for motion planning with multiple tasks in dynamic environments, in *IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, 2007, pp. 1316–1323.
[109] Y. Tsai, C. Lee, C. Lin and C. Huang, Development of flight path planning for multirotor aerial vehicles, *Aerospace* **2** (2015) 171–188.
[110] D. Roa and S. Williams, Large-scale path planning for underwater gliders in ocean currents, in *Australasian Conf. Robotics and Automation (ACRA)*, Sydney, Australia, 2009, pp. 1–9.
[111] R. Simmons and C. Urmson, Approaches for heuristically biasing RRT growth, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, NV, 2003, pp. 1178–1183.
[112] C. Zammit and E. J. van Kampen, Advancements for A* and RRT in 3D path planning of UAVs, in *AIAA Guidance, Navigation and Control Conf.*, San Diego, CA, 2019, pp. 1–17.
[113] S. Lee, A. Cho and C. Kee, Integrated waypoint path generation and following of an unmanned aerial vehicle, *Aircr. Eng. Aerosp. Technol.* **28**(5) (2010) 296–304.
[114] M. Clifton, G. Paul, N. Kwok, D. Liu and D. Wang, Evaluating performance of multiple RRTs, in *IEEE Conf. Mechatronic and Embedded Systems and Application*, Bejing, China, 2009, pp. 564–569.
[115] G. Paul, Multiple rapidly-exploring random tree (RRT), MathWorks, [Online Database]. https://www.mathworks.com/matlabcentral/fileexchan ge/21443-multiple-rapidly-exploring-random-tree-rrt-?requestedDomain=www.mathworks.com.

Christian Zammit received his B. Eng. and M. Sc. degrees from the University of Malta, in 2011 and 2014, respectively. From 2011–2014, he was Researcher at the University of Malta, and worked on on various EU FP7 Projects. Now, he holds the position of a Senior Lecturer at Malta College of Arts Science and Technology (MCAST) and is a Ph.D. candidate at the Technical University of Delft.

Christian Zammit is the author of nine technical publications. His research interests include: path planning, autonomous systems, unmanned aircraft systems, uncertainty modeling and guidance systems.

Erik-Jan van Kampen received his B.Sc and M.Sc degrees in Aerospace Engineering from the Delft University of Technology in The Netherlands in 2004 and 2006, respectively. In 2010, he received his Ph.D. from the Delft University of Technology on the topic of interval optimization. Since then, he has been Assistant Professor at the group of Control and Simulation in Delft. Erik-Jan van Kampen is the author of over 140 technical publications. His research interests include fault tolerant flight control and reinforcement learning.