Comparison between A* and RRT Algorithms for UAV Path Planning

C. Zammit[*] and E. van Kampen[†]

Delft University of Technology, Delft, 2629HS, The Netherlands

Unmanned Aerial Vehicles (UAVs) are being integrated into a wide range of indoor and outdoor applications. In this light, robust and efficient path planning is paramount. An extensive literature review showed that the A* and Rapidly–Exploring Random Tree (RRT) algorithms and their variants are the most promising path planning algorithms candidates for 3D UAV scenarios. These two algorithms are tested in different complexity 3D scenarios consisting of a box and a combination of vertical and horizontal plane obstacles with apertures. The path length and generation time are considered as the performance measures. The A* with a spectrum of resolutions, the standard RRT with different step–size constraints, RRT without step size constraints and the Multiple RRT (MRRT) with various seeds are implemented and their performance measures compared. Results confirm that all algorithms are able to generate a path in all scenarios for all resolutions, step sizes and seeds considered, respectively. Overall A*'s path length is more optimal and generation time is shorter than RRT projecting A* as a better candidate for online 3D path planning of UAVs.

I. Introduction

Unmanned Aerial Vehicles (UAVs) are slowly being integrated into all spheres of life. UAVs are being proposed for a wide range of indoor and outdoor applications varying from surveillance, search and rescue to leisure flying. The inclusion of UAVs into the national airspace and military applications requires robust autonomous guidance, navigation control (GNC) systems that shall ensure safe navigation in view of uncertainties present within UAV systems and the environment in which they will operate.

Autonomous GNC systems rely on robust and effective path planning. Path planning is the process of automatically generating feasible and optimal paths to a predefined goal point in view of static and dynamic environmental and model constraints and uncertainties. Path planning was initially approximated as a 2D problem[1–3] but with the introduction of complex 3D environmental and model constraints focus shifted onto 3D path planning approaches.[4–6] Besides fixed[7,8] and moving obstacles,[9,10] different studies consider a wide spectrum of constraints including: weather,[11,12] no–fly zones,[13,14] communication,[15,16] fuel[1,14] and model constraints.[13,17]

Although certain constraints such as buildings, no–fly zones and particular kinematic constraints can be defined accurately, in reality the absolute majority can only be defined with an element of uncertainty, for example weather, fuel consumption and vehicles' position and state. Furthermore, constraints can pop–up whilst in flight and the path planning algorithm shall be able to mitigate these *a priori* unknown situations.[13]

Different algorithms exist for path planning of automated vehicles. These path planning algorithms can be segmented into three main categories: Graph–based, Sampling–based and Interpolation approaches.

Graph–based searching algorithms define the state space into an occupancy grid defining obstacles as inaccessible grid points. Then algorithms search the available grid points giving a solution if a path exist.[27] Graph–based algorithms guarantee a solution if the grid resolution is adequate.[29]

Sampling–based searching algorithms select a non–structural finite number of points in the configuration space and create connections between these points.[29,49] Sampling–based methods are generic, efficient, simple and probabilistically–complete (guarantees a solution as time approaches infinity).[29]

[*]Ph. D. Candidate, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands.

[†]Assistant Professor, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands, and AIAA Member.

Interpolation techniques, common in autonomous path planning, are usually employed as a post path planning stage. Interpolation techniques' ultimate aim is to optimise a path using smoothed segments to enhance performance and limit workload on control systems and associated actuators.

The twofold aim of this paper is to provide a literature review of the state–of–the–art path planning algorithms that leads to the performance analysis of the two most utilised path planning algorithms, that is the A* and Rapidly–Exploring Random Tree (RRT), in the context of 3D UAV path planning.[29] In this study, path length and computational time with and without a common path smoothing algorithm are the performance parameters considered. In the context of UAVs, path length determines the time of traversal of a path and consequently the time the UAV takes to reach the goal point. The path planning time is fundamental especially in a high time–varying and dynamic environment in which by the end of path planning some parameters may vary. Also, this time will limit the response of the UAV control algorithm to changes in both UAV model and environment. Performance analysis is important in determining the strengths, weaknesses and appropriateness of utilising either of the algorithms for 3D UAV path planning.

The paper will be organised as follows. Section II introduces and discusses the theoretical aspect of the A* and RRT algorithms, followed by a literature review of the above mentioned methods and their associated variants, in Section III. Section IV presents the implementations of the RRT variants, the smoothing algorithm and the test environment. In Section V the results are presented, analysed and discussed. Section VI concludes the study by highlighting the main strongholds and shortcomings of the algorithms in view of 3D UAV path planning.

II. The A* and RRT Algorithms

A. The A* algorithm

The A* algorithm constructs the optimal path based on an evaluation function $f(n)$ that determines the actual cost of an optimal path constrained to pass through n, from a point x_{init} to the goal node of n, $x_{goal} \in \mathbb{R}^M$.[26] n is any node and x_{init} is the starting node in the M–Dimensional available space such that n, $x_{init} \in \mathbb{R}^M$. This evaluation function $f(n)$ is the summation of the actual cost from x_{init} to a node n ($g(n)$) and the actual cost from n to the goal point of n ($h(n)$), where f, g, h: $\in \mathbb{R}^M \to \mathbb{R}$:

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

$g(n)$ and $h(n)$ cannot be determined absolutely as the cost evaluation is a function of the selected resolution. Therefore the cost is estimated by $\hat{g}(n)$ and $\hat{h}(n)$, respectively. Therefore, the evaluation function $\hat{f}(n) \to f(n)$ as the *resolution* $\to \infty$, is estimated by:

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

The A* algorithm first defines x_{init} and x_{goal}: the start and goal nodes, respectively. Consequently, the A* algorithm generates successive nodes through the successor operator (Γ) which defines a set of successor nodes $n_i \in \mathbb{R}^M$ and the associated cost $c_i \in \mathbb{R}$ for each particular node n based on a predefined heuristic graph movement, such that $\mathbb{R}^M \times \mathbb{R} \to \mathbb{R}^M$. *"Open"* nodes are all successor nodes n_i except for nodes that are already on the optimal path or reside on an obstacle. A *"Closed"* node is any node in $\mathbb{R}^M$ that forms part of the optimal path or resides on an obstacle. Algorithm 1 and Figure 1 describe and illustrate the underlying principles of the A* Algorithm.[26]

B. Rapidly-Exploring Random Tree (RRT)

The RRT algorithm grows trees of feasible trajectories by randomly planting seeds. The resultant tree interconnects the start and goal points.[24] Algorithm 2 and Figure 2 describe and illustrate the governing principles of the RRT algorithm.

$\tau \in (\mathbb{R}^M, \mathbb{R}^M \times \mathbb{R})$ refers to tree nodes and their associated edges from the starting node. X_{free} refers to all obstacle–free nodes in the environment. $x_{init} \in \mathbb{R}^M$: start node, $x_{goal} \in \mathbb{R}^M$: goal node, $x_{rand} \in \mathbb{R}^M$: random node, $x_{near} \in \mathbb{R}^M$: nearest node, $l_{near} \in \mathbb{R}^M$: nearest point on edge to x_{rand}, $x_{edge} \in \mathbb{R}^M$: nearest node on edge, $x_{new} \in \mathbb{R}^M$: new node of tree a D distance from x_{near} or l_{near} and K is the maximum preset iterates.

Algorithm 1: A* Algorithm

```
1: Assign $x_{init}$ as *open*
2: Calculate $\hat{f}(x_{init})$ from the start point $x_{init}$ to all possible *open* nodes $n_i$
3: Find the smallest $\hat{f}(x_{init})$ and select the associated node $n$
4: **If** $\hat{f}(x_{init})$ is *empty*
5:     No path Exist
6: **End**
7: **While** $n \notin x_{goal}$
8:     Node $n$ is marked *closed*
9:     Apply the successor operator $\Gamma$
10:    The $\hat{f}$ function is re–calculated for successor nodes of $n$ marking these nodes as *open*
11:    **If** $\hat{f}$ is *empty*
12:        No path Exist
13:    **Else If** $\hat{f}(n_i)$ is now smaller than when $n_i$ was *closed*
14:        Successor closed nodes of $n$ are remarked as *open*
15:    **End**
16:**End**
17:Parent node $n$ is *closed* (Path Found)[26]
```

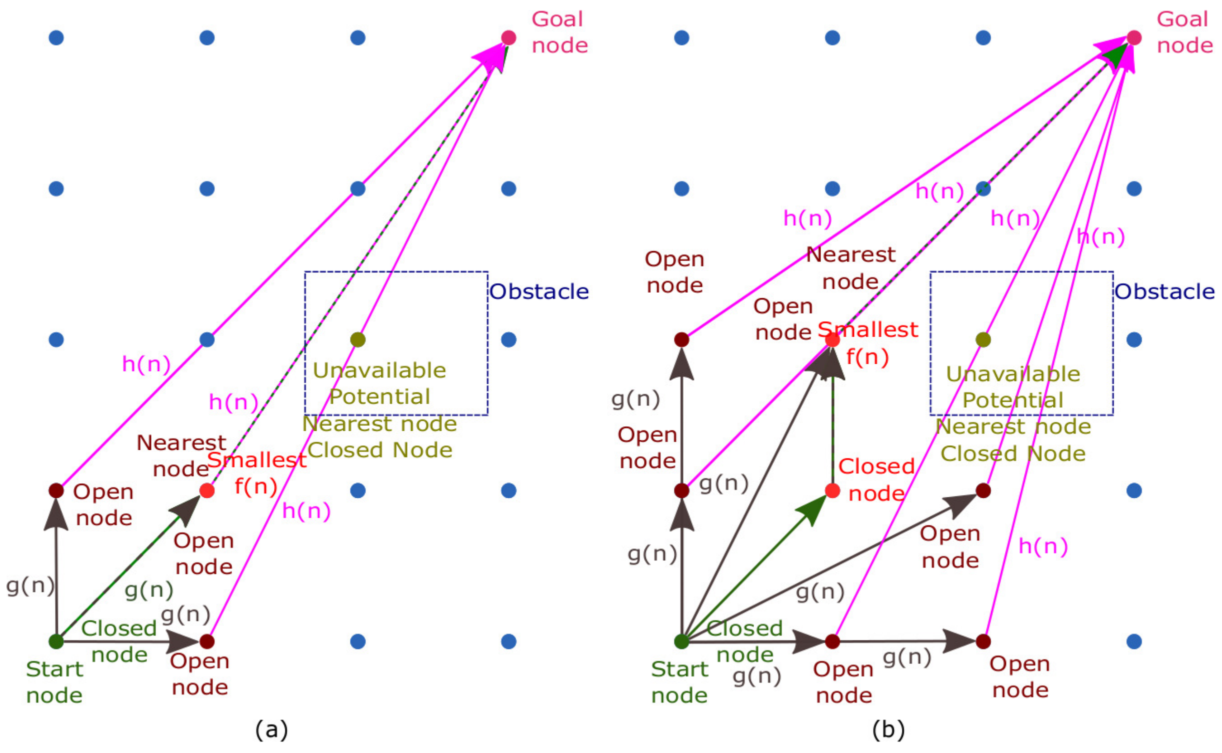

Figure 1. The A* Algorithm: (a) First Iterate and (b) Second Iterate

Algorithm 2: RRT Algorithm

```
1:  Assign start point $x_{init}$ to tree $\tau$
2:  **Repeat** for $K$ iterates **AND While** $x_{goal} \notin \tau$
3:      Generate random state $x_{rand} \in X_{free}$
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:          Define $x_{new}$, $D$ distance from $x_{near}$ to $x_{rand}$.
7:          **If** $x_{near}$ can be connected to $x_{new}$, without collision, then
8:              $x_{new}$ is a new node of $\tau$
9:          **End**
10:     **Else** Define $x_{new}$, $D$ distance from nearest node on $l_{near}$, $x_{edge}$ to $x_{rand}$
11:         **If** $x_{edge}$ can be connected to $x_{new}$, without collision, then
12:             $x_{new}$ is a new node of $\tau$
13:         **End**
14:     **End**
15:     **If** Distance($x_{new}$ to $x_{goal}$) $<$ D **AND** $x_{new}$ can be connected to $x_{goal}$, without collision then
16:         $x_{goal} \in \tau$
17:     **End**
18: **End**
19: **If** $x_{goal} \in \tau$ then path has been found
20: **Else** path not found
21: **End**
```

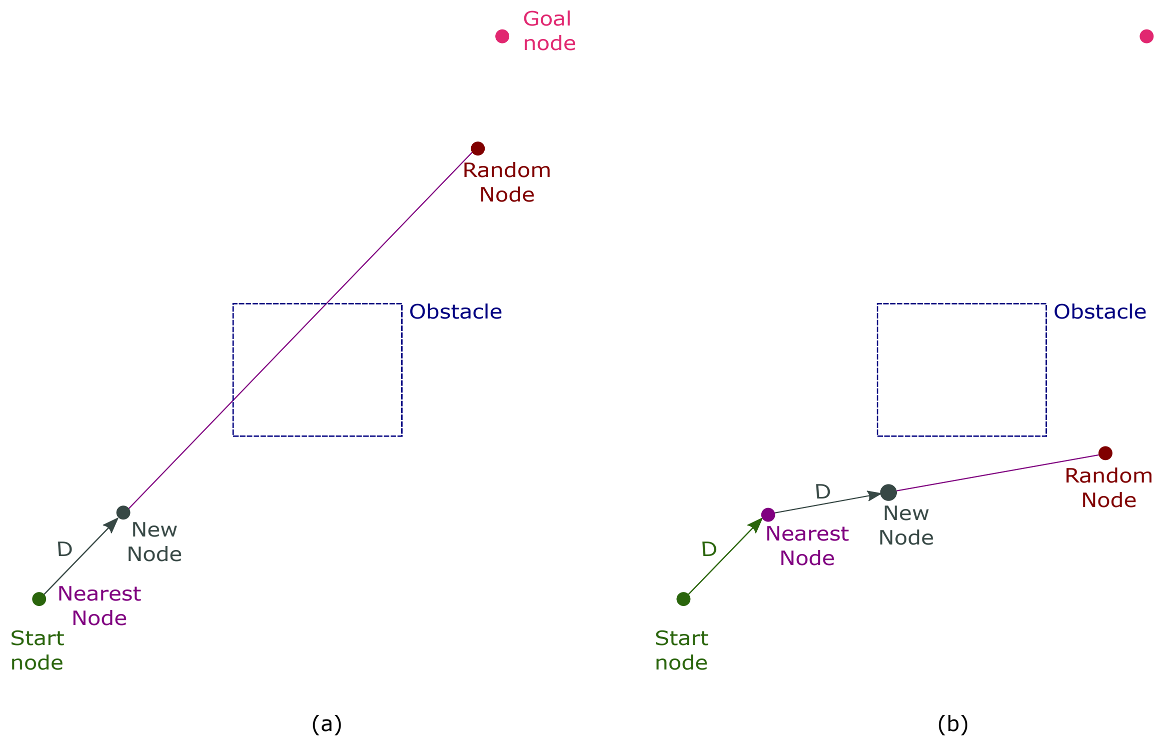

Figure 2. RRT: (a) First Iterate and (b) Second Iterate

III. Literature Review

The literature review will focus on the implementation, utilisation, advantages and disadvantages of graph–based and sampling–based path planning approaches and their variants. A review of the post path planning smoothing algorithms follows. This review will conclude with a comparison of graph–based and sampling–based approaches in a common scenario.

A. Graph–based Approaches

The Dijkstra algorithm was the pioneer graph–based search algorithm.[28] The A* algorithm is an advancement of the Dijkstra algorithm. Due to their relative simplicity both have been widely used for path planning in various fields.

1. Dijkstra Algorithm

Dijkstra's algorithm searches minimum cost paths from two predefined points after exhaustively searching all possible combinations.[28] It has been applied in urban environments,[30] in multi–vehicle simulations[31] and automated race driving in the DARPA challenge.[32, 33] The Continuous Dijkstra and its variants find the shortest paths through analogy with Snell's Law and optical properties.[37]

2. A Algorithm*

The A* algorithm uses a heuristic evaluation function to determine the cost of neighbouring nodes.[18] It is computationally efficient but due to its heuristic nature it offers no guarantee of success even if a path exists.[19] The A* was applied both on grid graphs and visibility graphs.

A number of variants were developed. The Dynamic A* (D*) which is an evolution of the A* in which only the new nodes are updated.[20] The field D* generates low–cost and smooth paths by linear–interpolation planning and re–planning.[34] The focused D* heuristically searches locally from previous search results consequently reducing computational time.[21] Similarly, the Lifelong Planning A* (LPA*) an incremental A*, is equivalent to A* for the first run but for consequent searches it re–uses parts of previous search trees that are identical to the new search tree.[23] The D* Lite developed on the LPA is shorter than the D*, utilises one tie–breaking criterion when comparing priorities and is computationally simpler than D*.[22]

Theta* is composed of the Basic Theta*, which is simple and fast but does not guarantee the optimal path, and the Angle–Propagation Theta*, which results in better worst–case complexity per vertex but generates slower, complexer and longer paths.[35] The Phi*, is an advancement over the basic Theta*, by eliminating nodes no longer in the line–of–sight of their parents.[36] Phi* searches in every angle and executes fast re–planning although it takes more calculation time. The Incremental Phi* does this by using a pre–processing stage[36] used in the Differential A*.[38] The Block A* performs pre–smoothing using a look–up table with predefined short paths prior to the initiation of the A* search.[40] The Block A* is faster than A* and Theta* but does not guarantee a solution.[36] The Jump Point Search (JPS) neglects neighbourhood cells in the surroundings of evaluated cells.[41] This algorithm is fast by exploiting symmetries although it cannot search in every direction.[42]

Other A* variants such as Fringe–Saving A*[46] and Differential A*[38] identify the largest unaltered search areas from the previous search and resume the search from there.[45] Furthermore, incremental heuristic algorithms, such as the Generalized Adaptive A*[47] and Path– and Tree–Adaptive A*,[48] update heuristics with knowledge gained in previous searches enriching performance. The consistency of heuristics is dependent upon cost edges.[45]

The Anytime Repairing A* (ARA*), Anytime D* (AD*) and Anytime Tree Restoring A* (ATRA*) use anytime techniques on A* and D* respectively to quickly generate sub–optimal paths until the allocated time runs up.[43] Truncated incremental search applied to LPA* (TLPA*), D* (TD*) and D* Lite (TD* Lite) selectively re–expands the search utilising previous solution states in view of target sub–optimality bound.[44] Although these variants enhance performance they remain computationally expensive to apply in high–dimensional situations.[34]

The extensive and successful application of the A* algorithm and its numerous variants in different fields indicates that it is well a matured and developed algorithm.

B. Sampling–based Approaches

The Probabilistic Roadmap Method (PRM)[50] and the Rapidly–Exploring Random Tree (RRT),[51] which create cyclic and acyclic random graphs respectively, are the most commonly used sampling–based approaches.[29]

1. *Probabilistic Roadmap Method (PRM)*

The PRM method is composed of the learning phase in which sample points are connected to one another forming a roadmap using a local planner and the query phase in which a graph search algorithm searches a feasible path from the start and end points using a previously constructed roadmap.[50] The following are variants of PRMs.

The Lazy PRM delays collision checking to only when necessary.[52] The PRM* introduces a variable distance metric that enhances optimality.[25] The cell–based PRM (CPRM) discretises the configuration space into cells prioritising cells close to direct paths or disconnected components biasing searches to solution areas consequently resulting in faster re–planning.[53] The Dynamic Roadmap (DRM) generates a roadmap assuming an obstacle–free space, maps workspace maps to roadmap edges and invalidates edges residing on obstacles during online re–planning.[54] Similarly, Pomarlan *et. al.*[55] increases the cost of near invalidated edges pushing the path away from obstacles.

The Flexible Anytime Dynamic PRM (FADPRM) segments the environment into desirable zones, an A* search is done to continually improve the solution based on priority heuristics near frontier edges.[56] The Elastic Roadmap uses feedback controllers to connect roadmap vertices (associated with workspace features) in a continually–changing environment.[57] Reactive Deformation Roadmap (RDR) generates a roadmap based on dynamic milestones and reactive links. The roadmap attract milestones and repel obstacles while ensures connectivity with the addition and removal of milestones and associated links.[58]

2. *Rapidly–Exploring Random Tree (RRT)*

The Rapidly–Exploring Random Tree (RRT)[51] constructs a unidirectional search tree by randomly selecting and interconnecting obstacle–free states until a tree branch reaches the goal node. The integration of motion parameters including system dynamics is a key advantage of the RRT with respect to PRM.[51] The standard RRT is efficient in complex high–dimensional environments although paths may not be optimal and may require smoothing.[59–61] RRT properties allow them to be applied in non–holonomic and motion constraint environments as RRT trees expand uniformly in unoccupied spaces.[62]

The RRT–Connect or bidirectional RRT grows a tree also from the goal node.[63] Execution Extended RRT (ERRT) re–plans invalidated paths probabilistically using both new random sample points and nodes from the invalidated path.[65] Moreover, evolutionary algorithms with bi–directional Rapidly Exploring Evolutionary Trees (RET) were proposed by Martin *et. al.*, reporting improvement in changing environments making use of Spatial KD–Trees for efficient neighbouring node look–up,[65] although both RRT and RET depends upon prior knowledge of the environment.[64] But Gros *et. al.* notes that unidirectional tree structures are more suited for cluttered environments with non–holonomic differential constraints than bidirectional approaches due to inherent flexibility.[5]

The basic RRT lacks optimality in cost function. The RRT* advancement,[66] sensibly reduces cost by considering a minimum length cost, exhibits asymptotic optimality, but consequently suffers in defining the local optimum from the start.[25] RRT* only considers path–quality when adding or removing nodes and does not consider the configuration–cost function when sampling. RRT* converges slowly in high–dimensional environments.[67] The RRT*–Smart was designed to increase the rate of convergence over the RRT*, producing optimum or near optimum paths, reducing execution time.[76]

Frazzoli *et. al.*[79] developed an RRT–like algorithm that connects states to all nodes not just the closest. An obstacle–free distance cost function is utilised and after checking for collisions the algorithm is given a definite time to re–plan. The cell–RRT utilises probabilistic approaches after decomposing the environment to make optimal use of planning approaches.[85] Tests with different decomposition granularities and planner heuristics show that goal–biasing reduces path length and solution's success rate.[85] Performance is enhanced by considering previously defined path segments and environmental analysis prior environment decomposition.[85]

Transition–based RRT (T–RRT) uses a Metropolis–like transition test to bias exploration to low–cost regions by adding nodes in these areas.[60] On the negative side, the path–quality criterion is neglected during the path creation when creating edges, offering no guarantee of path optimality.[60] The Transition–based RRT* (T–RRT*) combines the path–quality criterion of the RRT* with the low–cost region bias of the T–RRT.[60] Different physical (lane changes, road edges and terrain grade) and logical environmental (dynamics and obstacle–avoidance including both fixed and moving) biases were utilised for efficient RRT–tree construction.[73–75] Shang *et. al.*[86] biased the RRT in a complex 3D environment on its own planning history. Like in graph–based approaches, anytime paradigms were applied to RRTs[68–70] to quickly obtain a feasible path which converges to optimum using previous searches.

Consequently, the Anytime T–RRT (AT–RRT) applies the anytime methodology to re–plan T–RRT paths, converging to optimum.[60] Other anytime applications to RRT are the RRT–Roadmap (RRM) which balances path exploration and refinement based on RRT*[71] a meta–approach using path hybridization with short–cutting.[72] The Dynamic RRT (DRRT) discards children of invalidated vertices and re–plans from the goal node to the current node.[77] In the Anytime Dynamic RRT (AD–RRT), the benefits of both paradigms is exploited.[78] In the multipartite RRT (MP–RRT) sampling and re–planning is performed as in ERRT, discarding invalid nodes as in DRRT.[80]

The RRT^X is also an asymptotically optimal real–time re–planning algorithm that generates continually valid and optimal paths in a dynamic environment. In the event of a collision due to a newly detected obstacle, the search tree is updated and the path–to–goal rooted sub–tree is re–planned.[87] Similarly to RRT^X, the Partial Motion Planner (PMP) assigns a fixed time window for planning and execution. This real–time planner, builds a path with a predefined length from the current state to the goal node. The environmental model is then updated and the process is repeated while planning occurs.[81] The Closed Loop RRT (CL–RRT), also a real–time planner, uses the fixed time–window for re–routing a tree from the model's current state in a closed–loop feedback approach. The CL–RRT method works for nonlinear controllers and models with unstable dynamics. The associated stabilisation controller reduces prediction errors and.[73] The Greedy Incremental Path–Directed (GRIP) also considers kinodynamic constraints when growing a tree from current state in a finite computational time window. This process is done prior model's initial movement.[82] Consequently, the planner updates the tree by deterministically propagating valid nodes from previous searches while discarding invalid nodes leading to collisions. This greedy approach is biased to probabilistically complete state–space exploration.[82]

Reconfigurable Random Forests (RRF) uses the concept of MP–RRT by separating and growing disconnected trees, consequently interconnecting trees and trimming them when trees become too dense.[83] The Lazy Reconfigurable Forest (LRF) maintains the forest of trees to adapt to moving trees and obstacles. Only the edges along the path are validated not the entire forest as with the DRRT approach.[84] Similarly, Multiple RRTs (MRRT) that simultaneously expand more than one tree were also proposed. Their aim is to enhance algorithmic performance and expansion issues.[62] Multiple Incremental RRTs (MIRRT) uses the incremental extension to maintain formed trees for future path planning requests.[62]

Probabilistic approaches evolved from PRMs to RRTs and their extensions to either mitigate inherent shortcomings and/or to best perform in particular scenarios. Similarly to A*, the extend to which this methodology was enhanced and applied strengthens its candidature for robust and effective UAV path planning.

C. Interpolation Techniques

Interpolation techniques construct a new smoother path based on a previously defined path to improve trajectory continuity, feasibility, vehicle and environmental kinematic and dynamic constraints.[27,88]

Straight lines and circular shapes are the simplest, shortest and computationally cheapest interpolation methods.[89,90] The path is not continuous and depends on global path nodes.[27] Clothoid Curves, defined by Fresnel integrals,[91] make linear–to–curved transitions smoother using linear changes in curvature. Clothoid curves are continuous, suited for local planning, computationally intensive and non–smooth.[27] Also, different order Polynomial Curves are utilised to define curves based on the positional, angular and curvature constraints of the interconnecting points. Polynomial Curves are continuous and smooth but computing coefficients for specific motion states is computationally intensive.[27]

Bézier Curves, based on the Bernstein Polynomials, use control points to define a continuously–changing curvature. Bézier Curves have been used extensively in numerous fields due to their modularity and low computational demand with low curve degree.[27]

Spline Curves are piecewise polynomials defined as a set of polynomial curves, B–Splines, Bézier Curves and/or Clothoid Curves.[27] A Spline curve is not computationally intensive but is not optimal as the algorithm stresses is upon continuity between path segments.[27]

D. Comparing Approaches

Tsai *et. al.*[92] first applied the RRT for 3D path planning of multirotor UAVs and the Dijikstra algorithm for path cost enhancement. In the second configuration of the same setup, 3D path planning was realised with the RRT–Connect algorithm and the path cost is enhanced using an amended A* that considers only the flight path angle instead of discretising the environment into the grid. Furthermore, Bézier Curves are employed for 3D path smoothing. They report that path efficiency and computational cost are improved in the second configuration when compared with the first.[92]

Rao *et. al.*[93] applied and compared the A* and RRT–Connect path planning algorithms to generated feasible energy–optimal 2D paths for Autonomous Underwater Vehicles (AUVs) based on ocean currents. The RRT–Connect tree is Voronoi–biased through some heuristics adjusted by the Iterative k–Nearest RRT (IkRRT) which considers k–neighbours instead of a single neighbour.[94] Rao *et. al.*[93] concluded that both methods are suitable for this application with RRT avoiding high energy shallow regions due to the elimination of grid discretisation while RRT–Connect struggled to find solutions against Voronoi bias. On the other hand, the A* can handle highly difficult missions against strong currents.[93]

IV. RRT algorithm variants, Smoothing algorithm and Test Environment

The RRT algorithm described in Section IIB is constrained by a step size. This is included to emulate the UAV model's translational constraints. This limitation increases path planning time. In this paper these translational constraints are waived and the segmentation of the path into waypoints based on UAV's translational constraints shall be considered only by the path following algorithm. Moreover, this adaption is included to offer a fair comparison between the standard RRT and Multiple RRT (MRRT) algorithms. The latter initiates with a predetermined number of trees which grows per iterate towards the random node and consequently interconnects without considering step size constraints.

A. RRT algorithm without step size constraint

The RRT algorithm is slightly altered by removing the step size constraint. The altered algorithm is defined in Algorithm 3 utilising the same variables as in Algorithm 2. The governing principles of this altered algorithm is illustrated in Figure 3.

B. Multiple Rapidly–Exploring Random Tree (MRRT)

In MRRT, an arbitrary number of RRTs are simultaneous expanded, with two starting from the start and end nodes and the remaining starting from predefined or randomly–selected nodes.[62]

$\tau_{1..N} \in (\mathbb{R}^M, \mathbb{R}^M \times \mathbb{R})$ refers to tree nodes and their associated edges starting from the starting and goal nodes, respectively. N is the number of trees and N_{Max} is maximum number of allowed trees. The other variables are the same as in Algorithm 3. Algorithm 3 and Figure 4 describe and illustrate the governing principles of the MRRT algorithm.

C. Smoothing Algorithm

A standard smoothing algorithm is utilised by both A* and RRT algorithms. x_{p1} and $x_{p2} \in \mathbb{R}^M$ are two randomly–selected nodes from the A*/RRT/MRRT path nodes τ_{final}. x_{pnt1} and x_{pnt2} are two randomly–selected points, one on the segment connecting x_{p1} to x_{p1+1} and the other on the segment connecting x_{p2} to x_{p2+1}. E is the maximum number of smoothing iterates. The smoothing principle is defined in algorithm 4 and illustrated in Figure 5.

D. Experimental Setup

The experimental space is defined as a cube of 1x1x1 (the units are generic and can be factored according to the application scenario) with centre (0,0,0) and limits [-0.5 $\rightarrow$ 0.5, -0.5 $\rightarrow$ 0.5, -0.5 $\rightarrow$ 0.5]. These test

Algorithm 3: RRT Algorithm without step size constraint

```
1: Assign start point $x_{init}$ to tree $\tau$
2: Repeat for $K$ iterates AND While $x_{goal} \notin \tau$
3:     Generate random state $x_{rand} \in X_{free}$
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:         If $x_{near}$ can be connected to $x_{rand}$, without collision, then
7:             $x_{rand}$ is a new node of $\tau$
8:             If $x_{rand}$ can be connected to $x_{goal}$, without collision then
9:                 $x_{goal} \in \tau$
10:            End
11:        End
12:    Else
13:        If $x_{edge}$ can be connected to $x_{rand}$, without collision, then
14:            $x_{rand}$ is a new node of $\tau$
15:            If $x_{rand}$ can be connected to $x_{goal}$, without collision then
16:                $x_{goal} \in \tau$
17:            End
18:        End
19:    End
20: End
21: If $x_{goal} \in \tau$ then path has been found
22: Else path not found
23: End
```

Algorithm 4: Multiple RRT Algorithm

```
1: Assign start point $x_{init}$ to tree $\tau_1$, goal point $x_{goal}$ to tree $\tau_2$ and $A$ predefined nodes to trees $\tau_{3...A--2}$
2: Repeat for $K$ iterates OR (N == 1)
3:     Generate random state $x_{rand} \in X_{free}$
4:     Find the nearest neighbouring node $x_{near}$ or nearest neighbouring edge $l_{near}$ for each tree $\tau_{1..N}$,
         if possible (due to obstacles).
5:     If more than one tree $\tau_{i,j,k....}$ can be connected to $x_{rand}$, without collision, then $x_{rand}$ is the common
         node $x_{new}$ connecting two or more trees to one tree $\tau_j$.
6:         Else If one tree $\tau_i$ can be connected to $x_{rand}$, without collision, then $x_{new} == x_{rand}$ is new node
             of $\tau_i$.
7:             Else If node is not on obstacle and cannot be connected to a tree, set $x_{rand}$ as new node
                 $x_{new}$ of new tree $\tau_{N+1}$ if $N < N_{Max}$
8:             End
9:         End
10:    End
11: End
12: If N == 1 then path has been found
13: Else path not found
14: End
```

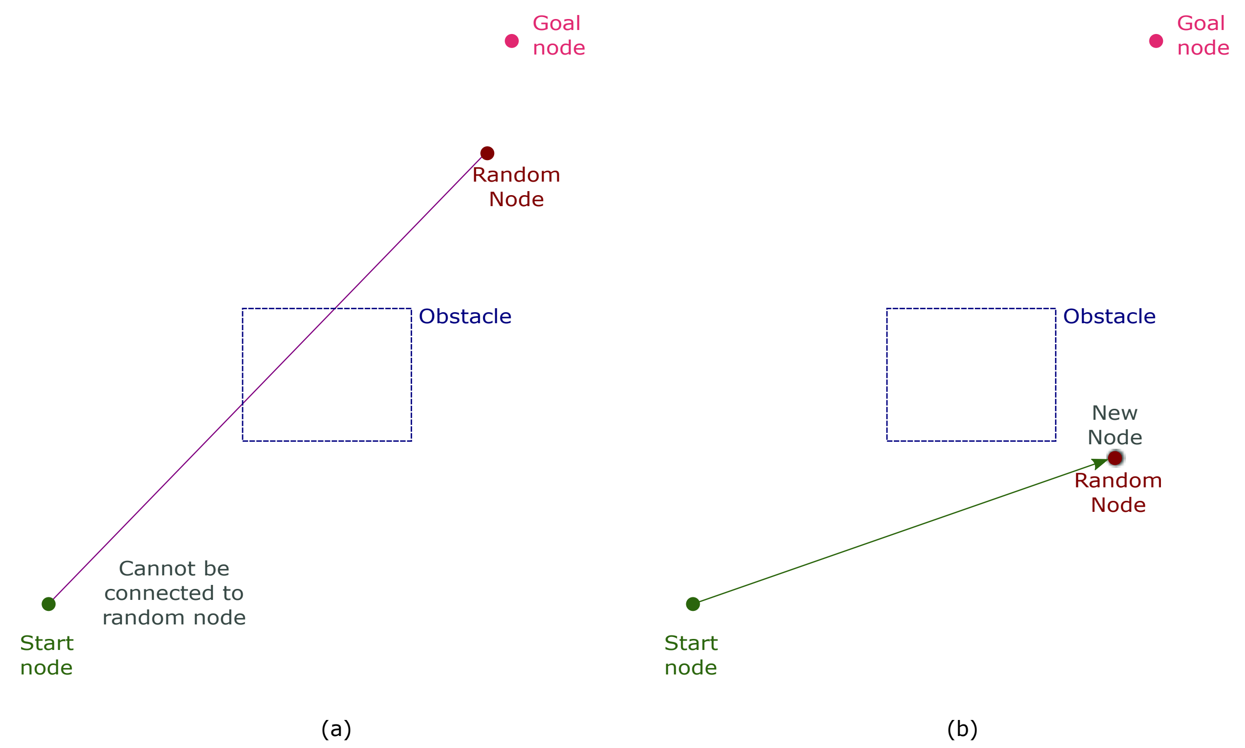

Figure 3. RRT Algorithm without step size constraint: (a) First Iterate and (b) Second Iterate

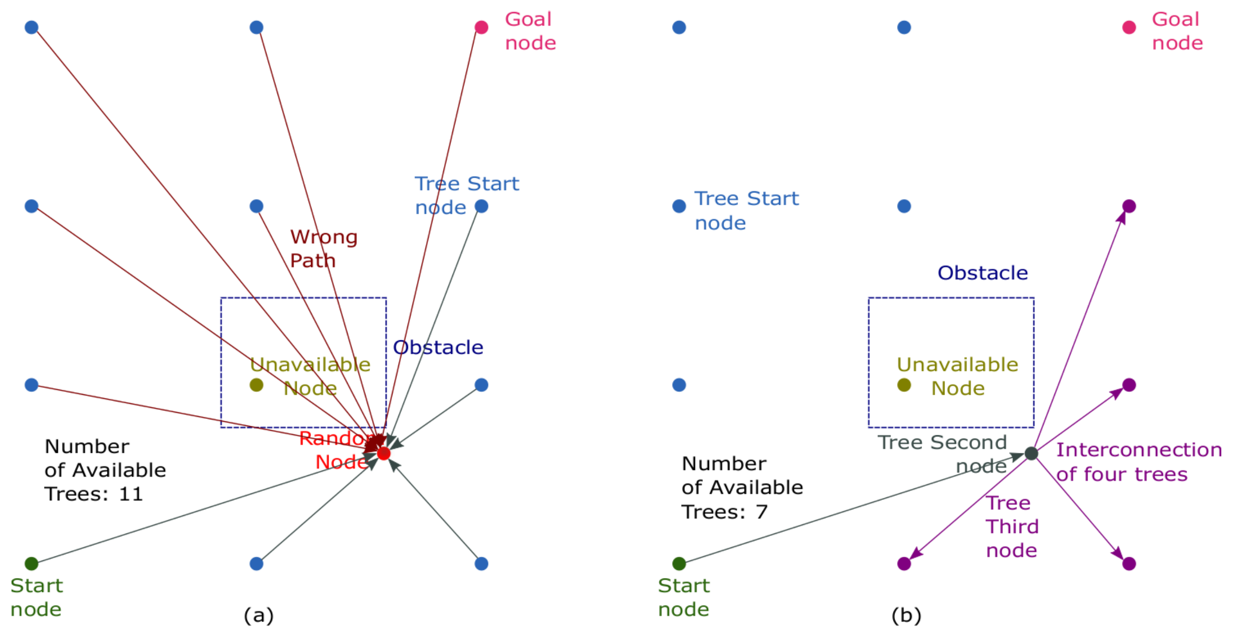

Figure 4. Multiple RRT: (a) First Iterate (b) Interconnection of trees (five trees become one) and re–parenting

Algorithm 5: Smoothing Algorithm

```
1: Repeat for $E$
2:     Randomly select $x_{p1}$ and $x_{p2}$ from $\tau_{final}$
3:     Swap $x_{p1}$ and $x_{p2}$ if $x_{p2}$ is a node further than $x_{p1}$ to the goal node $x_{goal}$
4:     Randomly select $x_{pnt1}$ and $x_{pnt2}$ from the path points
5:     If segment connecting $x_{pnt1}$ and $x_{pnt2}$ is obstacle free then remove all intermediate nodes
       i.e. from $x_{p1+1}$ to $x_{p2}$
6: End
```

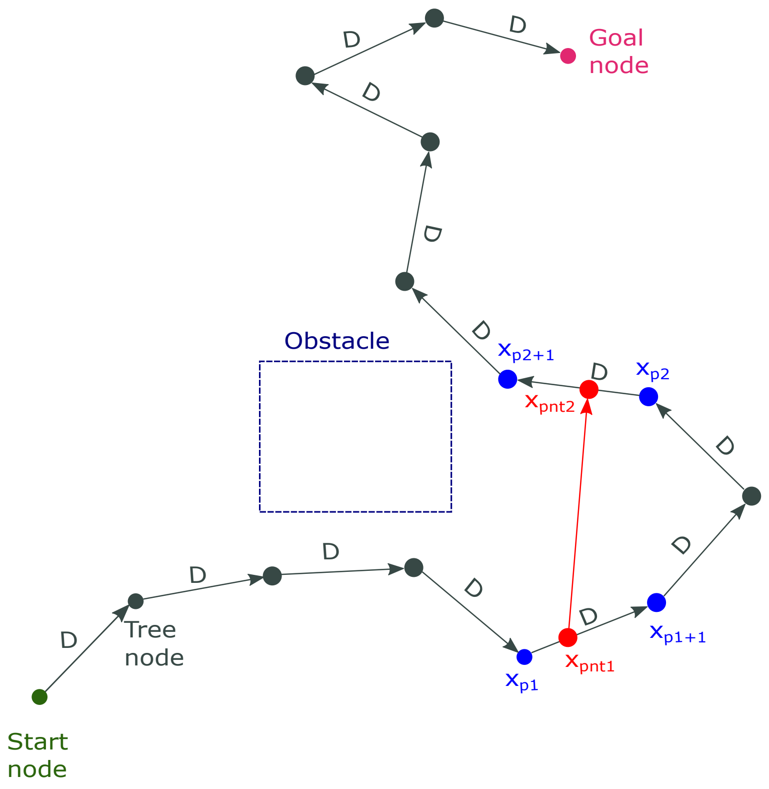

Figure 5. Smoothing Algorithm

scenario were originally developed by Clifton *et. al.*[95] for assessing the MRRT algorithm and made available online in.[96] For all tests the start and goal nodes are defined at [0,-0.5,0] and [0,0.5,0], respectively. Three different obstacle scenarios, illustrated in Figure 6 are considered:

1) Two obstacle planes at y=0 and y=0.2 with a square opening from x=0.15 to x=0.35 and z=0.15 to z=0.35 for both planes.
2) Five obstacle planes:
    2.1) Two planes at y=-0.2 and y=0.2 with a square opening from x=0.15 to x=0.35 and z=0.15 to z=0.35 for both planes;
    2.2) One plane at y=0 with a square opening from x=-0.35 to x=-0.15 and z=-0.35 to z=-0.15; and
    2.3) Two planes at z=-0.25 and z=0.25 with no openings.
3) Same as 2) with the addition of two planes at y =-0.4 and y=0.4 with a square opening from x=-0.35 to x=-0.15 and z=-0.35 to z=-0.15.

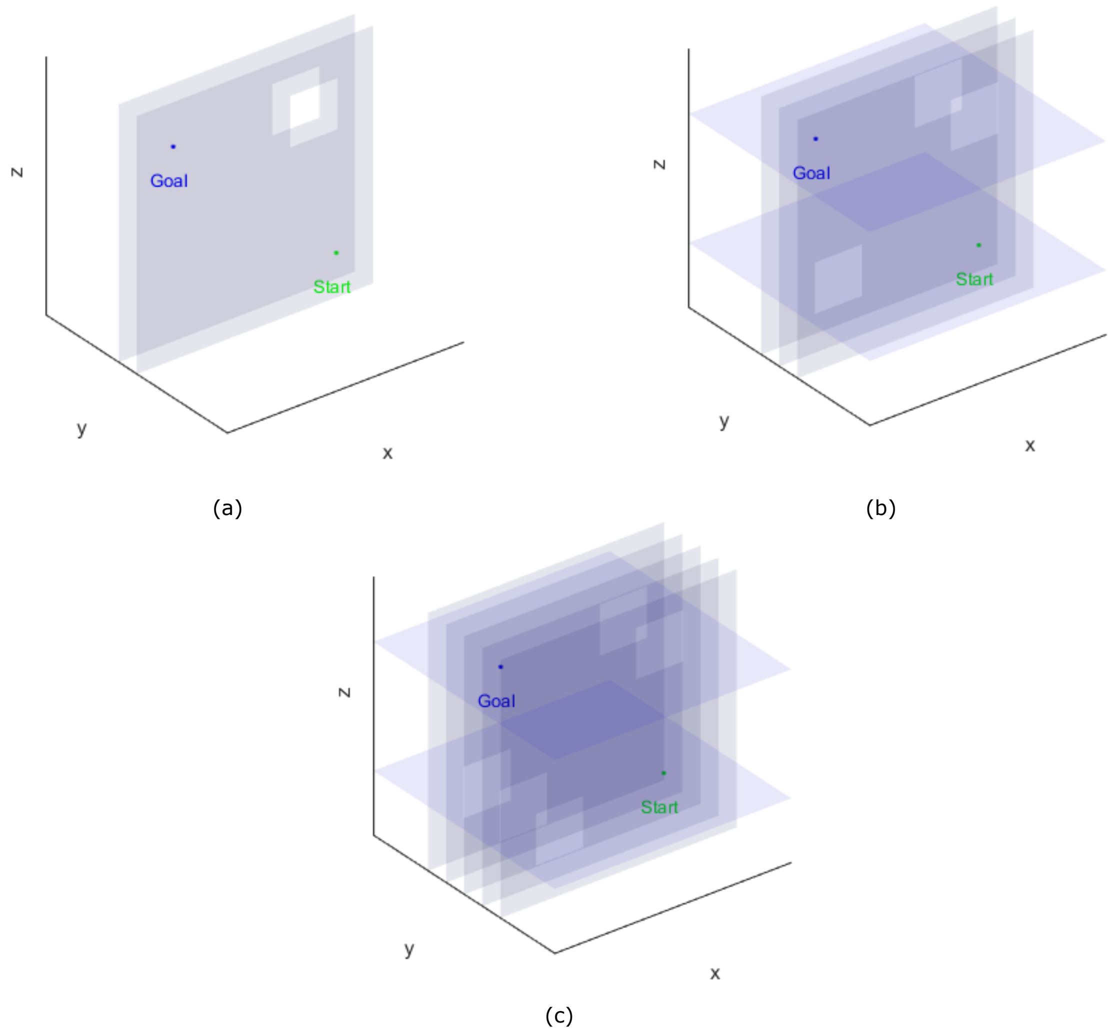

Figure 6. Obstacle Scenarios: (a) First Scenario (b) Second Scenario and (c) Third Scenario

The resolution, the step size per iterate and the number of seeds/nodes per axis are considered as the control variables for the A*, RRT and MRRT algorithms, respectively. For the A* the resolutions considered start from 11 pixels (10 spaces) to 29 pixels (28 spaces) in steps of two in a unity square box of 1x1x1. The

unity square box represents a general measure of distance. The path length is generic with respect to the unity square box in plots, depending on the environment in which the algorithm is implemented (example: cm, m or km). The step size per iterate is varied in steps of $\frac{1}{10+(2xi)}$ (i is incremented by 1 every step) between $\frac{1}{10}$ and $\frac{1}{28}$ to reflect the resolution considered for the A* approach. For the MRRT, the number of seeds per axis ranges from 2 to 20 also in steps of two.
The maximum number of smoothing iterates E is set at 1000. The random generator for the RRT algorithm (A* not required) utilises the same random sequence for all runs. Also, for the RRT algorithm the maximum number of iterates (K) is set to 2000. The maximum number of trees (N) is set as a function of (K):

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

For both algorithms each test is performed 100 times and the path length before and after smoothing and the associated time are considered as performance measures [a]. The unsmoothed length, smoothed length, path generation time and path smoothing time per 100 iterates are illustrated in Figure 7 – 12.

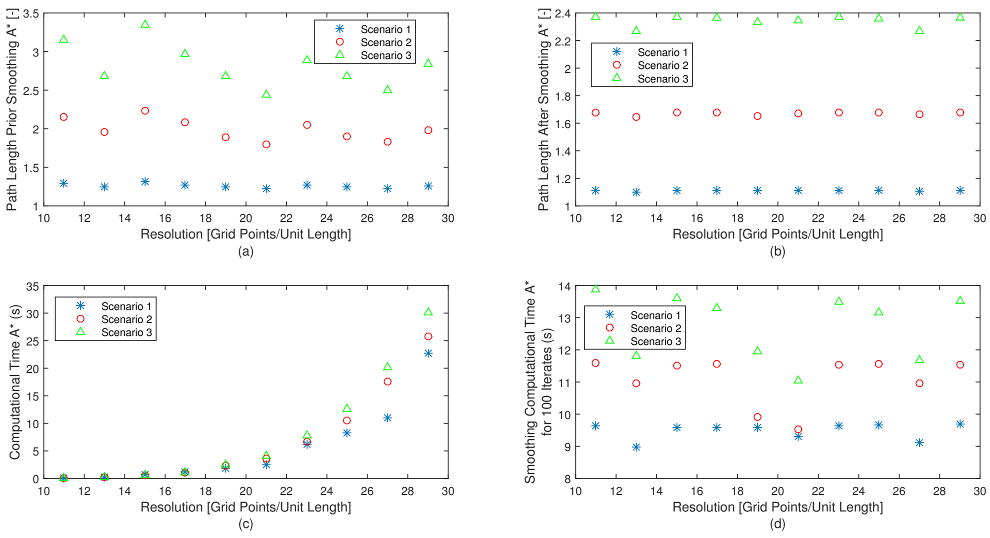

Figure 7. A* Results: (a) Path Length Prior Smoothing (Generic length with respect to a unity square box) (b) Path Length After Smoothing (c) Computational Time and (d) Smoothing Computational Time for 100 Iterates

V. Results

Figure 7 shows that a solution is possible in all scenarios for all resolutions considered. Although this may project the A* as guaranteeing a solution when it exists, this is not the case as remarked in literature.[19] From Figure 7 (a) it is clear that the path length prior smoothing is primarily dependent upon the scenario's difficulty in finding a possible path. In fact in scenario 2 the length is 50% more with respect to scenario 1 since the planner needs to pass from three windows on opposite alternating extreme sides of obstacle planes instead of two windows parallel to each other as seen in Figure 6 (b) and (a), respectively. Similarly, in scenario 3, the length has doubled, since now the planner needs to traverse five planes with windows each on opposite extremes as in Figure 6 (c).

The distances and therefore positions of *'open'* grid points effect the path length prior smoothing, since changing the resolution will ultimately change *'open'* grid point positions resulting in shorter or longer possible path for the same scenario. In situations, like scenarios 2 and 3, where few possible paths exist, the change in resolution and consequently grid point positions will effect the length. This hypothesis is confirmed in Figure 7 (a) by the relatively high ripple factor of scenarios 2 and 3. On the other hand, since the number of available paths is significantly larger for scenario 1 this ripple effect is highly diminished.

[a]The tests are performed using an Intel Xeon ES–1650, 3.2GHz

The path length after smoothing further confirms the dependence of the length on the path difficulty. As any point on the path is now considered available, the rippling effect is sensibly reduced. Overall, the mean smoothed path length was shortened to 88%, 84% and 83% of the mean unsmoothed length for consecutive scenarios. This confirms the validity of the smoothing algorithm. The increase of the smoothing iterates from 1,000 to 10,000 have an insignificant effect on the smoothed path length although this increase results in an increase in the number of path points. An increase in path points, increases the control maneuvering demand on the path following control system, consequently reducing performance.

The path generation time is dependent primarily on the grid resolution and secondly on the path difficulty. The higher the resolution, the more grid points are available to be considered by the A* algorithm. Moreover, the higher the path difficulty, the longer the possible paths and consequently more time is required to find the path.

The path smoothing time is in the region of milliseconds per iterate, implying that it is much smaller than the path generation time. Figure 7 (d) shows that the path smoothing time is only slightly dependent on path difficulty.

Theoretically, the path length is inversely proportional with resolution implying that as the resolution approaches zero the path length approaches infinity and vice–versa. This relationship is not visualised in the scenarios considered as the obstacles in all scenarios involve 2–D planes with windows. In these situations, the resolution has minimal effect since, based on grid positions and the inherent path construction process, it is either possible to construct a path or not. Therefore, to confirm this hypothesis, a simple obstacle scenario consisting of a box 0.4x0.4x0.4 centred at (0,0,0) was constructed as shown Figure 8.

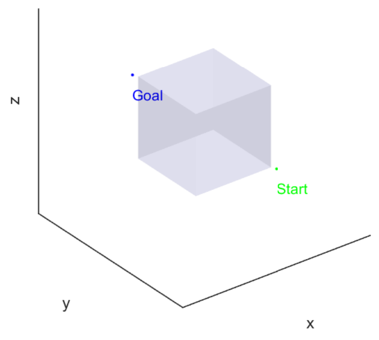

Figure 8. A* Simple Obstacle Scenario

Results in Figure 9 confirm the hypothesis about inverse proportionality. Moreover, due to the relative simplicity of the path, the smoothing algorithm has marginal effect on path length. As predicted, the computational time for path generation exhibits an inverse relationship with resolution as A* is required to process more points as the resolution increases.

Figure 10 shows that the RRT was able to construct a path in all scenarios for all step sizes. This confirms that RRT is probabilistically complete.[29] From Figure 10 (a) it can be deduced that for simple scenarios (Scenario 1) the path length prior smoothing is independent of the step size. Theoretically, for an optimal path the step size is directly proportional to path length, as the planner can go around an obstacle tighter to the obstacle for shorter step sizes. But tests confirm that the lengthier the tree branches the greater the probability of intersection with obstacle planes. Therefore, the larger the step size the lesser the number of possible tree branches that are used to construct the path, reducing the zig–zagging effect. In fact, the smaller the step size the more lower amplitude zig–zagging will result. This opposite relationship confirms the lack of optimality expressed in literature.[29]

The lack of optimality can be deduced from the difference in length between the unsmoothed and smoothed paths. The mean smoothed path length was 54%, 50% and 51% of the mean unsmoothed path length for scenarios 1 to 3, respectively. The same line profile was retained for all three scenarios. This implies that the smoothing algorithm's optimality potential is limited by the unsmoothed path points as theoretically the step size must be directly proportional to path length. Similar to A*, the increase in

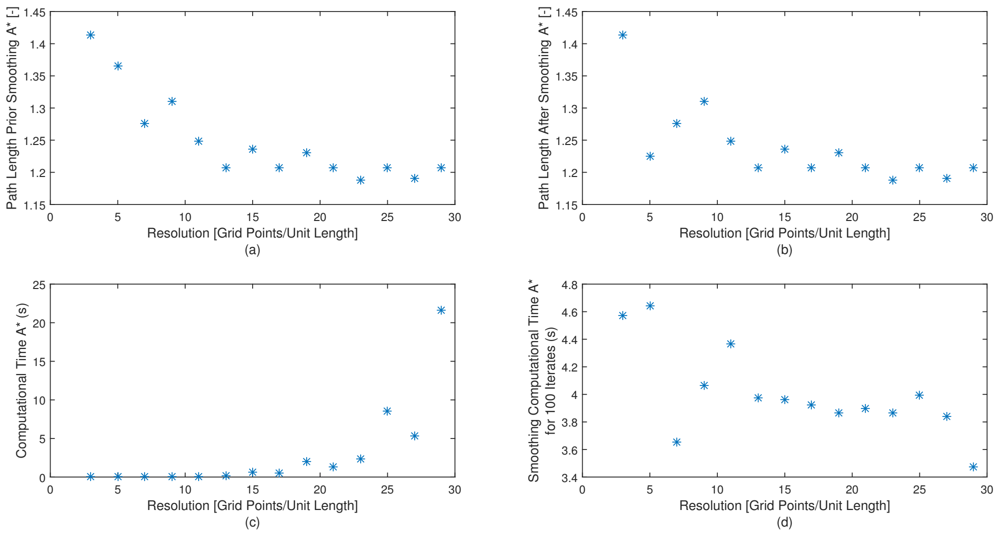

Figure 9. A* Simple Scenario Results: (a) Path Length Prior Smoothing (Generic length with respect to a unity square box) (b) Path Length After Smoothing (c) Computational Time and (d) Smoothing Computational Time for 100 Iterates

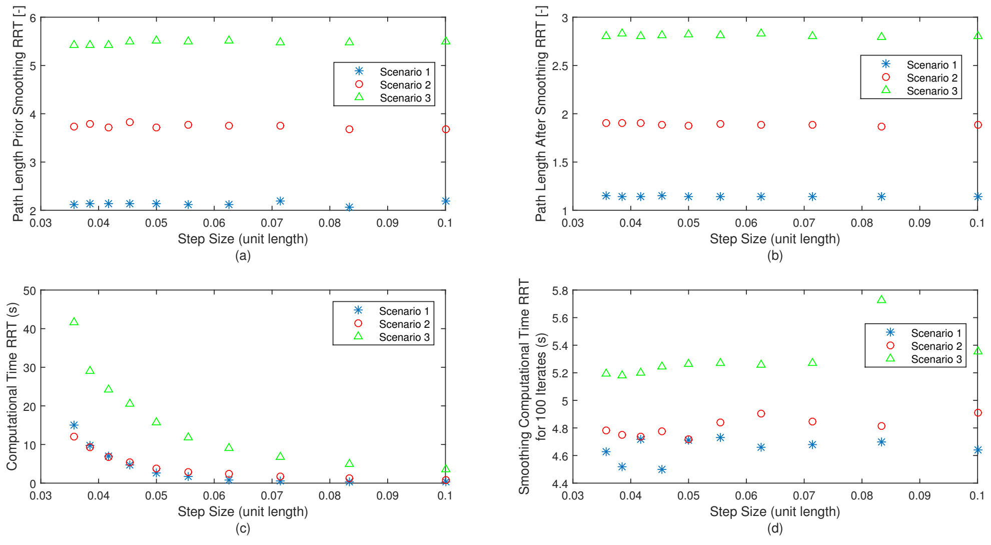

Figure 10. RRT Results: (a) Path Length Prior Smoothing (Generic length with respect to a unity square box) (b) Path Length After Smoothing (c) Computational Time and (d) Smoothing Computational Time for 100 Iterates

smoothing iterates from 1,000 to 10,000 resulted in a negligible effect on path planning performance, although the smoothing time increased by a factor 2.23. This shows that although the smoothing iterates are increased by 10 times, after a specific number of iterates the effect of smoothing saturates and the path length approaches an optimum.

Overall, the unsmoothed path length for A* is shorter than that for RRT even as the scenario complexity increases. In fact, A*'s unsmoothed path length is only 59%, 53% and 51% of the unsmoothed RRT path. This further confirms the optimality of A* and the lack of optimality for RRT. The results cannot be directly compared as the RRT is constrained by a step size per iterate implying that the planner can theoretically move to every point on the circumference of a circle of radius equal to the step size and centre at the nearest node while in the A* the planner is limited to a predefined set of points. Moreover, the mean variance in length for different scenarios is smaller for RRT than for A* for the same scenario volume, as the RRT explores more volume (leading to zig–zagging) than A* which always considers the shortest segment irrespective of the scenario considered.

The length reduction by smoothing of the RRT is more effective than the A* as the latter is able to construct shortest paths for a particular resolution while the RRT lacks in optimality. The curve shapes of the unsmoothed and smoothed lengths is similar as to A* case, confirming that the smoothing algorithm is limited by the unsmoothed path points.

As predicted, and similarly to A*, the path generation time is inversely proportional to the step size since the tree will require more time to explore more volume, vital to find a path, if the tree branches are shorter. Similar to A*, the simpler the scenario the less time is required to find the path. This is because the amount of branches created prior to finding the path is much higher for obstacle rich scenarios compared to simpler scenarios. As for the A*, the path smoothing time is almost independent of the scenario and step size considered, although for scenario 3 it is marginally higher. Furthermore, the path smoothing time is negligible (5 milliseconds per iterate) when compared to the path computation time.

Parameter	Scenario 1	Scenario 2	Scenario 3
Path Length Prior Smoothing [–]	1.8316	3.3794	5.2724
Path Length After Smoothing [–]	1.1315	1.8806	2.8099
Computational Time RRT (s)	0.0094	0.6656	1.6364
Smoothing Computational Time for 100 Iterates (s)	3.5967	4.6568	5.1689

Table 1. RRT without step size constraints

The results of Table 1 illustrate the mean values for 100 iterates for each scenario without limiting the path planner's progression step per iterate. The path length prior and after smoothing for RRT with step size constraint is 17%, 11% and 4% (unsmoothed) and 1%, 0.4% and 0.04% (smoothed) longer than the respective values of RRT without step size constraints for scenarios 1 to 3, respectively. Results confirm that the path length prior smoothing is marginally improved by removing constraints, especially for simple scenarios, as the algorithm can directly reach the goal node if a path exist (irrespective of length) as illustrated in Figure 11 (c). This cannot be done when the RRT is constraint, even when a direct path exist, as the distance between the nearest node to the goal node and the goal node must be equal or smaller than the step size. As a result, instead of a direct line, a zig–zag path will be created as a randomly generated node will not reside on the direct path as illustrated in Figure 11 (a) and (c). For more complex scenarios, the direct connection between a node and a goal node is highly limited by the obstacle rich scenario, nullifying the difference in path length between the two algorithms as illustrated in Figure 11 (b) and (d). The smoothing algorithm was able to smoothen the low amplitude ripple in the RRT with step size constraint, nearing the performance of the latter method with the RRT without constraints. This is particularly evident in obstacle rich scenarios as the ripple factor is more limited.

The path generation time is reduced by factor of 454, 7 and 10 for the RRT with step size constraint for consecutive scenarios. This substantial improvement results as the planner is able to connect directly to the random node (if possible) irrespective of length. Therefore one iterate in unconstrained RRT will require numerous iterates (with associated time) and also giving rise to a zig–zagging nature. This result confirms that, especially for simple scenarios, significant amount of time is wasted by the RRT without constraints when a path can be easily connected directly. The smoothing time of the unconstrained RRT decreased by

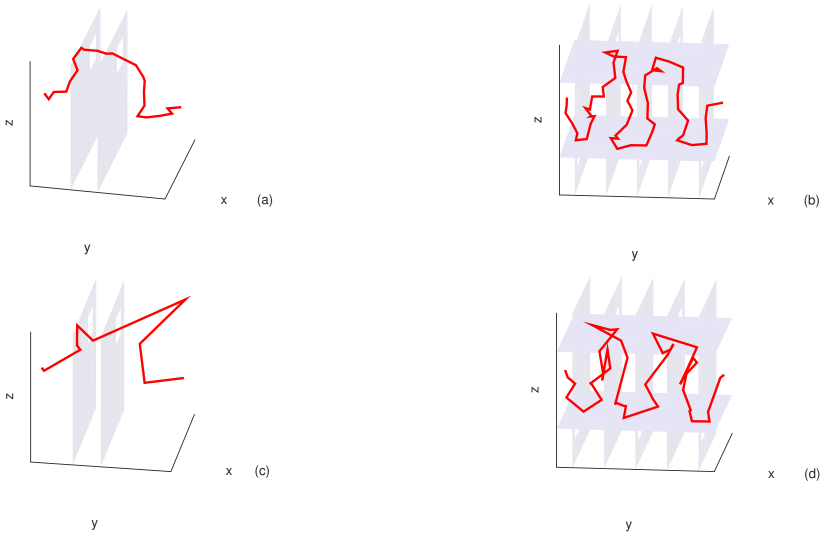

Figure 11. RRT vs RRT without constraints: (a) RRT path for Scenario 1 (b) RRT path for Scenario 3 (c) RRT without constraint path for Scenario 1 and (d) RRT without constraint path for Scenario 3

23%, 3% and 2% with respect to the original RRT for consecutive scenarios, respectively. The increase in smoothing time for simple scenarios results as the unconstrained RRT is able to generate almost optimal paths. In fact, the unsmoothed path contains less than 10 nodes for scenario 1 for all iterates. In this light, the smoothing algorithm requires only few iterates to optimise the paths.

The path length results of Table 2 confirm the optimality of the A* in finding the shortest possible path and the lack of optimality of the RRT even if the RRT planner is allowed to travel to any point within the domain. The path generation time for the unconstrained RRT is significantly lower than for A* especially for simple paths as the unconstrained RRT is not limited by grid position and successor operator limitations and is able to traverse unlimited distance in one iterate. Moreover, the smoothing time is half or less for the unconstrained RRT with respect to A*. A*'s smoothing time is higher even if A* is almost optimal as the smoothing algorithm is required to process more points in A* when compared to the unconstrained RRT.

Parameter	Scenario 1	Scenario 2	Scenario 3
Path Length Prior Smoothing [–]	145%	170%	189%
Path Length After Smoothing [–]	102%	113%	123%
Computational Time RRT (s)	0.16%	9%	20%
Smoothing Computational Time for 100 Iterates (s)	43%	49%	50%

Table 2. RRT without step size constraints with respect to A*

Figure 12 confirms that the MRRT generated a path in all situations. The path length prior smoothing is primarily dependent upon the complexity of the scenario and independent of the number of seeds–per–axis. The mean path length prior smoothing for MRRT is longer when compared with the RRT without constraints for consecutive scenarios as illustrated in Table 3. This implies that although the MRRT is unconstrained in length, as the RRT and seeds are uniformly distributed, the unconstrained RRT was able to perform better especially for simpler scenarios. The MRRT interconnects different trees with unconstrained length just as the unconstrained RRT interconnects between nodes independent of length. This generates, for both algorithms and simple scenarios, high amplitude zig–zagging in the path prior smoothing as illustrated in Figure 13 (a) and (c). To interconnect trees situated evenly throughout the environment the MRRT's path length is longer than the unconstrained RRT as the latter can directly (although not optimally) connect to random nodes without exploring a high percentage of the environment. This high amplitude zig–zagging

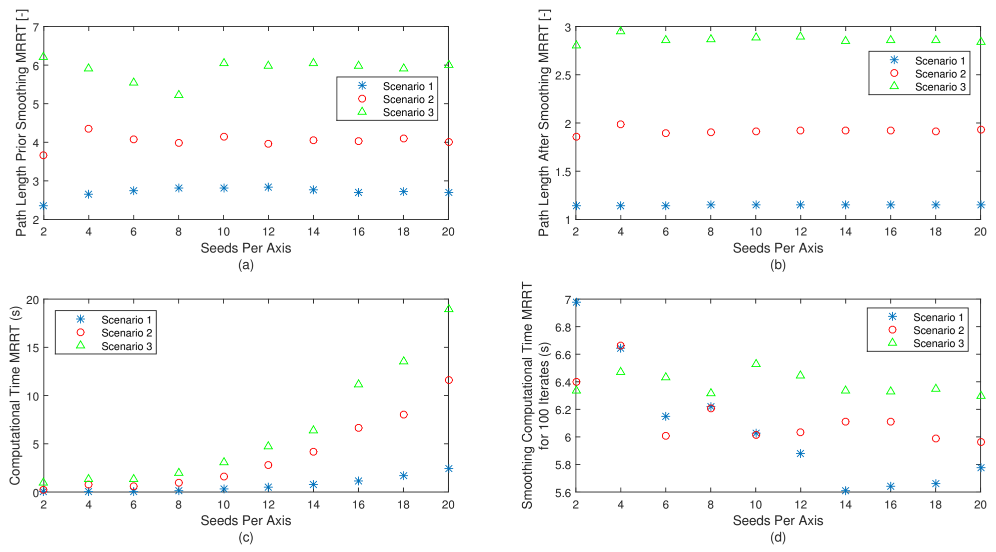

Figure 12. MRRT Results: (a) Path Length Prior Smoothing (Generic length with respect to a unity square box) (b) Path Length After Smoothing (c) Computational Time and (d) Smoothing Computational Time for 100 Iterates

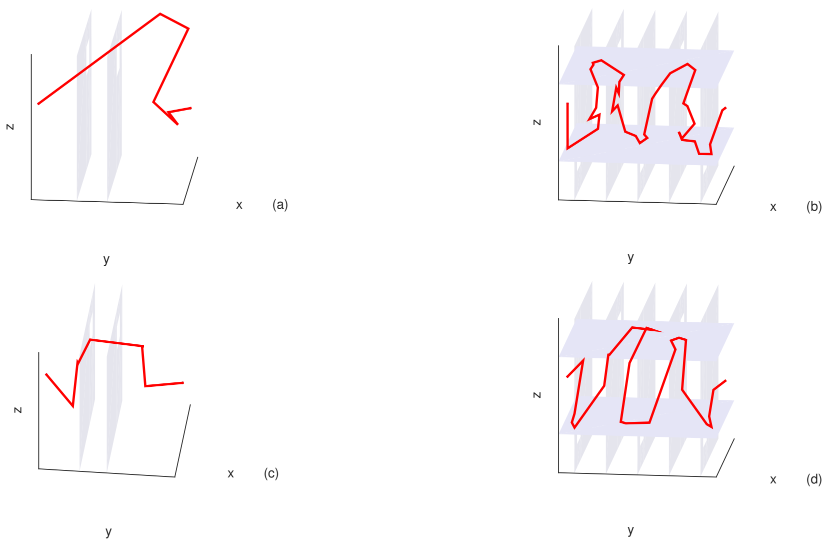

Figure 13. RRT without constraints vs MRRT: (a) RRT path without constraint for Scenario 1 (b) RRT path without constraint for Scenario 3 (c) MRRT path for Scenario 1 with 10 seeds–per–axis and (d) MRRT path for Scenario 3 with 10 seeds–per–axis

effect is attenuated with dense obstacle scenarios as interconnection between distant trees is very difficult as shown in Figure 13 (b) and (d). Also as the amount of available trees is significantly reduced due to a reduction in available space in the latter scenarios, interconnections are reduced resulting in a reduction in difference in path length for the two algorithms.

Moreover, as shown in Table 4 the MRRT mean unsmoothed path length is longer than RRT for consecutive scenarios. Even the A* algorithm outperformed MRRT in unsmoothed path length as the path length for MRRT is more than double than A* for all scenarios, as shown in Table 5.

Parameter	Scenario 1	Scenario 2	Scenario 3
Path Length Prior Smoothing [–]	148%	119%	112%
Path Length After Smoothing [–]	101%	102%	102%
Computational Time RRT (s)	7600%	600%	200%
Smoothing Computational Time for 100 Iterates (s)	168%	132%	123%

Table 3. MRRT with respect to RRT without step size constraint

Parameter	Scenario 1	Scenario 2	Scenario 3
Path Length Prior Smoothing [–]	127%	108%	107%
Path Length After Smoothing [–]	101%	102%	102%
Computational Time RRT (s)	17%	81%	38%
Smoothing Computational Time for 100 Iterates (s)	130%	128%	121%

Table 4. MRRT with respect to RRT

Parameter	Scenario 1	Scenario 2	Scenario 3
Path Length Prior Smoothing [–]	216%	203%	209%
Path Length After Smoothing [–]	104%	115%	123%
Computational Time RRT (s)	13%	55%	80%
Smoothing Computational Time for 100 Iterates (s)	64%	56%	50%

Table 5. MRRT with respect to A*

As for all previous three algorithms considered, for MRRT the smoothing algorithm was able to significantly reduce the path length. For MRRT, the mean path length after smoothing is almost equivalent to all other algorithms for all scenarios, except for the A* algorithm in the second and third scenarios in which an increase of 15% and 23% resulted as tabulated in Tables 3 to 5. Results show that the smoothing algorithm attenuates the difference in performance of all the considered algorithms. In conclusion, MRRT was outperformed in terms of path length by all approaches considered, even if conceptually it is the most complex.

For MRRT, the path planning time is primarily dependent upon the scenario complexity as for the other approaches. Moreover, the time increases as the number of seeds–per–axis increases. This is because the planner is required to expand and possibly interconnect far more trees (by a factor of 3) than in low seed–per–axis situations. The mean path generation time for MRRT is factorwise longer with respect to the unconstrained RRT for consecutive scenarios. Oppositely, the mean path generation time for MRRT is lower with respect to RRT and A* especially in simple scenarios as illustrated in Tables 4 and 5. This shows that the MRRT is computationally more efficient with respect to the standard RRT and A*. The latter two algorithms are constrained by either path points or length while MRRT tree propagation is unconstrained and explores the environment evenly from the beginning. Therefore it is unfair to compare them with MRRT. It is clear that the unconstrained conceptually simpler RRT outperformed the MRRT algorithm by multiple factors especially for simple scenarios, projecting the former for real–time applications.

The smoothing time for MRRT was longer for low seeds–per–axis situations, especially for simple scenarios. This is because with lower seeds–per–axis the trees from each seed were able to explore a larger area prior to interconnection than for higher seed–per–axis situations. This increases zig–zagging and the number of nodes to smoothen increases, effectively increasing time. This effect nullified and even superimposed (in the 2 and 4 seeds–per–axis situations) the scenario difficulty effect visualised for higher seeds–per–axis situations. The mean smoothing time for MRRT is longer with respect to both RRT algorithms as the number of nodes is higher for MRRT with respect to the other RRT algorithms therefore more interconnections are possible. Oppositely, the smoothing time is lower for MRRT with respect to A* as path points, since the A* algorithm is almost optimal and therefore interconnection between points on path segments without obstacle collision is more probable.

VI. Conclusion

This paper analysed the performance of the A*, RRT and its variants in 3D environments in view of UAV path planning applications. In accordance with literature, the A* generated almost optimal paths while the RRT always generated non–optimal paths. In fact, the difference in path length after smoothing is only slightly improved for A* while it was significantly improved for RRT. Furthermore, the A* only explored areas necessary for path construction while the RRT evenly explored the environment. The A* generated shorter paths (even after smoothing) in less time with respect to RRT. This difference in path generation time was attenuated by the elimination of the step size constraints while the path length prior smoothing was only marginally improved due to higher amplitude path zig–zagging. The evenly–distributed MRRT did not exhibit a noteworthy improvement over RRT.

In the context of UAV 3D path planning, besides obstacle avoidance, the path generation time is fundamental, especially in dynamic environments where the path needs to be continuously updated, while optimal path length is crucial in long distance situations since UAV on–board resource dependence such as fuel and battery become the bottleneck. The A*'s optimality and its low computational demand makes it a key candidate for online UAV 3D path planning with static and dynamic obstacles. Moreover, A* allows the environment to be discretised differently according to different exigencies of different parts of the scenario, making optimal use of resources. RRT and its variants are suited to generate paths efficiently in evenly distributed and focused 3D area exploration applications. RRT and its variants can perform better in terms of path construction time and length than A* if the randomly–selected nodes are intelligently selected in view of obstacle shape, position and dynamics.

The inclusion of UAV kinematic and dynamic model and sensor constraints and uncertainties together with fixed and moving obstacle position, speed and direction will confirm the applicability of the A* and RRT algorithms in real life dynamic environments.

References

[1]Sujit, P. B., and Ghose, D., "Search by UAVs with Flight Time Constraints using Game Theoretical Models," *AIAA Guidance, Navigation and Control Conference*, San Francisco, CA, 15–19 Aug. 2005, pp. 1–11.

[2]Tisdale, J., Kim, Zuwhan Kim Zuwhan and Hedrick, J., "Autonomous UAV path planning and estimation," *IEEE Robotics & Automation Magazine*, Vol. 16, No. 2, 2009, pp. 35–42.

[3]Kim, M.–H., Baik, H. and Lee, S., "Response Threshold Model Based UAV Search Planning and Task Allocation," *Journal of Intelligent & Robotic Systems*, Vol. 75, No. 3–4, 2014, pp. 625–640.

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

[5]Gros, M., Schöttl, A., and Fichter, W., "Spline and OBB–based Path–Planning for Small UAVs with the Finite Receding–Horizon Incremental–Sampling Tree Algorithm", *AIAA Guidance, Navigation and Control Conference*, Boston, MA, 19–22 Aug., 2013, pp. 1–17.

[6]Amin, J. N., Boskovic, J. D. and Mehra, R. K, "A Fast and Efficient Approach to Path Planning for Unmanned Vehicles", *AIAA Guidance, Navigation and Control Conference*, Keystone, CO, 21–24 Aug., 2006, pp. 1–9.

[7]Foo, J. L., Knutzon, J., Kalivarapu, V., Oliver, J. and Winer, E., "Path Planning of Unmanned Aerial Vehicles using B–Splines and Particle Swarm Optimization", *Journal of Aerospace Computing, Information, and Communication*, Vol. 6, No. 4, 2009, pp. 271–290.

[8]Dai, R. and Cochran, J., "Path Planning and State Estimation for Unmanned Aerial Vehicles in Hostile Environments", *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 595–601.

[9]Bollino, K. P. and Lewis, L. R., "Collision–Free Multi–UAV Optimal Path Planning and Cooperative Control for Tactical Applications", *AIAA Guidance, Navigation and Control Conference*, Honolulu, HI, 18–21 Aug., 2008, pp. 1–18.

[10]Sun, X., Cai, C. and Shen, X., "A New Cloud Model Based Human–Machine Cooperative Path Planning Method", *Journal of Intelligent & Robotic Systems*, Vol. 79, 2014, pp. 3–19.

[11]Park, S., Choi, H.–L., Roy, N., and How, J., "Learning Covariance Dynamics for Path Planning of UAV Sensors in a Large–Scale Dynamic Environment", *AIAA Guidance, Navigation and Control Conference*, Chicago, IL, 10–13 Aug., 2009, pp. 1–18.

[12]Crispin, C. and Sobester, A., "An Intelligent , Heuristic Path Planner for Multiple Agent Unmanned Air Systems", *AIAA Information Technology at Aerospace*, Kissemmee, FL, 5–9 Jan., 2015, pp. 1–13.

[13]Boskovic, J. D., Knoebel, N., Moshtagh, N. and Larson, G.L., "Collaborative Mission Planning & Autonomous Control Technology (CoMPACT) System Employing Swarms of UAVs", *AIAA Guidance, Navigation and Control Conference*, Chicago, IL, 10–13 Aug., 2009, pp. 1–24.

[14]Wu, P. P., Campbell, D. and Merz, T., "Multi–Objective Four–Dimensional Vehicle Motion Planning in Large Dynamic Environments", *IEEE Transactions on Systems, Man and Cybernetics– Part B Cybernatics*, Vol. 41, No. 3, Jun. 2011, pp. 621–634.

[15]Luke, S., Sullivan, K. and Balan, G. C., "Tunably Decentralized Algorithms for Cooperative Target Observation", George Mason University, GMU–CS–TR–2004–1, 2004.

[16]Sujit, P. B. and Ghose, D., "Two–agent cooperative search using game models with endurance–time constraints", *Engineering Optimization*, Vol. 42, No. 7, Jul. 2010, pp. 617–639.

[17]Ryan, A. and Hedrick, J. K., "A mode–switching path planner for UAV–assisted search and rescue", *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Spain, 12–15 Aug. 2005, pp. 1471–1476.

[18]Tseng, F. H., Liang, T. T. and Lee, C. H. Chou, L. D. and Chao, H., "A Star Search Algorithm for Civil UAV Path Planning with 3G Communication", *10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH–MSP)*, Kitakyushu, Japan, 27–29 Aug. 2014, pp. 942–945.

[19]Sathyaraj, M. B., Jain, L. C., Finn, A. and Drake, S., "A Multiple UAVs path planning algorithms: a comparative study", *Fuzzy Optimization and Decision Making*, Vol. 7, No. 3, 2008, pp. 257–267.

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

[21]Stentz, A., "The focused D* algorithm for real–time replanning.", *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada, 20–25, Aug. 1995, pp. 16521659.

[22]Koenig, S. and Likhachev, M., "D* Lite", *Proceeding of the AIAA Conference on Artificial Intelligence*, Indianapolis, IN, 7–10, Jul. 2002, pp. 476–483.

[23]Koenig, S. and Likhachev, M., "Incremental A*", *Proceeding of the Natural Information Processing Systems*, Vancouver, British Columbia, Canada, 3–8 December, 2001.

[24]Lavalle S. M. and Kuffner J. J., "Randomized kinodynamic planning", *International Journal of Robotics Research*, Vol. 20, No. 3, pp. 378–400, 2001.

[25]Karaman S. and Frazzoli E., "Sampling–Based Algorithms for Optimal Motion Planning", *International Journal of Robotics Research*, Vol. 30, No. 7, pp. 846–894, 2011.

[26]Hart, P. E., Nilsson, N. J. and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 3, pp. 100–107, 1968.

[27]Gonzàlez, D., P*èrez*, J., Milanès, V., and Nashashibi, F., "A Review of Motion Planning Techniques for Automated Vehicles", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 4, pp. 1135–1145, 2016.

[28]Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs", *Math. 1*, Pp. 269–271, 1959.

[29]Ghandi, S. and Masehian, E., "Review and taxonomies of assembly and disassembly path planning problems and approaches", *CAD Computer Aided Design*, Vol. 67–68, No. October, pp. 58–86, 2015.

[30]Li, Q., Zeng, Z., Yang, B. and Zhang, T. "Hierarchical route planning based on taxi gps–trajectories", *17th International Conference on Geoinformatics*, Fairfax, VA, 12–14 Aug. 2009, pp. 1–5.

[31]Kala, R. and Warwick, K. "Multi–level planning for semi–autonomous vehicles in traffic scenarios based on separation maximization", *Journal of Intelligent and Robotic Systems*, Vol. 72, No. 3–4, pp. 559–590, 2015.

[32]Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J. and Satterfield, B. "Little ben: The ben franklin racing team' s entry in the 2007 darpa urban challenge", *Journal of Field Robotics*, Vol. 25, No. 9, pp. 598–614, 2008.

[33]Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Rein– holtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D. "Odin: Team victortango' s entry in the darpa urban challenge", *Journal of Field Robotics*, Vol. 25, No. 8, pp. 467–492, 2008.

[34]Ferguson, D. and Stentz, A. "Using interpolation to improve path planning: The field D* algorithm", *Journal of Field Robotics*, Vol. 23, No. 2, pp. 79–101, 2006. *Journal of Field Robotics*, Vol. 25, No. 8, pp. 467–492, 2008.

[35]Daniel, K., Nash, A., Koenig, S. and Felner, A. "Theta*: Any–angle path planning on grids", *Journal of Artificial Intelligence*, Vol. 39, pp. 533–579, 2010.

[36]Nash, A., Koenig, S. and Likhachev, M. "Incremental Phi*: Incremental Any–Angle Path Planning on Grids", *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1824–1830, 2009.

[37]Mitchell, J. and Papadimitriou, C. "The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision", *Journal of the Association for Computing Machinery (ACM)*, Vol. 38, No. 1, pp. 18–73, 1991.

[38]Trovato, K. and Dorst, L. "Differential A*", *IEEE Transaction on Knowledge and Data Engineering*, Vol. 14, No. 6, pp. 1218-1229, 2002.

[39]Nash, A. and Koenig, S. "Any–Angle Path Planning", *Artificial Intelligence (AI) Magazine*, Vol. 34, No. 4, pp. 9–31, 2013.

[40]Yap, P., Burch, N., Holte, R. and Schaeffer, J. "Block A*: Database–driven search with applications in any–angle path–planning", *Proceedings of the Conference of Artificial Intelligence (AAAI)*, 2011.

[41]Harabor, D. and Grastien, A. “Online Graph Pruning for Path finding On Grid Maps”, *Proceedings of the Conference of Artificial Intelligence (AAAI)*, 2011.

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

[43]Likhachev, M., Ferguson, D., Gordon, G., Stentz, A. and Thrun, S., “Anytime search in dynamic graph”, *Artificial Intelligence*, Vol. 172, No. 14, pp. 1613-1643, 2008.

[44]Ainea, S. and Likhachevb, M. “Truncated incremental search”, *Artificial Intelligence*, Vol. 234, pp. 49–77, 2016.

[45]Gochev, K., Safonova, A. and Likhachev, M. “Anytime Tree–restoring A* Graph Search”, *Proceedings of the Seventh Annual Symposium on Combinatorial Search SoCS*, Prague, Czech Republic, 15–17 Aug. 2014, pp. 80–88.

[46]Sun, X., Yeoh, W. and Koenig, S. “Dynamic fringe–saving A*”, *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent systems (AAMAS)*, Budapest, Hungary, May 2009, pp. 891–898.

[47]Sun, X., Koenig, S. and Yeoh, W. “Generalized adaptive A*”, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent systems (AAMAS)*, Estoril, Portugal, May 2008, pp. 469–476.

[48]Hernández, C., Sun, X., Koenig, S. and Meseguer, P. “Tree adaptive A*”, *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent systems (AAMAS)*, Taipei, Taiwan, May 2011, pp. 123–130.

[49]Short, A., Pan, Z., Larkin, Z. and van Duin, S. “Recent Progress on Sampling Based Dynamic Motion Planning Algorithms”, *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Alberta, Canada, Jul. 2016, pp. 1305–1311.

[50]Kavraki, L. E., *Š*vestka, P., Latombe, J. C. and Overmars, M. H. “Probabilistic roadmaps for path planning in high–dimensional configuration spaces”, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 14, pp. 566–580, 1996.

[51]LaValle S. M. “Probabilistic roadmaps for path planning in high–dimensional configuration spaces”, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 14, pp. 566–580, 1996.

[52]Bohlin, R. and Kavraki, L. E. “Path planning using lazy PRM”, *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 521–528, 2000.

[53]Klasing, K., Wollherr, D. and Buss, M. “Cell–based probabilistic roadmaps (cprm) for efficient path planning in large environments”, *Proceedings of International Conference on Advanced Robotics*, 2007.

[54]Leven, K. P. and Hutchinson, S. “Toward real–time path planning in changing environments”, *Algorithmic and Computational Robotics: New Directions: The Fourth International Workshop on the Algorithmic Foundations of Robotics*, pp. 363–376, 2000.

[55]Pomarlan, M. and Sucan, I. A. “Motion planning for manipulators in dynamically changing environments using real–time mapping of free workspace”, *IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, Hungary, 19–21 Nov. 2013, pp. 483 487.

[56]Belghith, K., Kabanza, F., Hartman, L. and Nkambou, R. “Any–time dynamic path–planning with flexible probabilistic roadmaps”, *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 15–22 May. 2013, pp. 2372–2377.

[57]Yang, B. Y. and Brock, O. “Elastic roadmaps motion generation for autonomous mobile manipulation”, *Autonomous Robots*, Vol. 28, No. 1, pp. 113–130, 2010.

[58]Gayle, A., Sud, M., Lin, C. and Manocha, D. “Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments”, *International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, 29 Oct–02 Nov 2007, pp. 3777–3783.

[59]LaValle, S. M. and Kuffner, J. J. “Randomized kinodynamic planning”, *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, 10–15 May 1999, pp. 473–479.

[60]Devaurs, D., Siméon, T. and Cortés, J. “Optimal Path Planning in Complex Cost Spaces With Sampling–Based Algorithms”, *IEEE Transactions on Automation Science and Engineering*, Institute of Electrical and Electronics Engineers, 2015.

[61]Geraerts, R. and Overmars, M. “Creating high–quality paths for motion planning”, *International Journal of Robotics Research*, Vol. 26, No. 8, pp. 845–863, 2007.

[62]Tsardoulias, E. G., Iliakopoulou, A., Kargakos, A. and Petrou, L. “A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density”, *Journal of Intelligent and Robotic Systems*, pp. 1–30, 2016.

[63]Kuffner, J. J. and LaValle, S. M. “Rrt–connect: An efficient approach to single–query path planning”, *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, 20–24 Apr. 2000, Vol. 2, pp. 521–528.

[64]Martin, S. R., Wright, S. E. and Sheppard, J. W. “Offline and Online Evolutionary Bi–Directional RRT Algorithms for Efficient Re–Planning in Dynamic Environments”, *Proceedings of the IEEE International Conference on Automation Science and Engineering*, Scottsdale, AZ, 22–25 Sep. 2007, Vol. 2, pp. 1131–1136.

[65]Bruce, J. and Veloso, M. “Real–time Randomized Path Planning for Robot Navigation”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 30 Sep.–4 Oct. 2002, Vol. 3, pp. 2383–2388.

[66]Karaman S. and Frazzoli E., “Optimal kinodynamic motion planning using incremental sampling–based methods”, *Proceedings of the 49th IEEE Conference on Decision and Control*, Atlanta, GA, 15–17 Dec. 2010, pp. 7681–7687.

[67]Devaurs, D., Siméon, T. and Cortés, J. “Enhancing the transition–based RRT to deal with complex cost spaces”, *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 15–22 May. 2013, pp. 4561–4568.

[68]Ferguson, D. and Stentz, A. “Anytime RRTs”, *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 9–16 Oct. 2006, pp. 5369–5375.

[69]Zhu, Q., Wu, Y., Wu, G. and Wang, X. “An improved anytime RRTs algorithm”, *International Conference on Artificial Intelligence and Computational Intelligence*, Shanghai, China, 7–8 Nov. 2009, pp. 268–272.

[70]Abbasi–Yadkori, Y., Modayil, J. and C. Szepesvári, J. “Extending rapidly–exploring random trees for asymptotically optimal anytime motion planning”, *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Shanghai, China, 18–22 Oct. 2010, pp. 127–132.

[71]Alterovitz, R., Patil, S. and Derbakova, A. “Rapidly–exploring roadmaps: weighing exploration vs. refinement in optimal motion planning”, *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 9–13 May 2011, pp. 3706–3712.

[72]Luna, R., Şucan, I., Moll, M. and Kavraki, L. “Anytime solution optimization for sampling–based motion planning”, *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 15–22 May. 2013, pp. 5068–5074.

[73]Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. and Fiore, G. “Real–Time Motion Planning With Applications to Autonomous Urban Driving”, *IEEE Transactions on Control Systems Technology*, Vol. 17, No. 5, pp. 1105–1118, 2009.

[74]Braid, D., Broggi, A. and Schmiedel G. “The TerraMax autonomous vehicle”, *Journal of Field Robotics*, Vol. 23, No. 5, pp. 693–708, 2006.

[75]Ryu, R.–H., Ogay, D., Bulavintsev, D., Kim, H. and Park, J.–S. “Development and Experiences of an Autonomous Vehicle for High–Speed Navigation and Obstacle Avoidance”, *Frontiers of Intelligent Autonomous Systems*, Springer 2013, pp. 105–116.

[76]Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan and M., Muhammad, M. S. “DRRT*–SMART: a rapid convergence implementation of RRT*”, *Journal of Advanced Robotic Systems*, Vol. 10, No. 7, pp. 1–12.

[77]Ferguson, D., Karla, N. and Stentz, A. “Replanning with RRT”, *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 15–19 May. 2006, pp. 1243–1248.

[78]Ferguson, D. and Stentz, A. “Anytime, dynamic planning in high–dimensional search spaces”, *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 19–23 May. 2007, pp. 1310–1315.

[79]Frazzoli, E., Dahleh, M. A. and Feron, E. “Real–time motion planning for agile autonomous vehicles”, *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[80]Zucker, M., Kuffner, J. and Branicky, M. “Multipartite rrts for rapid replanning in dynamic environments”, *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 19–23 May. 2007, pp. 1603–1609.

[81]Benenson, R., Petti, S., Fraichard, T. and Parent, M. “Integrating perception and planning for autonomous navigation of urban vehicles”, *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 9–15 Oct. 2006, pp. 98–104.

[82]Bekris, K. E. and Kavraki L. E. “Greedy but safe replanning under kinodynamic constraints”, *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 19–23 May. 2007, pp. 704–710.

[83]Li, T. Y. and Shie, Y. C. “An incremental learning approach to motion planning with roadmap management”, *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, 11–15 May. 2002, pp. 3411–3416.

[84]Gayle, R., Klingler, K. R. and Xavier, P. G. ”Lazy reconfiguration forest (lrf)–an approach for motion planning with multiple tasks in dynamic environments”, *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 19–23 May. 2007, pp. 1316–1323.

[85]Guitton, J., Farges, J. L. and Chatila, R. “Cell–RRT: Decomposing the environment for better plan”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 10–15 Oct. 2009, pp. 5776–5781.

[86]Shang, W., Liu, J., Ning, R. and Liu, M. “Computational path planner for product assembly in complex environments”, *Chinese Journal of Mechanical Engineering*, Springer 2013, Vol. 26, No. 2, pp. 282–292.

[87]Otte, M. and Frazzoli, E. “RRT^X: Asymptotically Optimal Single–Query Sampling–Based Motion Planning with Quick Replanning”, *The International Journal of Robotics Research*, Vol. 29, No. 7, pp. 797–822.

[88]Labakhua, L., Nunes, U., Rodrigues, R. and Leite, F. S. “Smooth trajectory planning for fully automated passengers vehicles: spline and clothoid based methods and its simulation”, *Informatics in Control Automation and Robotics*, Springer 2008, pp. 169–182.

[89]Reeds, J. and Shepp, L. “Optimal paths for a car that goes both forwards and backwards”, *Pacific Journal of Mathematics*, Vol. 145, No. 2, pp. 367–393, 1996.

[90]Horst, J. and Barbera, A. “Trajectory generation for an on–road autonomous vehicle”, *Defense and Security Symposium*, International Society for Optics and Photonics, NIST Interagency/Internal Report (NISTIR), Report No. 7262, June 2006.

[91]Brezak, M. and Petrovic, I. “Real–time approximation of clothoids with bounded error for path planning applications”, *IEEE Transactions on Robotics*, Vol. 30, No. 2, pp. 507515, 2014.

[92]Tsai, Y., Lee, C., Lin, C. and Huang, C. “Development of Flight Path Planning for Multirotor Aerial Vehicles”, *Aerospace 2015*, Vol. 2, pp. 171–188, 2015.

[93]Roa, D. and Williams, S. “Large–scale path planning for Underwater Gliders in ocean currents”, *Australasian Conference on Robotics and Automation (ACRA)*, Sydney, Australia, December 2–4 2009, pp. 1–9.

[94]Simmons, R. and Urmson, C. “Approaches for heuristically biasing RRT growth”, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1178–1183, 2003.

[95]Clifton, M., Paul, G., Kwok, N., Liu, D. and Wang, D. “Evaluating Performance of Multiple RRTs”, *IEEE conference on Mechatronic and Embedded Systems and Application*, Bejing, China, 12–15 Oct. 2009, pp. 564–569.

[96]Paul, G. “Multiple Rapidly–exploring Random Tree (RRT)”, *MATHWORKS*, [Online Database], https://www.mathworks.com/matlabcentral/fileexchange/21443-multiple-rapidly-exploring-random-tree--rrt-?requestedDomain=www.mathworks.com, [retreived 30 October, 2016].