Comparison of A* and RRT in real–time 3D path planning of UAVs

Zammit, Christian; van Kampen, Erik-jan

DOI
10.2514/6.2020-0861

Publication date
2020

Document Version
Final published version

Published in
AIAA Scitech 2020 Forum

AIAA SciTech Forum
6-10 January 2020, Orlando, FL
AIAA Scitech 2020 Forum

10.2514/6.2020-0861

Comparison of A* and RRT in real–time 3D path planning of UAVs

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 military, industrial and commercial applications. Such applications require faultless autonomous systems to coordinate, guide, navigate and control different UAVs of different sizes, designed for different purposes with different capabilities. In this regard, different path planning algorithms were developed to ensure that UAVs are supplied with collision–free path paramount to which are the A* and the RRT algorithms, a graph–based and a sampling–based algorithm respectively. Such algorithms shall ideally operate in real–time to furnish the UAV navigation system with real–time, valid, obstacle–free paths in view of changes in the environment or other external or user–defined restrictions. Owing this need, in this paper a real–time platform to assess the performance of the A* and RRT algorithm with an associated smoothing algorithm was developed and tested using 3, 3D obstacle environment with different complexities. The salient user–defined, system–defined and internal constants were independently considered and their effect on performance assessed. Results showed that the A* outperformed the RRT algorithm in both path length and computational time for all scenarios considered with difference increasing with scenario complexity. But, both algorithms can be utilised if the associated parameters are attentively chosen based on the scenario the UAV will operate as both algorithm reached a 100% success rate for all scenario at specific parameter assignments.

I. Introduction

Unmanned Aerial Vehicle (UAVs) are potential candidates for a wide range of applications in both civil and military setups. In these scenarios, different UAVs requires varying levels of autonomy, reliability and efficiency based on the assigned task. To reach a goal or set of goals, UAVs are equipped with sensory, processing and actuator systems with different accuracy, redundancy, preciseness, latency, reliability and computational power.

Real-time, efficient and reliable paths are fundamental to ensure that the UAV autonomously reaches the goal safely and in due time. Path planning is the process of automatically generating feasible and optimal 2D[1,2] or 3D[3,4] paths. Different UAVs have different levels of path planning autonomy varying from solely human-controlled[5] and shared human-controlled[6] systems to fully autonomous goal-oriented systems[7] designed for different applications. Some of these applications include: agricultural remote sensing,[8] ground vehicle tracking,[9] traffic surveillance,[10] package delivery,[11,12] medicinal delivery in remote areas[13] and ambulance drone.[14] Moreover, UAVs can be utilised in situations where the mission is too difficult or too dangerous for human pilots such as monitor critical structures in natural disasters, search and rescue and monitor weather inside a storm.[15]

The path planning algorithms utilise sensory, processing and actuator systems to generate paths in view of different kinematic, dynamic[16,17] and environmental[18,19] time–varying constraints. Once a path is generated through the path planning algorithm, a path following algorithm will generate the control parameters for the UAV to follow the generated path. Although these control parameters can be generated offline, real-time

*Ph. D. Candidate, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands.
†Senior Lecturer, Electrical Engineering, Gozo Campus, Malta College of Arts, Science and Technology (MCAST), Malta and AIAA Member
‡Assistant Professor, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands, and AIAA Member.

path planning allows the path following algorithm to amend control instructions in view of unpredictable and/or uncertain model and environmental changes. Currently, general–purpose and dedicated UAV systems incorporate advanced control algorithms that can allow UAVs to manoeuvre in cluttered environments if the control algorithm is provided with accurate, timely and efficient real-time attitude and directional information. This goal–driven, autonomous UAV can be realised via a real-time path planning algorithm governed by the already available path following and control state-of-the-art systems.

Even in indoor applications, UAVs are expected to operate in a time-varying environment even if the start and goal positions are identical or interchange continuously. A UAV may encounter moving unpredictable obstacles, such as persons, door and window openings and other UAVs as well as internal unpredictable events such as fuel limitations, loss of movement in 1 or more Degrees of Freedom (DoF) and loss of partial or complete sensory information. In these real–life situations the control, path following and planning algorithms must operate safely and efficiently irrespective of any shortcomings in the sensory and actuator systems.[20] To cater for these situations besides a real–time control algorithm, a real–time path planner is a must.

The aim of this paper is to assess the appropriateness for real–time 3D UAV path planning of graph–based and sampling–based path planning approaches. An extensive literature review of the state-of-the-art path planning algorithms presented in[21] concluded that A* and RRT are the two most utilised graph–based and sampling–based path planning methods, respectively. Further analysis in[21,22] showed that these algorithms and their variants are key candidates for 3D UAV path planning. Therefore, these two algorithms will be assessed for performance mainly in terms of computational time, success rate and path length throughout this paper in view of real-time application.

The paper will be organised as follows. Section II will present the state-of-the-art in real-time path planning. Section III provides a brief resume of the considered path planning algorithms (A* and RRT) and the smoothing algorithm defined in depth in our previous work.[21,22] Section IV will define the theoretical aspect of the developed algorithm designed to assess the appropriateness of the considered path planning algorithms in real–time applications. Section V will define the experimental scenario with the associated arbitrary–defined parameters. The following section (Section VI), will present, analyse and assess the results in view of 3D UAV path planning in real–time. The paper will conclude by Section VII which based on the benefits and shortcomings will rate the appropriateness of the implemented algorithm for 3D UAV path planning in real–time.

II. Real–time path planning literature review

A. Introduction

Real–time path planning is considered as a desirable feature,[23] a requirement[24] and paramount[25] for real-time autonomous manoeuvring of vehicles let alone UAVs in real, dynamic environments. Real-time path planners are requested to generate paths in the presence of other cooperating or enemy UAVs, unexpected UAV damage, altering model constraints and definitions, and in view of uncertainties in the environment in which they operate.[3,23,24] Furthermore, the path planner must make optimal use of available resources such as computational power and fuel.[26]

A path planner is considered to be real–time if the time required to generate a path is smaller than the time to traverse the path.[3,27,28] For Short *et. al.*[29] a realistic path planner must react in synchronisation with information update from the sensory systems. Furthermore, such path planner must generate a path even with restricted global information.[20]

As computational time is the bottleneck of real–time path planning, Karaman and Frazzoli,[30] remarked that computational time per iterate shall be sub–bounded. In certain iterates, longer computational times may be required due to fewer path solutions. if not tackled these will increase the overall computational time making the path planning algorithm unsuitable for real–time path planning. Sub–bounds will attenuate this situation even if a solution is not found in that iterate, but a minor movement on the previously generated path may yield a path possibly in lesser time.

Real-time path planning of UAVs require high fidelity modelling of the environment. Simultaneous Localisation and Mapping (SLAM) algorithms can be utilised to generate maps for realistic environments to facilitate real–time path planning. Such algorithms can be utilised to mitigate in absence or partial absence of GPS information, discontinuity of sensor information and noise.[31]

This literature review will be segmented into three main path planning categories: Optimisation algo-

rithms, graph–based methods and sampling–based methods. Their application in 3D real–time path planning will be analysed and assessed.

B. Optimisation Algorithms

In both real-time and offline applications there is no guarantee of convergence to the goal let alone in a predetermined time either because no path exists, the environment is not enough known or the path planning algorithm intrinsically cannot generate the path in the particular situation.[26] In such situations, a trade-off between computational time and optimality was considered by Frazzoli[26] for a finite–state automation method to compute trajectories for multiple UAVs in safety–critical, high performance vehicles with complex dynamics.

Disturbances from external torques initiating through for example wind shears, sensor inaccuracies and parameter uncertainties will increase further the path planning and following of complex systems such as UAVs.[25] Real–time applications of optimal control theory showed that feedback control will enhance performance in such complex nonlinear systems.[32,33] Furthermore, through sensor fusion, the accuracy and responsiveness of the sensing system will be improved.[25] Gong *et. al.*[33] and Bollino *et. al.*[25] utilised a pseudospectral method for optimal control and path planning of complex systems, respectively. Simulation results of this method show that although pseudospectral methods are mainly utilised for path following providing optimal control instructions, they can be utilised for autonomous path planning.[25]

A single optimisation method cannot simultaneously guarantee target tracking and obstacle avoidance,[28] especially in 3D UAV path planning environments. Yao *et. al.*[28] proposed a combination of improved Lyapunov Guidance Vector Field (LGVF), the Interfered Fluid Dynamical System (IFDS) and the strategy of varying receding–horizon optimisation based on Model Predictive Control (MPC) to generate 3D paths for a UAV in dynamic environments under constraints. This hybrid algorithm was proposed since although MPC were successfully applied to generate suboptimal paths in real–time[34,35] and to generate paths in 2D scenarios, the computational efficiency and smoothness will deteriorate in 3D environments.[28]

Furthermore, Roberge *et. al.*,[36] compared Particle Swarm Optimisation and Genetic algorithms in real-time for the automatic path planning of fixed–wing UAVs in complex 3D environments. Both algorithms generated feasible and quasi–optimal trajectories in view of vehicle dynamics. Execution time was reduced through single-data multiple-data on an 8 core processor. Although both algorithms generated a path with 10s (a preset path planning time), Genetic Algorithms produced better trajectories. Roberge *et. al.*[36] remarked that both algorithms can generate a path for cruising the fastest fixed wing UAV available at the time of writing.

In dynamic real-world populated environments and considering the disturbances and eventualities mentioned above, 10s is too long to react and generate a new non-colliding path. Solving complex optimisation problems arising from these scenarios will result in high computational load.[37]

C. Graph–based Methods

Real–time path planning of complex dynamic environments still remains a challenge even for 2D environment, let alone 3D.[38] Kuwata *et. al.*[38] remarked that it is very difficult to model nonlinear dynamics especially for demanding manoeuvres when using generic graph–based path planning methods. This is because generic graph–based methods such as the original A* assume the environment to be static.[39] Similarly, Singh *et. al.*,[31] remarked that although graph–based methods are effective in configured environments, it is unsuitable for real–time path planning in large or complex environments. Local approaches of such graph–based methods can stall in local minima.[31]

Although as remarked by Kuwata *et. al.*,[38] graph–based methods are not optimal for real-time path planning, the differential A*,[40] based on the A* algorithm (a graph–based method), was developed. More efficient results in the majority of cases were achieved when compared to A*, proposing the algorithm as a candidate for real–time dynamic, re-planning.[40]

Fernandes *et. al.*,[39] extended the A* algorithm in cell decomposition, considering both position and orientation in the computation of the path. This approach made it possible to reduce the computational time while maintaining the same configuration space.[39]

A parallel non–deterministic adaption of the Dijkstra algorithm, a pioneer graph–based method, was applied to generate an energy efficient, global re–planner for real-time UAV rescue operations in dynamic

environments with a range of $200m^2$ utilising a map discretization of $1m^2$.[41] Results show tens of multiple times improvement over the sequential Dijkstra algorithm with an average path cost error of smaller 1.2%.[41]

D. Sampling–based Methods

Sampling–based methods are applicable to general dynamical models, their incremental nature makes it inherent for use in real–time applications whilst guaranteeing a solution and do not require enumeration of constraints allowing trajectory–wise checking of complex constraints.[42,43]

Advancements in sampling-based methods have proposed these algorithms for real-time path planning in dynamic and unknown environments.[29] Re-planning is essential in these situations as the environment is only partially known at one point in time, revealing more detail in the direction of the goal in every vehicle movement. This can also create situations, in which a previous non–colliding path may lead to a collision with more details in the environment. Therefore, the planner must react in real–time to mitigate with these situations whilst the UAV is moving. According to Kunz *et. al.*,[44] this reaction time shall not exceed 200ms to mimic human reaction.

Since according to Kuwata *et. al.*[43] the standard RRT is not able to generate safe and feasible paths in the presence of uncertainty in real–time for 2D applications, the latter proposed a closed-loop prediction in the framework of RRT, with a low–level Proportional–Integral speed controller and a pure pursuit steering controller to manoeuvre the vehicle based on real–time path planning information. Real–time execution requires reusing information from previous.[45,46] Otherwise only a sparse tree will be created when compared to the reuse approach in which computational resources are used to add new improved branches to the existing tree.[43] A non–colliding path is retained as long as possible to firstly limit *"no path"* situations and secondly to grant the necessary time for the path planning algorithm to generate efficient trajectories.[38] This Closed loop RRT technique shows that real-time path planning is the bottleneck even in 2D environments.

E. Conclusion

This review first highlighted that 3D path planning algorithms shall successfully and efficiently operate in real–time with restricted computational power, lack of onboard resources, sensor low responsiveness and inaccuracy and environmental uncertainties. Optimisation algorithms are prospective candidates for 3D UAV path planning in real–time. Although results are promising, this approach requires knowledge of UAV dynamics and non–linearities. Graph–based methods are intrinsically designed for static environments although amendments to the basic algorithms can extend their application to real–time situations due to their relative computational simplicity. Sampling–based methods and their adaptions are also worth considering especially if re–usability of previous non–colliding paths or branches are retained. Furthermore, hybrid approaches combining different algorithms with different strongholds can be utilised to mitigate inherent shortcomings of one of the discussed three approaches.

III. The A*, RRT and Smoothing Algorithms

A. Introduction

Graph–based methods divide the working space into an occupancy grid with obstacles defined as inaccessible grid points.[42,47] Such methods do not offer a guarantee of solution.[48] Oppositely, sampling–based methods create a path by connecting unevenly selected points from the configuration space.[29,42] The A* and RRT are the two most utilised algorithms for the graph–based and sampling–based methods, respectively. To optimise the path a smoothing algorithm was developed and applied to both algorithms. in our previous work.[21,22]

B. The A* Algorithm

The standard A* algorithm constructs an optimal path based on an evaluation function $f(n)$ that calculates the actual cost of an optimal path constrained to pass through n, from a point x_{init} to the goal node of n, $x_{goal} \in \mathbb{R}^M$.[49,50] 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: $\in \mathbb{R}^M \rightarrow \mathbb{R}$:

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

C. The RRT Algorithm

The RRT algorithm grows trees of feasible trajectories by randomly planting a number of seeds. Seeds are only considered if they lie on an obstacle free point. A point a predefined distance from the nearest seed is selected if the direct path to the latter does not collide with an obstacle. Ultimately a tree that interconnects the start and goal points will define a feasible path.[51–53] Paths generated by RRT are not optimal.[54,55]

D. The Smoothing Algorithm

The smoothing algorithm randomly selects two path points and consequently defines two points on the lines connecting these path points with their respective next path point. If an interconnection is possible without collision with obstacles then intermediate points between these two path points are eliminated.

E. Conclusion

Reference is made to our previous work[21] for a more in depth understanding of the working principle for the A*, RRT and smoothing algorithms. In addition, a detailed literature review of the current state-of-the-art in graph–based and sampling–based methods is available.

IV. The Real-time algorithm

A. Introduction

Both path planning algorithms were successfully implemented in static offline applications. Furthermore, the RRT without step-size constraints and the Multiple Rapidly–Exploring Random Tree (MRRT) algorithms were also considered in our previous works.[21,22] The characteristics, strongholds and shortcomings of these algorithms were identified, assessed and analysed through the use of different experimental scenarios in view of 3D UAV path planning.[21,22]

As was concluded in Section II, real–time path planning is almost a must for a UAV to autonomously reach a goal especially in the presence of static and dynamic obstacles.[23–25] Therefore, a testing platform was set up to assess and possibly improve the performance of the two most utilised graph–based and sampling–based methods.

B. Theoretical Aspect

In real-life path planning, the UAV path generation system must generate and/or update the existing path to goal ideally every instance say few milliseconds, irrespective of a changes in the UAV position and obstacles. Such approach demands high update rate. Such rate is beyond the computational power available onboard state-of-the-art UAVs. Therefore, the proposed real–time path planning algorithm only updates at predetermined time intervals derived from the nominal speed of the UAV. It is assumed that the obstacle positions and size remain constant. Although it is assumed that the UAV actuator system will move the UAV a predetermined distance defined by the preset time step in a certain direction set by the generated path, if variations exists due to internal (speed controller) or external factors (weather) the real–time path planning system can make use of updated positional information when updating the path in the subsequent iterate.

Figure 1 graphically illustrates the real–time path planning concept. The UAV sensory system have a limited field–of–view (FOV) (green–dotted circle) into which prospective new intermediate goal point positions can be selected. It is assumed that such area is known with certainty with respect to obstacles (Olive green dots). In this illustration it is assumed that the UAV is equipped with a 360^o FOV sensory system but the principle can be applied to smaller FOV sensory systems as usually the goal position is within 180^o of the current UAV heading.

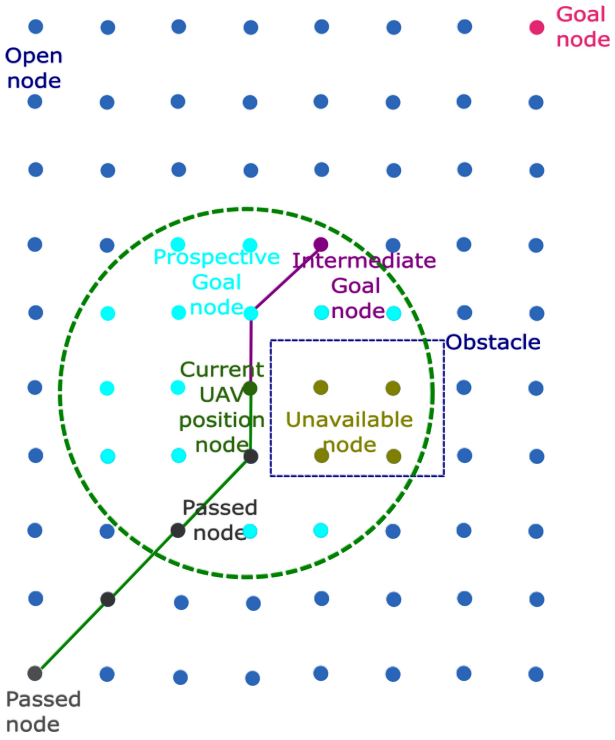

Figure 1. Real–time path planning with finite limited look–ahead distance

1. *Parameter Definition and Initiation*

Firstly the real–time initiates by defining the start (*start*) and goal (*goal*) positions and the considered resolution (*res*). In RRT since the randomly generated point can reside anywhere with an environmental space, the equivalent of resolution is the step size which denotes the distance that the current path point can move in the direction of the generated path point. This will be denoted by d_{step_RRT}. For clarity *res* will be considered in the definition of other parameters.

In real–time the UAV path planning system has a finite time to generate a feasible path to the final goal as otherwise the UAV must either stop intermittently in mid-air (if a quadrotor is considered) or the path planning algorithm becomes unfeasible. As cited in Section II A, a real–time path planner must generate a path segment in less than the time to traverse it.[3,27,28] For a real–time path planner stopping in mid–air is not an option. Therefore the maximum time between iterates need to be defined based on the following two assumptions. The UAV moved at a constant nominal speed of v_{UAV} in (km/hr) in a cubic environmental space of d_{env_space} distance on a 3D axis. Based on this work space the time to generate a path between iterates $t_{iterate\_max}$ is:

$$t_{iterate\_max}(s) = \frac{60 \times 60 \times d_{env\_space}}{(res - 1) \times v_{UAV}} \tag{2}$$

Another constant parameter closely related to $t_{iterate\_max}$ is the distance covered by the UAV in every iterate (d_{s_step}). This value is a function of the UAV speed and assumes that actuator systems are linear and no external factors such weather are effecting the UAV. Based on the above rationale the distance covered in $t_{iterate\_max}$ shall be $\leq d_{s\_step}$. This modular unit value was also arbitrary chosen although it can be varied based on the fidelity and confidence of the UAV sensory and actuator systems and environmental model.

Besides the maximum time to generate a path segment between iterates $t_{iterate\_max}$, the maximum time allocated to reach the goal point from the start position $t_{path\_gen\_max}$ was arbitrary set. This upper limit is included as situations can arise in which the UAV will venture around obstacles without actually getting closer to the goal. Moreover, UAV fuel autonomy is finite. This value shall denote double the time required to traverse the diagonal distance between two extreme points of the environmental space.

Moreover, d_{int_goal} denotes the distance between the current UAV position and the prospective intermediate goal point. This arbitrary value is a function of the range of the sensory system which shall ensure

that in d_{int_goal} all obstacles are know with certainty. In certain instances the resultant intermediate goal position may reside on an obstacle. In such situations, a new intermediate goal point must be defined. As the maximum look–ahead distance is defined by d_{int_goal}, the new intermediate goal point must reside nearer to the current UAV position. Therefore a distance reduction factor, $d_{factor} < 1$ (set at 0.9) was defined. Finally, *lim* defines the 3D boundaries of the working environment which may vary due to the shifting introduced by the A* ripple reduction algorithm.

After these constants are defined the A* ripple reduction algorithm is applied in case the A* algorithm is under review. An in depth definition of this algorithm is available in our previous work.[22] The new 3D limits introduced by this algorithm are assigned to *lim*. The current UAV position s_{cur} is set to *start*. In the case of the A* algorithm, the *start* location will vary by a maximum of half the distance between grid positions, in all 3-dimensions, due to the shifting introduced by the A* ripple reduction algorithm. Furthermore, the path possibility flag ($flag_{path}$), signals whether a path could be created in future iterates (=1) or not as either a path has been created or no possibility of a path exist (=0). Furthermore, the total path computational time *time* is set to 0s.

The real–time algorithm tries to find a path in case the distance between the current UAV position and the final goal point (d_{int_s-to-g}) is larger than the distance between three consequent grid points, the computational time since the start of the path generation process (*time*) has exceeded $t_{path\_gen\_max}$, the allotted time to generate an intermediate path between the current UAV position and the intermediate goal point has been exceeded and a path is possible. In case these conditions are satisfied, the iterate time $time_{iterate}$ is re-set to 0 and re-started.

2. The Move Function

Unless the current UAV position s_{cur} is not the start position *start*, the UAV shall be moved d_{s_step} distance in the direction of the previously constructed path $path_{smooth\_pre}$. The move function defined in Algorithm 2 considers all the situations in defining a feasible future s_{cur}. Besides the constants defined earlier, env_{para} define the environmental parameters namely the size of the environmental space and the size and position of obstacles in the environmental space. In A* as opposed to RRT, the size and position of obstacles may slightly vary due to the discretisation of the environmental space. $path_{smooth}$ denotes all points of the smoothed path generated in the previous iterate.

The move function initiates by checking that the distance to the intermediate goal point from the current UAV position is larger than the distance the UAV is assumed to move between the generation of two paths. In this case, the iterate count i is initialised to 1. d_{total} denotes the distance from s_{cur} to a previous smoothed path point in $path_{smooth\_pre}(i)$ such that, the distance between $path_{smooth\_pre}(i)$ and $path_{smooth\_pre}(i+1)$, $d_{int\_(i)-to-(i+1)}$, is smaller than d_{s_step}. Therefore,

$$d_{s\_step} < d_{total} + d_{int\_(i)-to-(i+1)} \tag{3}$$

For initiation, d_{total} is set to 0 since it is assumed that $d_{s\_step} < d_{int\_(1)-to-(2)}$. $d_{s\_step\_part}$ denotes the distance to be moved by the UAV ($d_{s\_step} - d_{total}$) from $path_{smooth\_pre}(i)$ to define the s_{cur_new}. Figure 2 graphically defines this parameter. As $d_{s\_step\_part}$ must be always smaller than d_{s_step}, the former is assigned to the latter. $flag_{small}$, a Boolean parameter denotes whether the distance between d_{s_step} is smaller than $path_{smooth\_pre}(1)$ and $path_{smooth\_pre}(2)$. In this case $flag_{small}$ is assigned to 0 and vice–versa otherwise. It is first assigned to 0 as the first condition is assumed, otherwise $flag_{small}$ will be toggled in the following While condition. The distance between the first and second smoothed path points ($d_{int\_(1)-to-(2)}$) and the number of points in the previously generated smoothed path (n_{path_points}) are calculated.

In case the distance between s_{cur} and the next smoothed path point is smaller than d_{s_step}, the principle illustrated in Figure 2 is applied. The algorithm traverses from s_{cur} along the smoothed path points one point at a time, each time finding the distance between adjacent points $d_{int\_(i)-to-(i+1)}$. Each time, $d_{s\_step\_part}$ is reduced by $d_{int\_(i-1)-to-(i)}$, defined and added to d_{total} in the previous iterate. Since $d_{s\_step} < d_{int\_(1)-to-(2)}$, $flag_{small} = 1$. The While function continues until when the travelled distance equals or exceeds d_{s_step} or the whole smoothed path was traversed.

In case, $flag_{small} == 1$ then the previous smoothed path point prior which the total distance from s_{cur} was smaller than d_{s_step} (labelled s_{cur_part} in Figure 2) need to be considered as the starting point to move $d_{s\_step\_part}$ to find the s_{cur_new} along the smoothed path. The distance to move d_{mov} is assigned to $d_{s\_step\_part}$ in this case otherwise d_{mov} is assigned to d_{s_step}.

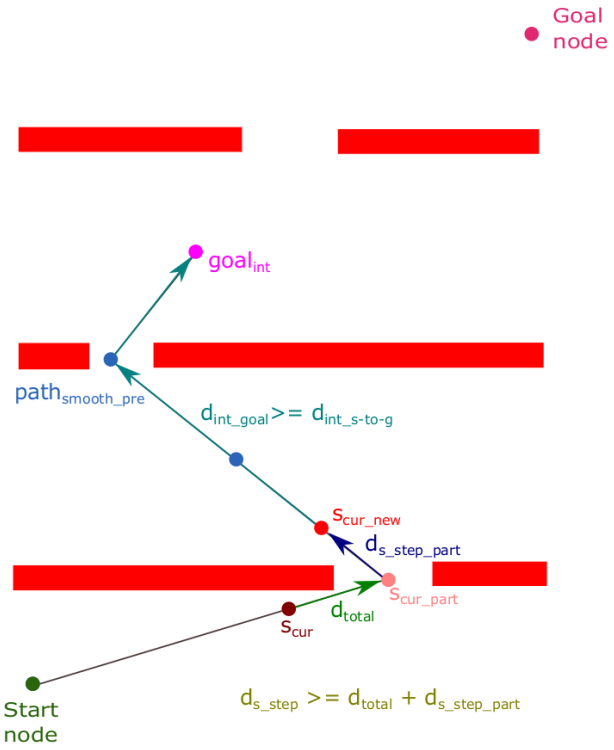

Figure 2. Illustration of the move function

A situation can exist when the s_{cur_new} resides within a distance of $\pm\frac{0.5}{(res-1)}$ from the obstacle planes. This buffer was included and not just checking whether the s_{cur_new} resides exactly on the obstacle plane, due to the environmental discretisation of the A* algorithm. Theoretically, a s_{cur_new} should never reside on an obstacle since a s_{cur_new} is a point on the UAV path. But, if this buffer is not included situations exist in which the UAV would be placed in a position less than half the distance from obstacle planes. This will potentially yield a new starting position on an obstacle plane after the discretisation of the new UAV position and the new intermediate goal points ($goal_{int}$).

In case, the s_{cur_new} is within a distance of $\pm\frac{0.5}{(res-1)}$ from the obstacle planes then a s_{cur_new}, a factor distance ($d_{factor} < 1$) of d_{mov} from the obstacle plane is checked. This process is repeated each time reducing the previous d_{mov} by d_{factor} until an obstacle–free s_{cur_new}, $\pm\frac{0.5}{(res-1)}$ distance from $path_{smooth\_pre}$ is found.

If the UAV is nearer to the intermediate goal point $goal_{int}$ by less than d_{s_step}, either no path has been created since either s_{cur} and/or $goal_{int}$ points are on an obstacle or no path is possible that can connect s_{cur} to $goal_{int}$. In these cases, the s_{cur_new} is assigned to NaN since the algorithm need to stop for the particular test case. In such case the algorithm was not successful in generating a feasible path. In case the last point in the previously generated smoothed path is the goal node, then the s_{cur_new} is assigned to the goal node and then the iterate is stopped in the main algorithm as a path to goal has been found. Otherwise, the s_{cur_new} is assigned to the last point in $path_{smooth\_pre}$. The resultant s_{cur_new} is fed into the main real–time algorithm to generate the next smoothed path $path_{smooth}$. After the s_{cur_new} has been defined the new distance between the latter and the $goal_{int}$ is re-calculated and set to d_{int_s-to-g}.

3. Main Real–time Algorithm

Provided that s_{cur_new} is defined and does not reside on the goal node, the new intermediate goal point $goal_{int}$ is defined as the final goal point ($goal$) if d_{int_s-to-g} is nearer by less than d_{int_goal} to the final goal point otherwise a new intermediate goal point d_{int_goal} distance from the current UAV position s_{cur} is defined in the direction of the goal. In case, that this new intermediate goal point resides on an obstacle, d_{int_goal} is reduced by a factor d_{factor}, similar to the s_{cur_new} calculation. This process is repeated each time reducing the previous distance ($d_{int\_goal} \times d_{factor}$) by d_{factor} until a new obstacle–free intermediate goal

point is found until the resultant $d_{s-to-int\_goal} > d_{s\_step}$. If the latter condition is not considered then a new $goal_{int}$ can reside nearer to s_{cur} by less than 1 step (d_{s_step}) possibly in the same position as the s_{cur_new}. This will effectively imply that the UAV will move by less than 1 step. In such cases no feasible path can be constructed.

The intermediate goal point can be selected in any direction from the current UAV position only if it is assumed that the sensory system has a 360^o field–of–view. With this approach, the UAV will waste time and resources in exploring areas in the vicinity of the start location, limiting progress towards the goal and increasing the risk of collision or attack from obstacles and enemy, respectively. Therefore, new goal points shall be selected in the direction of the goal point unless the new $goal_{int}$ resides on an obstacle.

Once the current UAV position and a valid intermediate goal point are defined, the A* or RRT algorithm are applied to generate a feasible path between the current UAV position and the intermediate goal point. If a path is not generated, then $flag_{path} = 0$ to terminate the While loop of line 04. The $time_{iterate}$ and $time$ are noted irrespective of whether a path has been created or not for analysis purposes. Then the previous position of the UAV s_{cur} is set to the new generated position (s_{cur_new}).

Finally, if the While loop at line 04 is terminated with the UAV reaching the goal position, then the algorithm was successful otherwise the UAV was not able to reach the goal either because a path was not possible or the time allocated was not enough.

Algorithm 1: Real–time Algorithm

01: Define $start$, $goal$, res, $t_{iterate\_max}$, d_{s_step}, $t_{path\_gen\_max}$, d_{int_goal}, d_{factor} and lim.
02: Apply the A* ripple reduction algorithm if A* is considered.[22] Update lim.
03: $s_{cur} = start$; $flag_{path} = 1$; $time = 0$
04: **While** $d_{int\_s-to-g} > \frac{2}{(res-1)}$ **And** $time < t_{path\_gen\_max}$ **And** $time_{iterate} < t_{iterate\_max}$ **And** $flag_{path} == 1$ **Then**
05: Re–set and start $time_{iterate}$
06: **If** $s_{cur} != start$ **Then**
07: $s_{cur\_new} = move(env_{para}, d_{s\_step}, path_{smooth\_pre}, goal, d_{int\_s-to-g})$.
08: **If** $s_{cur\_new} == NaN$ **Then** $flag_{path} = 0$ **End**
09: **End**
10: **While** $(flag_{path} == 1)$ **OR** $s_{cur} == goal$
11: Update d_{int_s-to-g}
12: **If** $d_{int\_s-to-g} < d_{int\_goal}$ **Then** $goal_{int} = goal$
13: **Else** $goal_{int}$ is d_{int_goal} from s_{cur_new} in the direction of $goal$.
14: $iterate = 1$; $d_{s-to-int\_goal} = goal_{int}$
15: **While** $goal_{int}$ is on obstacle **And** $d_{s-to-int\_goal} > d_{s\_step}$
16: $goal_{int}$ is $d_{int\_goal} \times {d_{factor}}^{iterate}$ from s_{cur_new} in the direction of $goal$.
17: $iterate = iterate + 1$; Re–calculate $d_{s-to-int\_goal}$
18: **End**
19: Apply the A* or RRT algorithms to define $path_{smooth}$ using as input arguments the obstacle scenario, res, s_{cur_new}, $goal_{int}$, $path_{smooth\_pre}$ and lim (refer[21])
20: **If** $path_{smooth} == NULL$ **Then** $flag_{path} = 0$ **End**
21: Stop $time_{iterate}$; $time = time + time_{iterate}$; $s_{cur} = s_{cur\_new}$.
22: **End**
23: **End**
24: **If** $d_{int\_s-to-g} < \frac{2}{(res-1)}$ **AND** $flag_{path} == 1$ **Then UAV reached** $goal$.
25: **Else** $goal$ **could not be reached End**
26: **End**

Algorithm 2: Move Function

$s_{cur} = move(env_{para}, d_{s\_step}, path_{smooth\_pre}, goal, d_{int\_s-to-g})$

01: **If** $d_{int\_s-to-g} > d_{s\_step}$
02: $i = 1; d_{total} = 0; d_{s\_step\_part} = d_{s\_step}; flag_{small} = 0$
03: Find $d_{int\_(i)-to-(i+1)}$ and n_{path_points}
04: **While** $(d_{total} < d_{s\_step})$ **AND** $(i < n_{path\_points})$
05: **If** $i > 1$ **Then** find $d_{int\_(i)-to-(i+1)}$ **End**
06: $d_{s\_step\_part} = d_{s\_step} - d_{total}$
07: $d_{total} = d_{int\_(i)-to-(i+1)} + d_{total}; i++; flag_{small} = 1$
08: **End**
09: **If** $flag_{small} == 1$ **Then** $i--$; $d_{mov} = d_{s\_step\_part}$ **Else** $d_{mov} = d_{s\_step}$ **End**
10: Define s_{cur_new}, d_{mov} from $path_{smooth\_pre}(i)$
11: **While** s_{cur_new} is on obstacle (check env_{para})
12: $d_{mov} = d_{mov} \times d_{factor}$
13: Define s_{cur_new}, d_{mov} from $path_{smooth\_pre}(i)$
14: **End**
15: **Else If** $path_{smooth}$ is empty **Then** $s_{cur} = NaN$
16: **Else If** $path_{smooth\_pre}(n_{path\_points}) == goal$ **Then** $s_{cur} = goal$
17: **Else** $s_{cur} = path_{smooth\_pre}(n_{path\_points})$ **End**
18: **End**
19: **End**

C. Conclusion

The previous subsection presented a platform for the real–time implementation of the A* and RRT algorithms applicable to a 3D environment. This generic real–time algorithm was designed to assess the applicability of both path planning algorithms with respect to time and sensory constraints. The UAV is required to be furnished with obstacle free directions whilst it is moving in a previously unknown environment with the aim of reaching a goal in the minimum time without being in a situation of waiting for new directions to follow as the path planning algorithm may not have already provided such information. The success of either or both algorithms will depend upon the considered environment and most importantly the UAV and its onboard systems. In the next section the constants defined above will be correlated with the parameters of real UAVs utilised for indoor applications. Moreover, the experimental scenarios considered will be defined.

V. Parameter Definition and Experimental Scenarios

A. Real–time algorithm parameter assignment

The following table (Table 1) defines the assigned values (Maximum, Minimum and Constant value) for each parameter considered in the real–time path planning algorithm explained in Section IV. Some of the parameters are inter–related and are defined based from UAV path planning literature. As in our previous work[21,22] the resolution (for A*) (step size (for RRT) reciprocal of resolution) was set at 11 to 29 increased in steps of 2.

Literature suggest that a 4s to 5s look–ahead is required for relatively low speed UAVs <50km/hr[56] and more than 20s look-ahead for high speed UAVs >500km/hr.[28] Sensor ranges are a function of the UAV speeds. In low speed application the range will be in the region of 10m[56,57] although large obstacle such as bushes and poles can be identified within a 20m to 25m range.[56] Oppositely, in high speed application such range is extended to a few kilo–meters.[28] The sensor frequency range is defined by researchers in UAV obstacle avoidance algorithms between 10Hz–25Hz.[6,15,43,56] The associated environmental refresh rate is

much smaller than the path computational time of a few seconds and therefore it can be assumed that the environment is refreshed between one step iterate and the next.

The environmental space in low speed applications is a cube in the range of 250–350m[8,57] increasing to 10–25km for high speed applications.[25,28] Obstacle sizes are intuitively defined based on the speed and environmental space. In fact obstacles are defined by Yu *et. al.*[58] in a range of $16m \times 10 \times 100m$ for medium–low speeds of 40km/hr in a $700m \times 700m$ square. Similarly, Call[15] defined a cube of $60m \times 60m \times 60m$ for a speed of 58km/hr in a $500m \times 500m \times 500m$ cube environment. Oppositely, the obstacles in high speed environment was set a few kilometre in all axis.[28]

In our analysis of UAV path planning in indoor applications we will assume that the UAV will travel between 5km/h and 50km/h if this variable is under analysis otherwise 15km/hr if fixed. Based on the above literature, for low to medium speeds in a $500m \times 500m \times 500m$ environmental space, d_{s_step}, the distance moved by the UAV in one iterate, was arbitrary set to the distance between three consequent graph positions translating into a range of 35.71m to 300m for resolutions 29 to 11 respectively. The distance considered shall always be greater than the distance between two graph points especially in A*, since after the discretisation of the environment or when the new prospective UAV position resides on an obstacle and the distance reduction factor is applied, the UAV future position may remain as current and the UAV would be blocked in the same position yielding no path. If d_{s_step} was not under consideration the distance between three consecutive graph positions was considered.

d_{int_goal} translates to approximately 54m to 150m look-ahead distance if a resolution varying from 29 to 11 is considered and a look–ahead circle (as illustrated in Figure 1) of 3 times the distance moved by the UAV in $t_{iterate\_max}$ (d_{s_step}). Theoretically, this parameter must be greater or equal to $d_{s\_step\_min}$ but some buffer must be considered as otherwise the UAV will be placed in a point at the edge of the unknown. Ideally $d_{int\_goal} > 50m$ to allow for this buffer. If this parameter is not under analysis it will be set to 100m irrespective of the resolution considered.

The maximum time to generate a path segment ($t_{iterate\_max}$) is dependent upon the time for the UAV to move d_{s_step}. This time will vary between 4.29s (highest UAV speed, with lowest d_{s_step}) and 36s (lowest UAV speed, with highest d_{s_step}). If this parameter is not under analysis it is set to $\frac{d_{s\_step}}{v_{UAV}}$. The maximum time to generate the whole path ($t_{path\_gen\_max}$) was set in a range of 45s to 360s based upon factor of 10 $\times$ factor of $t_{iterate\_max}$. The distance factor reduction applied in cases where the new intermediate goal point reside on an obstacle was set in a range of 0.5 to 0.95 and 0.8 if this parameter was fixed.

B. Experimental Scenarios

The experimental space is defined as a generic cube of 1x1x1 with centre (0,0,0) and limits [-0.5 $\rightarrow$ 0.5, -0.5 $\rightarrow$ 0.5, -0.5 $\rightarrow$ 0.5] as in our previous work.[21,22] The cube limits can vary by up to $\pm\frac{0.5}{(res-1)}$ in case of the application of the A* ripple reduction algorithm. The generic environmental space is normalised to arbitrary units so that the test scenarios can be exported to any environmental space. These test scenarios were originally developed by Clifton *et. al.*[59] and made available online in.[60] For all tests the start and goal nodes are defined at [0,-0.5,0] and [0,0.5,0], respectively. In case the A* ripple reduction algorithm is applied, the associated shifting is added to the start and goal points. Three different obstacle scenarios, illustrated in Figure 3 are considered. Scenario 1 consists of two Y-Z obstacle planes with 0.2x0.2 square opening later referred to as windows. Scenario 2 consists of three Y-Z with 0.2x0.2 windows and two X-Y planes without windows while Scenario 3 is similar to Scenario 2 but with five Y-Z planes instead of three.

VI. Results

A. Introduction

The real–time path planning algorithm defined in Section IV was implemented using both the A* and RRT algorithms in the experimental scenarios defined in Section V.B with the parameters defined in Section V.A. Each section will analyse each parameter separately keeping all other parameters constant so that its effect on path planning performance in terms of length and time will be assessed independently. All tests are performed using an Intel Xeon ES–1650, 3.2GHz.

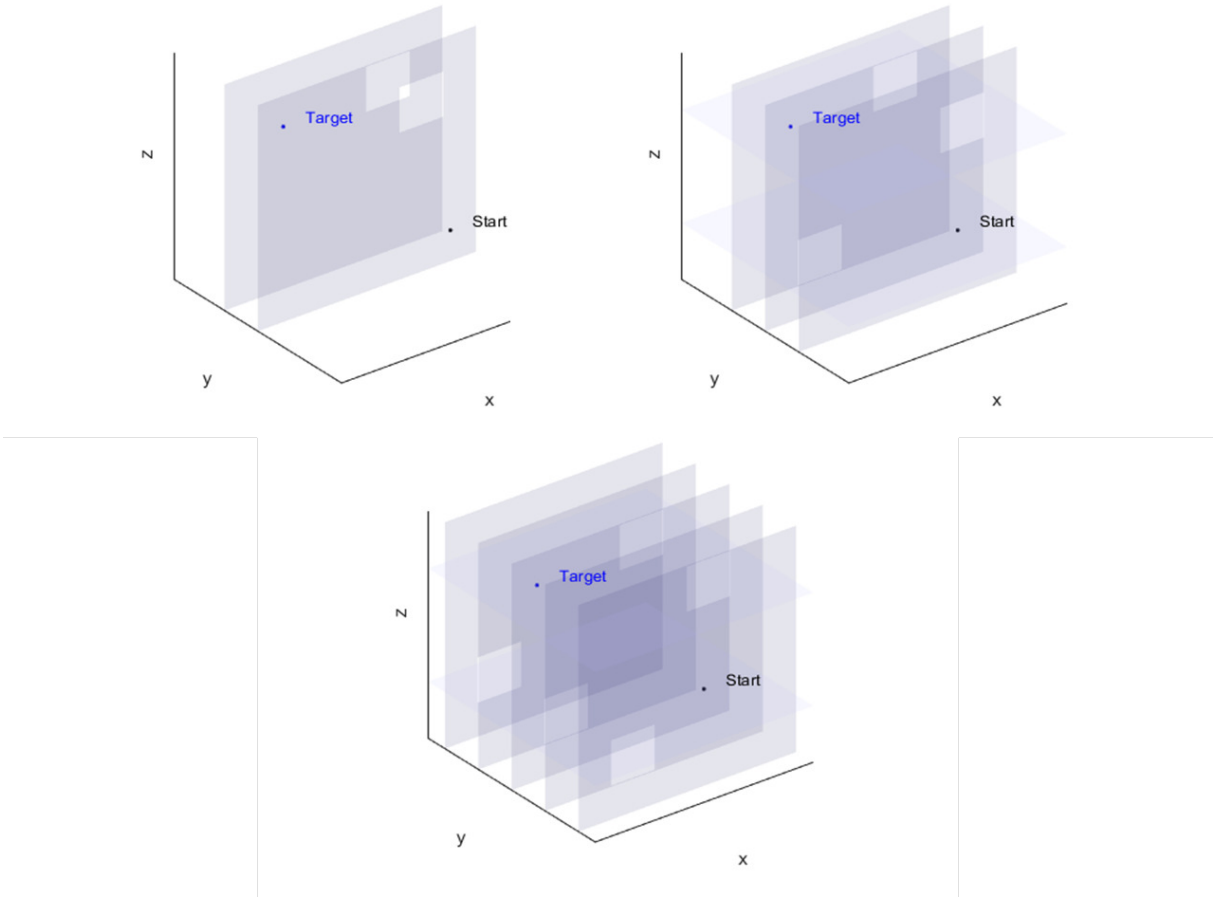

Figure 3. Obstacle scenarios: (a) First Scenario (b) Second Scenario and (c) Third Scenario modified from,[21] consisting of obstacle planes in the Y-Z with windows as openings and X-Y planes for scenarios 2 and 3 with no openings.

Table 1. Real–time algorithm parameter definition

Parameter	Minimum	Maximum	Nominal Value	Units
Resolution (res)	11	29	21	[-]
Step size RRT (d_{step_RRT})	$\frac{1}{res_{max}-1} = 0.0357$	$\frac{1}{res_{min}-1} = 0.1$	$\frac{1}{21-1} = 0.05$	[-]
UAV Speed (v_{UAV})	5	50	15	km/hr
Distance to travel per iterate (d_{s_step})	$\frac{2}{res_{max}-1} \times 500 = 35.71$	$\frac{u_{UAV\_max} \times 1000 \times t_{iterate\_max}}{60 \times 60} = 300$	$\frac{2}{res-1} \times 500$	m
Distance between current UAV position and prospective new intermediate goal point (d_{int_goal})	$d_{s\_step\_min} = 35.71$	150	100	m
Maximum time to generate path segment ($t_{iterate\_max}$)	$\frac{d_{s\_step\_min} \times 60 \times 60}{u_{UAV\_max} \times 1000} = 4.29$	$\frac{d_{s\_step\_max} \times 60 \times 60}{u_{UAV\_max} \times 1000} = 36$	$\frac{d_{s\_step} \times 60 \times 60}{v_{UAV} \times 1000}$	s
Maximum time to generate path ($t_{path\_gen\_max}$)	$10 \times t_{iterate\_max}(min) = 45$	$10 \times t_{iterate\_max}(max) = 360$	$10 \times t_{iterate\_max}$	s
Distance reduction factor (d_{factor})	0.5	0.95	0.8	[-]

B. Speed (v_{UAV})

Speed is one of the variable parameters that is considered for analysis. Speed is varied between 5km/hr and 50km/hr as defined in Table 1 in steps of 5km/hr. For each considered speed for both A* and RRT algorithms the test is performed 100 times and the mean with a 95% confidence interval is illustrated in Figure 4. Since both algorithms were not able to generate the path in all considered scenarios either because the time to generate a path between iterates or the total time exceeded the allocated, a bar graph showing the distribution of successful and unsuccessful paths for each scenario considered is illustrated in Figure 4 (c) and (f) for A* and RRT respectively. The unsuccessful runs were not considered in the path length and time vs. speed plots for both path planning algorithms.

The mean path length and standard deviation for A* remains almost constant for all considered speed situations. Similarly for RRT, the mean path length remains constant for all considered speeds although the success rate reduces with increase in speed for all scenarios. The standard deviation remains constant except for the high–end speeds of Scenario 3. This is attributed to the fact that as scenario complexity and speed increase the algorithm is less successful (refer to Figure 4 (f)) and therefore the mean and standard deviation are measured on a smaller sample made up of the best performance runs. In fact in this case, the amount of successful runs decreases reducing the sample size to less than 10 with no successful runs at 45km/hr and only 1 successful run for 40km/hr and 50km/hr.

As in our previous work[21,22] the path length depends mainly on Scenario complexity for both the A* and RRT algorithms as a longer path is required to traverse through obstacle plane windows residing in alternating sides of obstacle planes. Furthermore, the mean path length for A* is smaller than RRT confirming theory that the A* algorithm is more optimal than RRT.[42] The difference in path length between the A* and RRT algorithms increases further as the scenario complexity increases. A* generated paths are more optimal and the path length reduction by the smoothing algorithm is less than that for RRT. In complex scenarios the improvement introduced by the smoothing algorithm is limited since the elimination of an intermediate node is more likely to generate a colliding paths and therefore the oscillatory RRT–generated paths will be reduced by a lesser margin with respect to A* for the same situation.

The time to generate a path for A* is independent of the speed for all scenarios. This result confirms theory since the algorithm will utilise the same amount of time to generate a path as it considers approx-

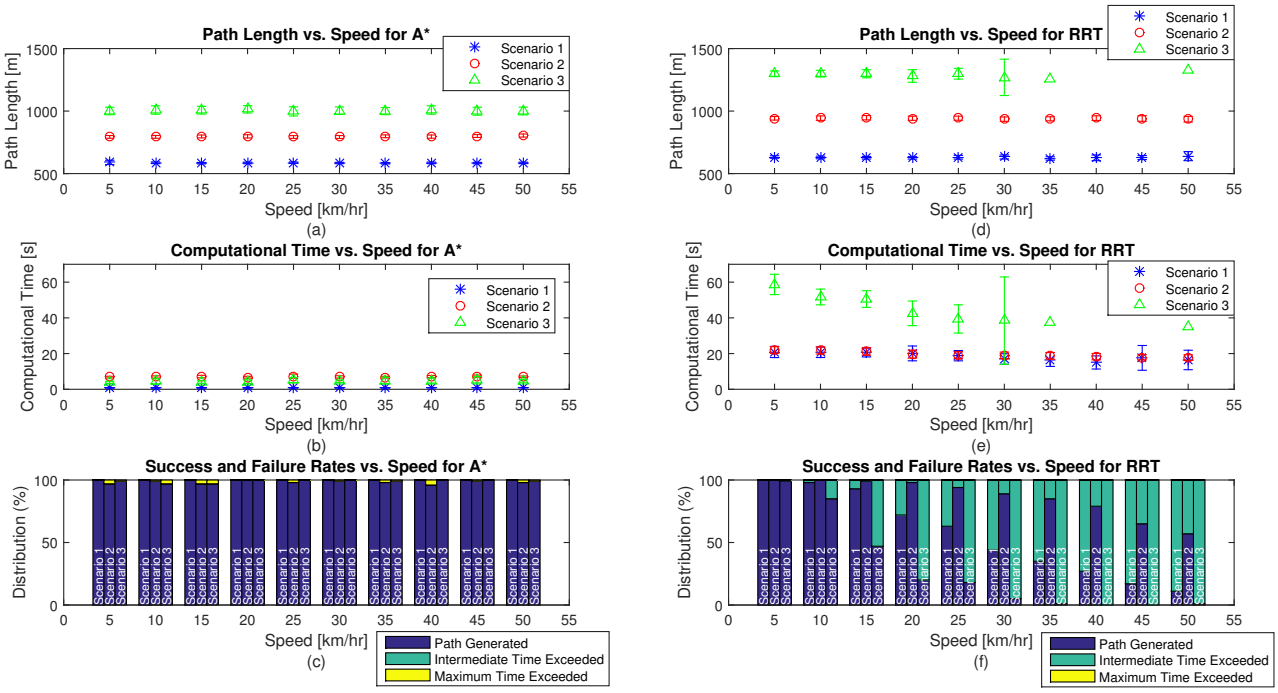

Figure 4. Performance parameters vs. speed: (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, d_{s\_step} = 50m, d_{int\_goal} = 100m, d_{factor} = 0.8$ **and** $t_{iterate\_max}, t_{path\_gen\_max}$ **are a function of the UAV speed).**

imately the same intermediate start and goal nodes irrespective of the time required by the UAV to travel to the next position provided that this time is longer than the time to generate the intermediate path. As complexity increased the mean time increased to approximately 7 times and 4 times for Scenarios 2 and 3, respectively with respect to Scenario 1 while the standard deviation increases proportionally. For the considered scenarios as the complexity increases the time increases as it takes longer to find a non-colliding path and an intermediate start and goal nodes.

For RRT the computational time reduces with increase in UAV speed for all scenarios. This time reduction is attributed to the analysis described earlier that the best performing runs for the same situation were considered as other runs did not generate a path in the allocated time. In fact the drop in computational time is in line with the drop in successful runs (refer to Figure 4 (f)). The drop rate in successful runs for Scenario 2 with increase in speed is lower than for the other scenarios. This behaviour is mirrored in the computational time.

Scenario 2 is the most successful scenario for RRT although it is more complex than Scenario 1. From our previous results,[21,22] the difference in the whole path generation time between Scenario 1 and 2 using the RRT algorithm is close to zero for all considered step sizes. Furthermore, in simple scenarios with oscillatory–generated paths such as in RRT the smoothing iterates required until the stopping condition is reached is larger than that of the same path in an obstacle–rich environment since it is more likely in the latter for a smoothed path segment to collide with an obstacle. The increase in computational time due to smoothing increases the time to generate an intermediate path, exceeding the maximum intermediate time allocation. Since for Scenario 2, the possibility of smoothing is lower than Scenario 1 while the difference in time to generate a path between intermediate nodes is similar in both cases then this explains the result that Scenario 2 is more successful than Scenario 1. Although more iterates are required for a longer Scenario 2 path than Scenario 1, the bottleneck in RRT is the maximum intermediate time not the total time. For Scenario 3 the situation is different. From our previous work,[21,22] a longer computational time (more than 5 times) was required to generate a complete path due to the condensed obstacle space and very limited path options. This had a major effect at high speeds as a longer path must be generated in the same time with increase in speed.

The computational time for A* when compared to RRT is shorter by multiple times for all scenarios. From our previous results it can be concluded for the considered resolution/step size that the computational time for Scenarios 1 and 2 is similar increasing by a 3 times factor in Scenario 3 for RRT with respect to A*.[21,22] During path construction the A* considers points between the current UAV position and the intermediate goal point. On the other hand, the RRT algorithm can produce seeds anywhere in the available space although the tree branch length is limited by d_{step_RRT}. This implies that the considered area in A* is much less than RRT. Therefore multiple times more time is required for RRT with respect to the A*. In fact the difference in computational time for Scenarios 1 and 2 increases by a factor of about 5 times for RRT with respect to A* as the whole environmental space is 5 times the look-ahead distance. Similarly, for Scenario 3 the difference is even larger since this 5 times factor is multiplied by the 3 times factor described earlier. Furthermore, the RRT algorithm is less optimal than the A* and therefore more smoothing iterates are required until less than 1% reduction results over the past 20 smoothing iterates.

In all considered situations the A* algorithm was able to generate a path in the allocated maximum intermediate and total time in a range between 96% to 100% with an average of 99%. The maximum time to generate the path was exceeded was the stopping condition triggered in the unsuccessful cases. Not the same can be deduced for RRT. As the considered speed increased the success rate dropped. For Scenario 2 the success rate dropped to a minimum of 57% at 50km/hr although the algorithm was able to generate a path in at least 94% of the cases from low speeds up to 30km/hr. The stopping condition triggered for all unsuccessful situations was always that the maximum allocated intermediate time was exceeded. This is mainly attributed to the lower optimality of the RRT algorithm with respect to the A* algorithm and the fact described earlier that the RRT algorithm considers all environmental space when generating an intermediate path. For Scenario 1 the success rate drop started even at low speeds of 10km/hr increasing with increase in speed up to 11% at the highest considered speed. A sharper drop was experienced in Scenario 3 where a 47% success rate was noted at a speed of 15km/hr dropping further to 5% at 30km/hr reaching 0 successful run at 45km/hr. This result show that A* can be applied for all speeds considered at a resolution of 21 and maximum and intermediate time allocation while RRT can only be successfully applied for low speeds not exceeding 10km/hr for Scenarios 1 and 2 and less than 5km/hr for Scenario 3 for a step size of 25m for the same time constraints as A* unless the intermediate step size is increased. This situation does not

eliminate the RRT algorithm from consideration as the amount of smoothing can be reduced reducing the intermediate time to generate a path ultimately increasing the success rate.

C. Distance to travel per iterate (d_{s_step})

The distance to travel per iterate is a function of the UAV speed as the distance the UAV travels in a pre–defined time dictates the maximum allowable time to generate a path. This parameter was otherwise set to double the distance between grid positions (A*) or double the step size (RRT). To assess the effect of this parameter on performance, it was varied between this minimum value and the look ahead distance (d_{int_goal}). Therefore, the distance to travel per iterate was varied from 50m to 100m in steps of 5m. The longer the distance to travel per iterate the more intermediate time is allocated for the path planning and smoothing algorithms. As for speed, for each considered distance for both A* and RRT algorithms the test is performed 100 times and the mean with a 95% confidence interval is illustrated in Figure 5. A similar bar graph representing the successful and unsuccessful runs is also illustrated in 5 (c) and (f) for A* and RRT, respectively.

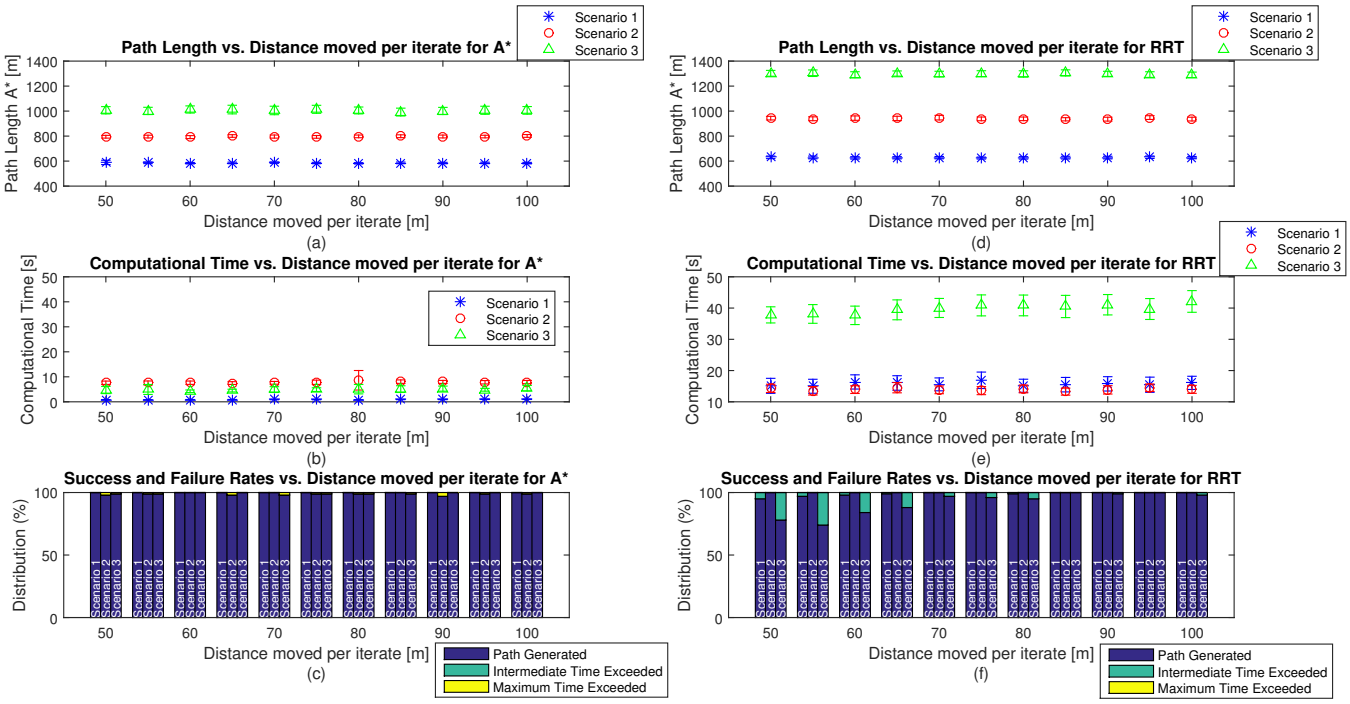

Figure 5. Performance parameters vs. Distance to travel per iterate (d_{s_step}): (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, v_{UAV} = 15km/hr, d_{int\_goal} = 100m, d_{factor} = 0.8$ and $t_{iterate\_max}, t_{path\_gen\_max}$ are a function of the distance to travel per iterate).

As for the speed analysis, the mean and standard deviation in path length for both the A* and RRT algorithms is independent of distance to travel per iterate (d_{s_step}) but depends primarily on scenario complexity. The range of d_{s_step} considered is equivalent to the distance of three consecutive grid positions or $2 \times d_{step\_RRT}$. This relatively small variation and the relatively small windows through which the planner must pass shall theoretically have minimal effect on the path length especially in obstacle–rich environment, as confirmed by the results of Figure 5 (a) and (d). Furthermore, as A* is more optimal, the mean path length for A* is shorter than RRT with the difference increasing with scenario complexity as deduced in the previous sub–section.

For both algorithms the computational time is independent of d_{s_step} although as d_{s_step} approach its minimum value (double the distance between grid points for A* and equivalent for RRT), the computational time decreases. This minor improvement results as the path planning task will be divided into a larger

simpler sub–tasks. This result is more evident for Scenario 3 of the RRT algorithm since the success rate at low d_{s_step} drops and therefore, as described earlier, the best performance results are considered in the mean and standard deviation calculations. As deduced from literature and in line with the conclusions drawn in the speed analysis the RRT algorithm required multiple times (4x to 10x) more time to compute a path with respect to A* algorithm for the same conditions.

The A* algorithm was able to generate a path in all situations for Scenario 1 and in at least 97% and 98% for Scenarios 2 and 3, respectively. The maximum computational time is the stopping condition for unsuccessful runs. Not the same can be concluded for RRT as the success rate dropped especially for Scenario 3 due to insufficient intermediate time as d_{s_step} approached its minimum value. A minor drop results for Scenario 1 while the RRT algorithm was 100% successful for Scenario 2. The result shows that the RRT is more computationally intensive since as d_{s_step} approach its minimum value and hence the allowable computational time is lowest (path planning algorithm needs to generate a path in the time required to traverse double the step size d_{step_RRT}), the success rate drops.

The bottleneck in the considered path planners is the computational time, especially for the RRT algorithm. The analysis carried out in this sub-section shows that the distance traversed by the UAV in each iterate will not effect the validity and optimality of the generated path and is independent of the computational time required to generate the path. From a practical point of view, reducing the dependency of the UAV from the planner by increasing d_{s_step}, will allow more time for the planner to generate a path improving the success rate without deteriorating the path length. Therefore the longer d_{s_step} the better. But the latter parameter is limited by the look-ahead distance which is governed by the sensory systems and provided that all obstacles residing between the current and future UAV positions do not cross the path line in the time the UAV is traversing.

D. Distance between current UAV position and prospective intermediate goal point (d_{int_goal})

The look-ahead distance defined as the distance between the current UAV position and a prospective intermediate goal point is another parameter that effects the performance of the path planning algorithms. This distance is mainly dependent upon the precision and accuracy of sensory systems available on-board a UAV and thus can vary between different UAV models. This parameter was varied between the distance travelled by the UAV per iterate to 150m which for the considered resolution/step size is 3 times the distance moved by the UAV in every step. In the former case the planner will define a path that will be totally traversed by the UAV in the next iterate while in the latter only a third of the path will be traversed. In practice the minimum value considered can result in situations where the intermediate goal node will reside exactly in the vicinity of an obstacle resulting in a collision into a no solution in the next iterate as no look–ahead will be considered.

Therefore, the look–ahead distance (d_{int_goal}) was varied from 50m to 150m in steps of 10m. As for the other two parameters, for each considered distance for both A* and RRT algorithms the test is performed 100 times and the mean with a 95% confidence interval is illustrated in Figure 6. A similar bar graph as in the previous results representing the successful and unsuccessful runs is also illustrated in Figure 6 (c) and (f) for the A* and RRT algorithms, respectively.

The results in Figure 6 show that the path length for A* is mainly dependent upon the scenario difficulty. Another interesting point is that as the look-ahead distance increases the path length reduces for simple scenarios and slightly increases for complex scenarios. For Scenario 1, an 8% difference between the lowest and highest look–ahead distance with respect to the mean results. This difference is attributed to the fact that the longer the look–ahead distance the lower the variation from the shortest line connecting the start and goal points when considering also that in this scenario only an upper turn is required to pass through plane windows. On the other hand, for Scenario 2 and 3, the lower path length results at the lowest look–ahead distance and as the look-ahead distance increases to $3 \times d_{s\_step}$. For obstacle-rich scenarios like Scenarios 2 and 3, a maxima results at $1.5 \times d_{s\_step}$. This results since in complex scenarios the intermediate goal point positions are very limited especially in Scenario 3 and such a look–ahead distance can offset the intermediate goal point which in the next iterate can change drastically. If this look–ahead distance is reduced to d_{s_step}, then the next UAV position will be the current intermediate goal point leading to very low overshoot in complex scenarios. For Scenario 3 this drop in path length is more evident than that of Scenario 2 due to the drop in success rate for Scenario 3 at low look–ahead distances. For Scenario 2, a drop in success rate is exhibited at the path length maxima, which otherwise would lead a close to null maxima just as for RRT. Furthermore, as the look–ahead increases from $1.5\times \rightarrow 3 \times d_{s\_step}$, the overshoot is limited as for the lower

consider look–ahead distance case due to the reason described for Scenario 1. The reduction margin is lower than that of Scenario 1 due to the limited free space.

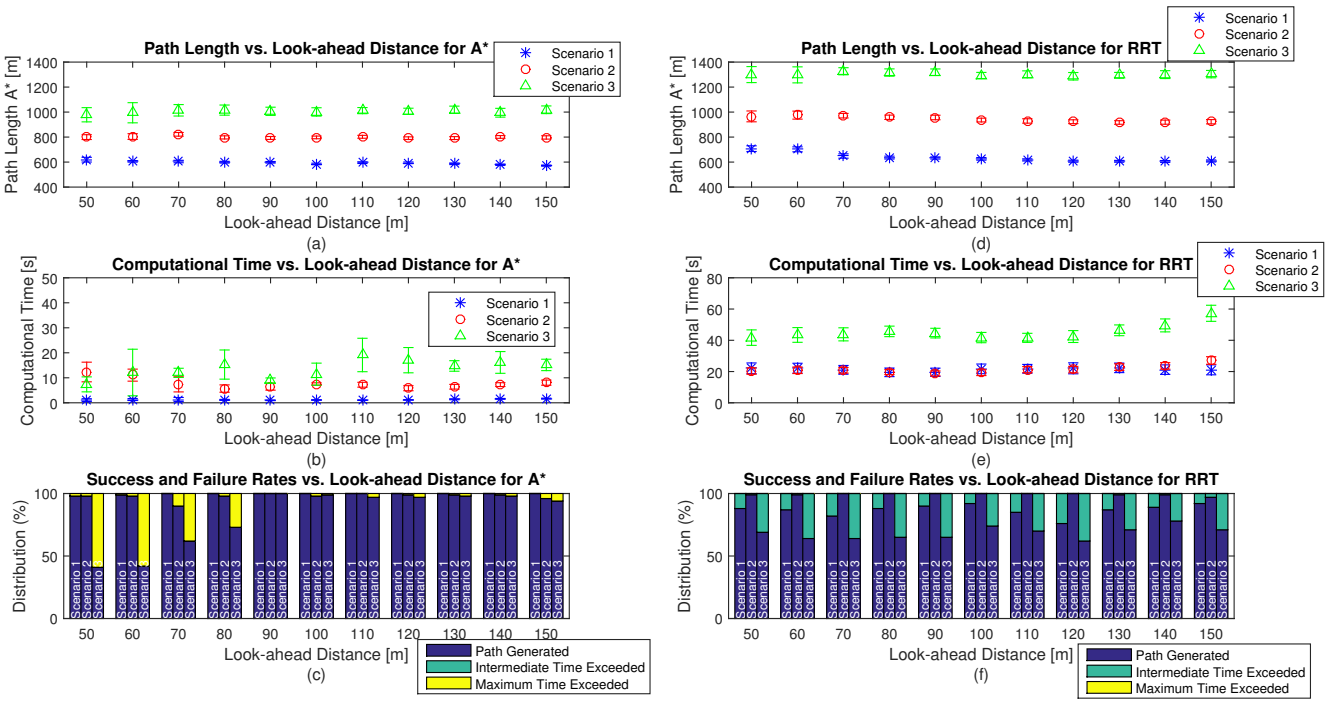

Figure 6. Performance parameters vs. Look–ahead distance (d_{int_goal}): (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, v_{UAV} = 15km/hr, d_{s\_step} = 50m, t_{iterate\_max} = 12s, t_{path\_gen\_max} = 120s$ **and** $d_{factor} = 0.8$**).**

As deduced in the above analysis A* produced shorter paths with respect to RRT for all scenarios considered and the difference in length increases with an increase in scenario complexity. As for A* in Scenario 1, the path length decreases by 16% with respect to the mean with increase in look–ahead distance. This result to double the drop for the same scenario in A*. In RRT intermediate goal points can reside on any obstacle free–point on the connecting line from the current UAV position to the final goal point. This advantage over the A* which is limited by grid positions, is more advantageous as the look–ahead distance increases as the intermediate goal point will reside nearer to the goal central horizontal area at the end of which is the final goal point, limiting overshoot especially for simple scenarios. For Scenario 2, just as Scenario 1 a lower drop of 6% was recorded due to a more obstacle–rich environment with decreasing path length with increase in look-ahead distance with the same maxima occurring at $1.5 \times d_{s\_step}$ for the same reasons described for A*. Similarly for Scenario 3, the same maxima is exhibited but the drop at low look–ahead distance is lower with respect to A*, confirming that the drop in success rate for A* in Scenario 3 is responsible for a reduction factor in path length at low look–ahead distances.

The computational time for both A* and RRT algorithms mainly depends upon the scenario complexity although for RRT Scenarios 1 and 2 yield almost the same result as more smoothing iterates are required in Scenario 1 with respect to Scenario 2, as the latter limits smoothing due to the obstacle–rich environment. The computational time difference between A* and RRT is the same as described earlier with A* superseding RRT in all scenarios. The variation in computational time with respect to look–ahead distance can be approximated by a parabola for all scenarios in both path planning algorithms. For A* a local minima in computational time is exhibited between 80m and 90m for all scenarios, whilst for RRT the lowest computational time is exhibited at 90m, 90m and 110m for Scenarios 1 to 3, respectively.

The success rate for A* is 100% for 90m in all scenarios. For Scenario 1, the success rate remains 100% for look–ahead distance larger than 70m dropping by 2% for the lowest considered resolution. For Scenario 2, a 10% unsuccessful rate results at a look–ahead distance of 70m with all other situations exhibiting an

unsuccessful rate of 5% or less. This occurs at the maxima of path length for this particular scenario. For Scenario 3, an unsuccessful rate of less than 8% is exhibited is for look–ahead distances greater than 80m. For lower look–ahead distances the success rate drops to 40%. The maximum time was exceeded in all unsuccessful runs. For RRT in the first Scenario the success rate varies between 74% at 100m (d_{int_goal}) to 92% (d_{int_goal}). For Scenario 2, the success rate varies from 97% to 100%. For Scenario 3, the success rate varies from 78% at 150m (d_{int_goal}) to 61% at 120m (d_{int_goal}). The maximum allowable intermediate time was exceeded in all unsuccessful runs. In conclusion, between 90m and 100m both the A* and RRT algorithms exhibited the best results.

From the path length, computational time and success rate results it can be concluded that the optimal look-ahead distance for both algorithms for the considered scenarios, resolution/step size and speed shall be $1.8\times \rightarrow 2\times$ the distance moved per iterate. In other words, the planner shall ideally define an obstacle–free position two steps from the current position with knowledge of the environment within this distance.

E. Maximum Intermediate time ($t_{iterate\_max}$)

The maximum intermediate time is the maximum time allocated for a path to be generated from the current UAV position to an intermediate goal point. As defined in Table 1, it is mainly dependent upon speed, the distance moved between iterates d_{s_step} and the computational power onboard the UAV. From the results in Sections B to D, it can be concluded that this parameter is the bottleneck, especially for RRT in complex scenarios. Therefore, to further assess the effect of this parameter in performance it was varied between 3m and 12m in steps of 1m. These values were defined in line with the minimum and maximum values defined in Table 1 assuming a constant speed of 15km/hr and the distance moved per iterate of double the distance between grid position or step size for A* and RRT, respectively, that is 12m. Figure 7 illustrate the mean test results for 100 runs with a 95% confidence interval. A similarly framework as in the previous subsections was utilised to represent path length, computational time and success rate for both the A* and RRT algorithms.

Results of Figure 7 (a) and (d) confirm the results described earlier that the path length is mainly dependent upon the scenario complexity and that the A* algorithm outperformed the RRT algorithm with the difference increasing with scenario complexity. Similarly for Figure 7 (b) and (e), the same conclusions as above can be drawn, with the A* outperforming the RRT in all scenarios considered with the difference increasing for Scenario 3.

The main scope of this analysis is to define the lowest intermediate time that shall ensure that the UAV has time to generate an intermediate path in real time for all considered scenarios. The maximum intermediate time was defined based on real UAV parameters for a nominal speed and using an off–the–shelf processor. This parameter is directly proportional to the distance moved per iterate and inversely proportional to the UAV speed. Therefore when increasing the maximum allowable intermediate time either the UAV speed is decreased or the distance moved per iterate is decreased or both. The effect of each of these parameters on path planning performance was already described in their respective sub–sections. Usually the UAV speed is defined by application and therefore its range is restricted. The lower limit of the distance moved per iterate is limited for the resolution/step size and if resolution increases so that the former parameter decreases, the computational time will increase due to larger resolution for A*. The limitation for RRT, a sampling–based algorithm is not that direct although the shorter the step size the more computational demand is required to generate a path for the same distance.

For the A*, for the considered resolution/step size, speed, distance travelled per iterate and look–ahead distance, the planner was never limited by the allocated intermediate time. The low unsuccessful runs ($\leq 3\%$) results since the maximum time to generate the path was exceeded and not the intermediate time was not enough. Therefore, the defined maximum intermediate time limit can be further reduced from 12s for all scenarios possibly allowing the UAV to increase its speed and/or increase in resolution.

Not the same can be concluded for RRT that was able to generate the path in less than 35% for the maximum considered intermediate time allocation in Scenario 3. For Scenario 1, at least 11s are required to achieve a success rate greater than 90% while for Scenario 2, at least 4s are required to achieve a 90% success rate. For the considered speed of 15km/hr and $d_{s\_step} = 50m$ the maximum intermediate computational time is 12s. This shows that for Scenario 3, the intermediate time must be increased by multiple factors, to achieve a success rate of 100%. For Scenario 1, this limit must be increased to a higher value by either decreasing the speed of decreasing the RRT step size with the latter leading to counter effects of increasing computational time. For Scenario 2, the limit under analysis can remain the same for the considered speed,

distance moved per iterate and RRT step size.

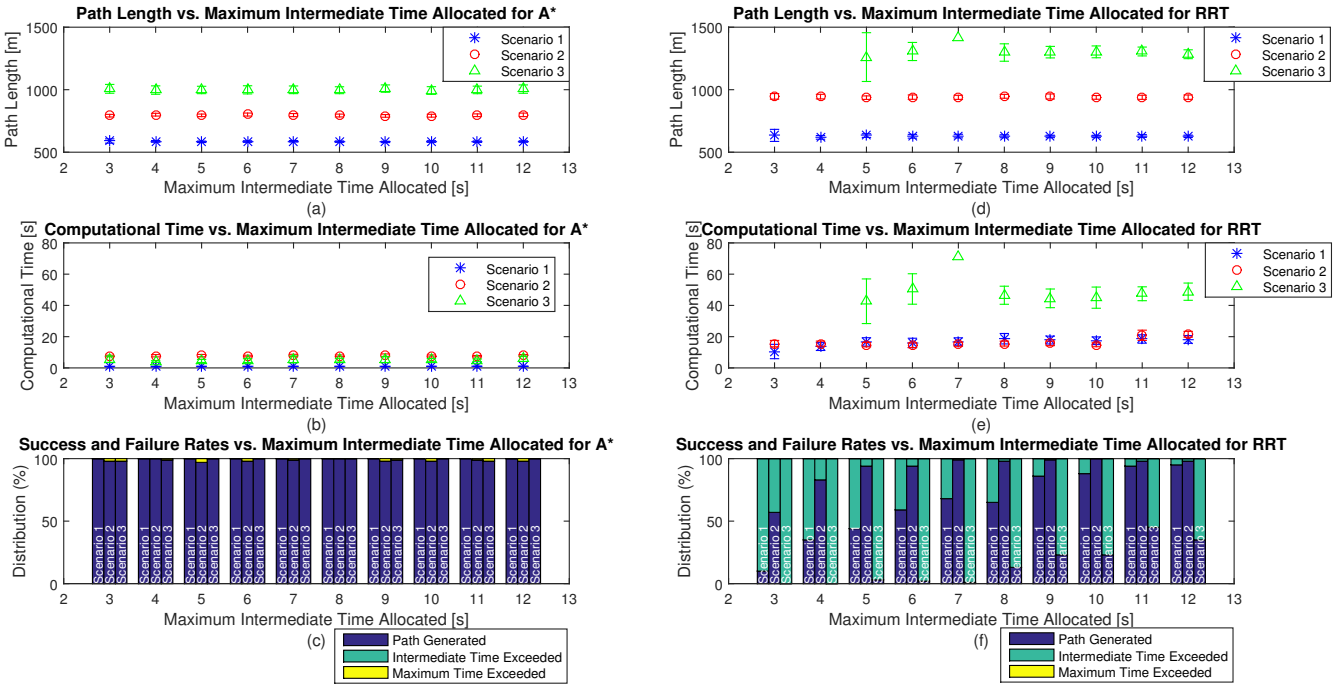

Figure 7. Performance parameters vs. Maximum Intermediate Time Allocated ($t_{iterate\_max}$): (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, v_{UAV} = 15km/hr, d_{s\_step} = 50m, d_{int\_goal} = 100m, d_{factor} = 0.8$ and $t_{path\_gen\_max}$ is a function ($\times 10$) of the maximum intermediate time allocated).

F. Maximum time to generate a path ($t_{path\_gen\_max}$)

This parameter is a function of the length of the path from start to goal. In the previous test analysis, this parameter was arbitrary defined at 10× the maximum intermediate time irrespective of the scenario difficulty. Such value will discriminate against longer paths. In fact, for this considered nominal value for all considered runs, only a small percentage of runs for Scenario 3 were stopped due to this maximum time exceeded limitation. Although this parameter can be defined as a function of the number of intermediate path generations, it could result that the path planning algorithm may waste time venturing around prior arriving to the goal. Moreover, the allocated time to reach a goal may be defined by the application. Therefore to define the best value for each Scenario this parameter was varied between 2× and 20× the maximum intermediate time ($t_{iterate\_max}$) in steps of 2. This implies that the masximum total time to generate a path was varied between 24 and 240 seconds. As in all other situations, Figure 8 illustrate the mean test results in terms of path length, computational time and success rate for 100 runs with a 95% confidence interval for both A* and RRT algorithms.

As concluded in all the previous analysis, results of Figure 8 (a) and (d) confirm that the path length is mainly dependent upon the scenario difficulty with the A* algorithm outperforming the RRT algorithm in all scenarios with the percentage difference in path length increasing with scenario difficulty. Similarly, as can be deduced from Figure 8 (b) and (e) the computational time is lower for A* with respect to RRT, for the respective scenarios, with the major difference of multiple times present for Scenario 3.

As previously, the main scope of this analysis is to define an adequate maximum time for the UAV to reach the goal point in view of a real time implementation. This parameter is dependent on a number of unrelated or inter–related and bounded or unbounded parameters namely scenario complexity, UAV speed, allocated, computational power, allocated intermediate time and the distance moved per iterate, assuming that the sensory system update rate is much higher than the allocated intermediate time. Therefore this

parameter must be chosen in view of these relationships and other external constraints such as a user defined maximum mission length.

For the A* algorithm, Scenario 1 was successful in all considered instances of maximum time allocation. This implies that for simple scenarios the A* algorithm can generate multiple path segment in $1 \times t_{iterate\_max}$, reaching the goal in less than 1 second. For Scenario 2, 5% of runs were unsuccessful since the total time was exceeded prior finding a solution. From Figure 8 (b), the upper bound in computational time never exceeded 8s and therefore it can be concluded that the A* will not be able to reach the goal irrespective of the allocated total time which in the lowest case was $2\times$ the highest computational time recorded. Similarly the same conclusions can be drawn for Scenario 3 although the unsuccessful rate is only 2% maximum. Although these values are low, A* could not guarantee a 100% solution for complex scenarios. This confirms theory that claims that A* does not offer a guarantee of solution.[48]

For the RRT algorithm, the success rate improved as the maximum time to generate the path increases from 24s to 240s. For Scenarios 1, 2 and 3 the success rate improved from 84% to 95%, from 83% to 100% and from 0% to 72%, respectively. These results confirm that the RRT algorithm is computational intensive especially in complex scenarios. Furthermore, if the allocated time is greater or equal 36s, the maximum intermediate time limitation is triggered and not the parameter under review. Besides Scenario 2, although for Scenario 1 and 3 an improvement is present 5% and 28% of the runs, respectively remain unsuccessful and does not decrease further unless the intermediate time is increased further.

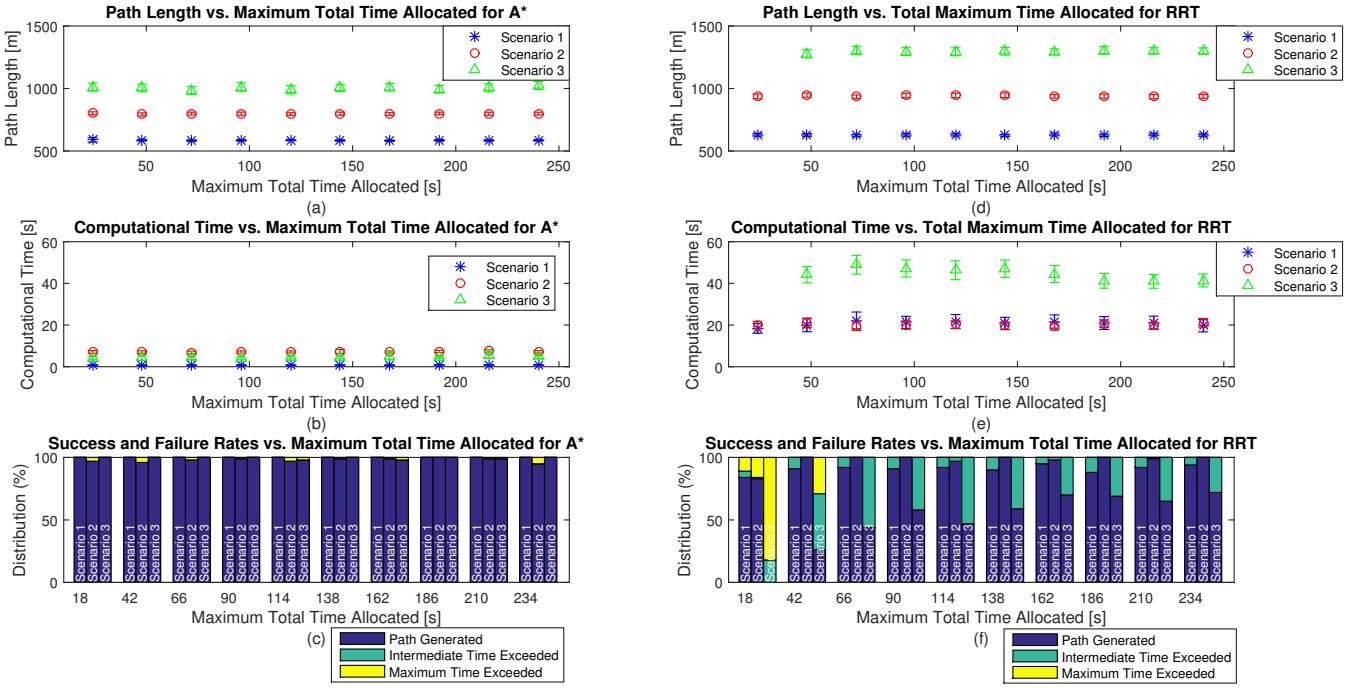

Figure 8. Performance parameters vs. Maximum Total Time Allocated ($t_{path\_gen\_max}$): (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, v_{UAV} = 15km/hr, d_{s\_step} = 50m, d_{int\_goal} = 100m, d_{factor} = 0.8$ and $t_{iterate\_max} = 12s$).

In conclusion although the maximum allowable time to reach the goal was increased, neglecting real–time constraints, a solution cannot be guaranteed for all scenarios for both algorithms, although a minimum of 95% success rate was recorded for A* for all situations considered and the maximum time exceeded condition was never triggered beyond $4 \times t_{path\_gen\_max}$ for all scenarios in RRT. Furthermore it can be concluded that for both A* and RRT at $6\times$ and above, the maximum allowable time to reach the goal has minimal to low effect on performance for the considered parameters listed in the caption of Figure 8 which were defined after respective analysis.

G. Distance reduction factor (d_{factor})

The distance reduction factor is a parameter utilised to re–evaluate the distance from the current UAV position to the next UAV position and from the current UAV position to an intermediate goal position as both may reside on an obstacle. When the next UAV position is defined this position is not on an obstacle as this point is selected on the path from the current UAV position to an intermediate goal point. But when the ripple reduction algorithm is applied, the current UAV position may reside on an obstacle. When the intermediate goal node is defined based on a look–ahead distance from the current UAV position, such point can reside on an obstacle. The excessive reduction of both distances can lead to longer computational time to generate a path as shorter path segments are generated and/or situations in which the UAV and the goal point will remain in the same position. On the other hand, too low reductions will also increase computational time as each reduction needs to be checked each time. Therefore, to best define this value this parameter was varied between 0.5 to 0.95 in steps of 0.05 and applied the tests on the same test platform as in the previous sub–sections. Figure 9 illustrate the mean path length, computational time and success rate test results for both A* and RRT algorithms.

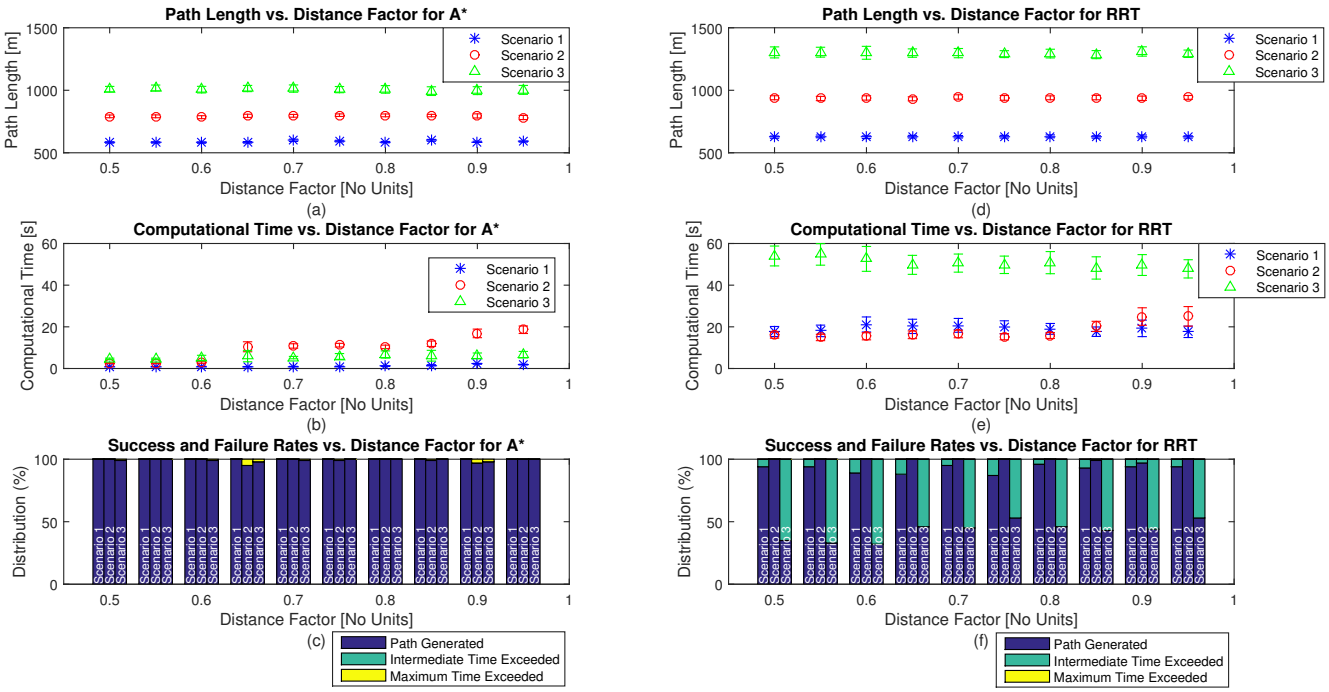

Figure 9. Performance parameters vs. Distance Factor (d_{factor}): (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{step\_RRT} = 50m, v_{UAV} = 15km/hr, d_{s\_step} = 50m, d_{int\_goal} = 100m$, $t_{iterate\_max} = 12s$ **and** $t_{path\_gen\_max} = 120s$**).**

Results in Figure 9 (a) and (d) show that the path length is mainly dependent upon the scenario complexity and independent of the distance factor parameter for both algorithms. This implies that irrespective of the reduction in length of path segments, in the case where the new intermediate goal point or current UAV position reside on an obstacle, the path length shall not increase.

As the distance factor approaches unity the computational time shall increase as the algorithm is required to consider more points in the line connecting the current and future UAV position and/or the current and future intermediate goal points. For all scenarios in A*, an exponential increase in mean computational time with multiple increases in confidence interval is exhibited at different values of distance factor, namely 0.85, 0.65 and 0.6 for Scenarios 1 to 3, respectively, with Scenario 2 exhibiting the largest ($> \times 3$) increase. This results impose the need to best define this parameter based on the different parameters considered and the scenario in which the algorithm is required to operate. Furthermore, Figure 9 (c) shows that at certain instances of distance factor namely, 0.55, 0.8 and 0.95 the A* algorithm is 100% successful. Therefore, this

show that A* can offer a high success rate of 98% or better for the parameters considered if the distance factor is attentively selected.

For the RRT algorithm, the computational time is unaffected with the distance factor as can be deduced from Figure 9 (e). As the RRT algorithm is a sampling based algorithm the environment is not quantised and therefore the possibility of the new UAV position residing on an obstacle after the movement function is computed is not applicable if a fixed obstacle environment is assumed. Not the same can be concluded for the A* algorithm. Only the possibility that the new intermediate goal position will reside on an obstacle remain. As obstacles are planes with 2 dimensions a slight deviation on the line connecting the current UAV position to the final goal position will eliminate this conflict. Therefore in the case of RRT only one step will be required irrespective of the distance deduction (through the d_{factor} parameter) will be enough to produce an obstacle–free intermediate goal point. Therefore, ideally a 0.95 factor shall be considered not to deviate too much from the predefined d_{int_goal} distance.

H. Conclusion

This section analysed the results of the developed path planning algorithm defined in Section IV in terms of UAV speed, distance to travel per iterate, distance between current UAV position and prospective intermediate goal point, maximum intermediate time, maximum total time and distance factor for both algorithms in all 3 defined scenarios. Results show the strongholds and shortcomings of both algorithms as each of the above mentioned parameters is varied keeping other parameters constant. Furthermore, through this analysis guidelines for the definition of empirical values for the analysed parameters were set out. These empirical values can be overwritten in view of user–defined and external demands such as UAV speed.

VII. Conclusion and Future Work

The aim of this paper was to develop a platform to assess the validity of the two most utilised path planning algorithm (A* and RRT) in view of 3D UAV path planning in real–time. Literature highlight the importance of real–time path planning for autonomous 2D systems.[23–25] UAV have to manoeuvre in complex, dynamic environments and therefore the need for real–time path planning is a must. Both path planning algorithms with a common smoothing algorithm were tested in 3 different scenario with varying complexity. Real–time path planning is governed by a set of user–defined parameters such as speed and time to reach the goal node, system–defined parameters such as look–ahead distance and computational power and internal constants such as resolution/step size and the distance reduction factor. The effect of the salient parameters on performance was assessed and analysed.

Results showed that the A* algorithm outperformed the RRT algorithm in both path length and computational time for all scenarios considered, with the difference increasing with scenario complexity. Also the A* was successful by more than 90% in all tests for all scenarios considered provided the look–ahead distance is at least double the distance moved per iterate. Oppositely, the RRT algorithm resulted in a lower success rate owing primarily to the longer computational required to produce paths from the current UAV position to an intermediate goal point. The analysis presented in Section VI outline the best empirical values for each considered parameter if such parameter is not restricted by user demands or hardware limitations. In a nutshell, this analysis showed that both algorithm can be applied in real–time with different a success rate even up to 90% for all scenarios considered. It is up to the designer of the real–time 3D UAV path planning system to decide the best configuration for the requested task/s based on the analysis of Section VI.

This work can be utilised in the future to configure a real UAV for autonomous 3D UAV movement in indoor obstacle–rich environment emulating the considered scenarios. The implementation can be extended to outdoor environments where other factors such as wind and rain may influence the dynamics and sensory systems onboard the UAV. Furthermore, the performance of the real–time A* and RRT algorithm with moving obstacles is an area that will further assess the robustness of the developed real–time algorithm. The future trajectory, size and speed of such moving obstacles may be known, known with a certain degree of uncertainty or totally unknown to the path planning algorithm.

References

[1]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.

[2]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.

[3]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.

[4]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.

[5]Gurdan, D., Stumpf, J., Achtelik, M., Doth, K–M, Hirzinger, G., Rus, D. "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," *IEEE International Conference on Robotics and Automation*, Roma, Italia, 10–14 April, 2007, pp. 361–366.

[6]Franchi, A., Secchi, C., Ryll, M., Bulthoff, H. and Giordano, P (2012). Shared Control: Balancing Autonomy and Human Assistance with a Group of Quadrotor UAVs. *IEEE Robotics & Automation Magazine* [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6290692 [Accessed 26 Aug. 2018].

[7]Dittrich, J. S., Adolf, F.-M., Langer, A. and Thielecke, F. "Mission Planning for Small UAV Systems in Uncertain Environments,", *2nd European Micro Aerial Vehicle Conference*, Braunschweig, Germany, 25-26 July, 2006.

[8]Barrientos, A., Colorado, J., Martinez, A., Rossi, C., Sanz, D. and Valente, "Aerial Remote Sensing in Agriculture: A Practical Approach to Area Coverage and Path Planning for Fleets of Mini Aerial Robots", *Journal of Field Robotics*, Vol. 28, pp. 667–689, 2011.

[9]Lee, J., Huang, R., Vaughn, A., Xiao, X., Hedrick, J. K., Zennaro, M. and Sengupta, R. 'Strategies of path-planning for a UAV to track a ground vehicle", *AINS Conference*, 2003, pp. 1–7.

[10]Puri, A. 'A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance", *Department of Computer Science and Engineering, University of South Florida*, 2005.

[11]Mayerowitz, S. "Amazon.com sees delivery drones as future (Update)", *Phys.org, The Associated Press*, [Online Database], https://phys.org/news/2013-12-amazon-unveils-futuristic-mini-drone-delivery.html, [retrieved 09 September, 2018].

[12]Wright, T. "In Rural Virginia, a Drone Makes the First Legal U.S. Package Delivery", *Air & Space Smithsonian*, [Online Database], https://www.airspacemag.com/daily-planet/rural-virginia-drone-makes-first-legal-us-package-delivery-180956053/?no-ist, [retrieved 09 September, 2018].

[13]Nakashima, R. "Drone company demos how blood air-drops will work in Rwanda", *AP News*, [Online Database], https://apnews.com/e5336fa71af347db99e11cd69ab16054/drone-company-demos-how-blood-air-drops-will-work-rwanda, [retrieved 09 September, 2018].

[14]Nakashima, TU Delft. "TU Delft ambulance drone drastically increases the chances of surviving cardiac arrest", *TU Delft*, [Online Database], https://www.tudelft.nl/2014/tu-delft/ambulance-drone-tu-delft-vergroot-overlevingskans-bij-hartstilstand-drastisch/, [retrieved 09 September, 2018].

[15]Call, B. (2006). *Obstacle avoidance for unmanned air vehicles.* Master Dissertation. Brigham Young University.

[16]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.

[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]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.

[19]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.

[20]Bollino, K. P. and Ryan Lewis, L., "Collision-free Multi-UAV Optimal Path Planning and Cooperative Control for Tactical Applications", *AIAA Guidance, Navigation and Control Conference*, Honolulu, HI, 21–24 Aug., 2008, pp. 1–18.

[21]Zammit, C. and van Kampen, E. J., "Comparison between A* and RRT Algorithms for UAV Path Planning", *AIAA Guidance, Navigation and Control Conference*, AIAA SciTech Forum, Kissimmee, FL, 8–12 Jan., 2018.

[22]Zammit, C. and van Kampen, E. J., "Advancements for A* and RRT in 3D path planning of UAVs", *AIAA Guidance, Navigation and Control Conference*, AIAA SciTech Forum, San Diego, CA, 7–11 Jan., 2019.

[23]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.

[24]Bethke, B., Bertuccelli, L. How, J. P., "Experimental Demonstration of MDP–Based Planning with Model Uncertainty," *AIAA Guidance, Navigation and Control Conference*, Honolulu, HI, 18–21 Aug. 2008, pp. 1–22.

[25]Bollino, K., Lewis, L. R., Sekhavat, P. and Ross, I. M., "Pseudospectral Optimal Control: A Clear Road for Autonomous Intelligent Path Planning," *AIAA Infotech Aerospace Conference and Exhibit*, Rohnert Park, CA, 7–10 May 2007, pp. 1–14.

[26]Frazzoli, E., "Maneuver–Based Motion Planning and Coordination for Multiple UAVs," *Proceedings of the AIAA/IEEE Digital Avionics Systems Conference*, Portsmouth, VA, 20–23 May 2002, pp. 1–11.

[27]Benenson, R. Petti, S., Fraichard, T. and Parent, M., "Integrating Perception and Planning for Autonomous Navigation of Urban Vehicles," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 9–15 Oct. 2006, pp. 98–104.

[28]Yao, P., Wang, H. and Su, Z., "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerospace Science and Technology*, Vol. 47, pp. 269–279, 2015.

[29]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.

[30]Karaman, S. and Frazzoli, E., "Sampling-based algorithms for optimal motion planning", *The International Journal of Robotics Research*, Vol. 30, No. 7, pp. 846–894, 2011.

[31]Singh, Y. and Sharma, S., "Optimal path planning of unmanned surface vehicles," *Indian Journal of Geo-Marine Sciences*, Vol. 47, No. 7, pp. 1325–1334, 2018.

[32]Ross, I. M., "A Unified Computational Framework for Real-Time Optimal Control," *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003, pp. 2210–2215.

[33]Gong, Q., Kang, W. and Ross, I.M., "A Pseudospectral Method for the Optimal Control of Constrained Feedback Linearizable Systems," *IEEE Transactions on Automatic Control*, Vol. 51, No. 7, pp. 1115–1129, 2006.

[34]Gao, X., Ren, J. and Chen, D., "Developing an effective algorithm for dynamic UAV path planning with incomplete threat information," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 226, No. 4, pp. 413–421, 2012.

[35]Peng, X. and Xu, D., "Intelligent online path planning for UAVs in adversarial environments," *International Journal of Advanced Robotic Systems*, Vol. 9, pp. 1–12, 2012.

[36]Roberge, V., Tarbouchi, M. and Labonte, G., "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions On Industrial Informatics*, Vol. 9, No. 1, pp. 132–141, 2013.

[37]Lavalle, S. M., "Motion Planning : The Essentials", *IEEE Robotics & Automation Magazine*, Vol. 18, No. 1, pp. 79–89, 2011.

[38]Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., and How, J. P., "Motion Planning in Complex Environments using Closed-loop Prediction", *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, HI, 18–21 Aug. 2008, pp. 1–22.

[39]Fernandes, E. Costa, P., Lima, J. and Veiga, G. "Towards an orientation enhanced astar algorithm for robotic navigation", *IEEE International Conference in Industrial Technology (ICIT)*, Amman, Jordan, 12–15 May, pp. 3320–3325, 2015.

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

[41]Palossi, D., Furci, M., Naldi, R., Marongiu, A., Marconi, L., Benini, L. "An energy-efficient parallel algorithm for real-time near-optimal UAV path planning", *Proceedings of the ACM International Conference on Computing Frontiers*, Como, Italy, 16–18 May, pp. 392–397, 2016.

[42]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.

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

[44]Kunz, T. Reiser, U., Stilman, M. and Verl, A. "Real-time path planning for a robot arm in changing environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 8–14 Oct. 2010, pp. 5906–5911.

[45]Stentz, A., "Optimal and Efficient Path Planning for Partially Known Environments", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA, 8–13 May, 1994, pp. 3310–3317.

[46]Petti, S. and Fraichard, T., "Safe Motion Planning in Dynamic Environments", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alta., Canada, 2–6 Aug., 2005, pp. 2210–2215.

[47]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.

[48]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.

[49]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.

[50]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.

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

[52]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.

[53]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.

[54]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.

[55]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.

[56]Hrabar, S., "3D Path Planning and Stereo-based Obstacle Avoidance for Rotorcraft UAVs", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 22–26 Sep. 2008, pp. 807–814.

[57]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.

[58]Yu, H., Beard, R. W. and Byrne, J. "Vision–based Navigation Frame Mapping and Path Planning for Micro Air Vehicles", *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinios, 11–13 Aug. 2009, pp. 1–10.

[59]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.

[60]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].