3D real-time path planning of UAVs in dynamic environments

Zammit, C.; van Kampen, E.

DOI
10.2514/6.2021-1955

Publication date
2021

Document Version
Final published version

Published in
AIAA Scitech 2021 Forum

AIAA SciTech Forum
11–15 & 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-1955

3D real–time path planning of UAVs in dynamic environments

C. Zammit* and E. van Kampen†
Delft University of Technology, Delft, 2629HS, The Netherlands

Unmanned Aerial Vehicles (UAVs) are taking active roles in personal, commercial, industrial and military applications due to their efficiency, availability and low–cost. UAVs must operate safely and in real–time in both static and dynamic environments. An extensive literature review, defines the dynamic environment term, the need for dynamic path planning and reviews different solutions. This paper presents a 3D real–time path planning platform to assess the performance of the A* and RRT algorithms. Four test scenarios with varying difficulty are constructed consisting of V–obstacles, cubes and 2D planes moving at time–varying speed, direction and orientation. Two rationales to either wait or move further in the direction of the goal when an intermediate goal point is not available are considered. Results show that for both A* and RRT the moving variant case performs better especially in complex scenarios. RRT performs better in simple scenarios and complex scenarios at low speed while A* performs better at high speeds in complex scenarios. A success rate of over 95% is recorded for three scenarios for all considered speeds and for both algorithm.

I. Introduction

The ever increasing availability of different Unmanned Aerial Vehicles (UAVs) for a wide range of personal, commercial, industrial and military applications, is increasing the need for robust, reliable and autonomous guidance, navigation and control systems that must operate in real–time even within obstacle–dense environments. The environment in which a UAV is operating may contain fixed and/or moving obstacles that may vary in size, speed and orientation. Furthermore, these sometimes time-variant characteristics, can change as the UAV is navigating to reach a pre–defined goal. Path planning algorithms are therefore required to navigate the UAV in such time-varying environment with the available computational, sensory and fuel resources available online.

In real indoor and outdoor environments, the UAV path planning algorithm must generate or update a path in real time to reach an intermediate or final goal position. These systems mainly rely on the real-time data of onboard sensory systems with their associated inaccuracies, latency and uncertainty to update obstacle sizes and positions, intermediate goal and current UAV positions. In dynamic environments the future size, position, speed and orientation of dynamic obstacles may be known or unknown beforehand. In the latter case, the UAV can only rely on the real-time data of sensory systems. But in the former the path planning algorithm can have apriori knowledge of future obstacle positions for example automatically operated doors or gates. Furthermore, in the same environment in which the path planning algorithm is guiding the assigned UAV, other UAVs may be operating. Their current and future assigned path maybe fixed as their task would be very specific. For example, in a hospital environment delivering blood samples from the laboratory to the consultants office is normal practice. In such cases a preassigned path maybe available or not to the path planning algorithm apriori or in real-time. Oppositely other UAVs operating in the same environment may operate independently or worst enemies.

In our previous work[1,2] fixed obstacle 2D planes with tight windows were considered to assess the performance of the two most utilised algorithms in the graph–based and sampling–based categories namely the A* and Rapidly–Exploring Random Tree (RRT) algorithms which will be briefly explained in Section III.

*Ph. D. Candidate, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands.
†Assistant Professor, Control & Simulation, TU Delft Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands, and AIAA Member.

The next natural step owing the positive performance results of both algorithms is to assess their validity in the dynamic environments described earlier. A valid path planning system shall be able to ensure that the path to a goal position is *always* obstacle free. In this light, the path planning algorithm must be integrated within a real–time or online platform. Such system shall re–assess already re–assigned paths and if necessary re–evaluate to ensure that a path from the current UAV position to a goal position is obstacle free based on real–time environmental awareness whilst the UAV is traversing the respective path segment, if and only if a path exists. Such an assessment platform was developed in our previous work[3] and will be briefly explained in Section III.

The aim of this paper is to assess the performance and consequently the validity of the A* and RRT algorithms for real–time 3D UAV path planning in dynamic environments. For the scope of this paper a single agent path planning system with no *a priori* knowledge of the obstacles paths, including other UAVs, shall be considered. The path length, computational time and success rates will be the performance measured considered to assess the usability of both the A* and RRT algorithms in dynamic environments. In the context of this work, a dynamic environment will contain both fixed and moving obstacles with the latter having either a time–variant size, position, speed or orientation or any combination of these possibilities. Furthermore, in this context, a real–time path planner is required to generate a path in the same or less time than the time required by the UAV to traverse the same path.[4–6]

The paper will be organised as follows. Section II will present the state-of-the-art in path planning in the presence of dynamic obstacles. Section III provides a brief resume of the A* and RRT algorithms, the smoothing algorithm (a post–path generation algorithm applied to the RRT algorithm) and the real–time path planning platform all extensively defined in our previous works.[1–3] Section IV will define the obstacle generation algorithm based on pre–defined static and dynamic obstacle characteristics. Section V will describe the amendments to the developed real–time algorithm developed in our previous work.[3] This will be followed by Section VI which will present and analyse the results based on pre–defined performance measures in view of real–time 3D UAV path planning. Finally, this paper will conclude (Section VII) with a resume of the performance results highlighting strongholds and shortcomings of both algorithms and integrated system's applicability to real 3D UAV path planning in dynamic environments for both algorithms. Furthermore, Section VII identifies future work that can be developed to implement this system in a real UAV environment.

II. Path planning in dynamic environments review

A. Introduction

Autonomous path planning is the process of automatically generating a feasible path to the final goal point even in the presence of static and dynamic constraints, obstacles and threats (COT).[1,7] Therefore, a sound path planner must always guarantee that the UAV will reach the goal node and remain at the goal point unless given instructions otherwise in the presence of COTs and irrespective of sensing and control uncertainties.[8] Some path planners focused on path optimisation in an obstacle–free environment[9–11] but in real situations the environment incorporates different COTs.[12–14] This review will analyse dynamic environmental modelling with its various obstacle definitions in the context of real–time 3D UAV path planning. It is not the scope of this work to review path planning of UAVs. For an extensive literature review on different path planning solutions refer to Zammit *et. al.*[1]

B. Dynamic Environment Definition

A dynamic environment may include a vast spectrum of static and time–varying constraints, obstacles and threats which can 'pop–up' whilst the UAV is navigating through such an environment.[12,15,16] These include wind, fuel, altitude constraints, flight profile requirements, UAV kinematic and dynamic holonomic and non-holonomic constraints, no fly zones, buildings, vehicles, changing weather conditions, control failures, moving targets, addition/removal of COTs, surface to air missiles, tanks, lane markings, irrigation system, other aircraft or UAVs and quality, loss or delay in communication between agents in a swarm setup.[7,15–25] Not all COTs incorporate the same level of complexity. Goerzen *et. al.*[8] defined obstacle–complexity in terms of obstacles edges and vertices, number of obstacles in a specific area and memory storage required for obstacle representation and acknowledged that is a commonly used metric. Fujimura[26] identified three categories of dynamic environments: Moving obstacles in a known environment, Static obstacles in an

unknown environment and Moving obstacles in an unknown environment. A combination of two or even all situations requires also dynamic motion planning.

C. The need for dynamic path planning

Path planning can be segmented into two main categories: global and local path planning.[27–29] A global planner generates a low-resolution, long range offline path based on a fidelity situational awareness. This path planning strategy is inadequate for a time–varying environment or in situations where obstacles, threats or constraints are unknown *apriori*. In these situations the path is optimised prior the execution of a path.[27,30] Oppositely, a local planner creates higher resolution paths in a smaller environmental space based on real–time information from on-board sensors. This reactive path planning approach is ideal for dynamic environments.[27,30] A hybrid approach is suggested to outweigh the weakness of one method with the strongholds of the other.[30–33]

A review on path planning over a span of 15 years (2000–2015) presented by Mac *et. al.*[27] showed that in 49% of the path planning projects, the obstacles and targets were considered to be static while in only 18% of the projects considered a combination of static and dynamic obstacles. 11% considered a dynamic target with 9% of which considered adaptive UAV velocity.[27] So although different studies considered dynamic COTs, the majority considered the agent to move at constant speed.[27,34] This highlights the need for path planning in dynamic environments, although path planning in dynamic environments is still considered a challenging aspect for researchers.[27]

In a dynamic and/or unknown environment, an autonomous path planning algorithm must react in real–time or over a predefined time window to re–plan utilising local onboard system information a collision free path to goal based on new previously unknown COT without the assistance of an operator or guidance ground systems.[8,12,15,16,21,24,35,36] This *obvious* requirement,[12] is a must for realistic path planning applications.[37] Such path planners are referred to as reactive path planners.[8,35,36,38] According to *Goerzen et. al.*[8] such planners are ideal to mitigate with rapid time–variant environment but suffer in global planning problems finding difficulty in reaching the final goal position and therefore lacking in path length optimisation. Although the future position of obstacles can be estimated in certain situations, such planner is required to ensure non–collision in a given time window if these obstacles change state in an unexpected manner.[39]

Ensuring safety in an unknown dynamic environment for a UAV operating in real–time is an important requirement.[24] Kuwata *et. al.*[24] defined safety as a state into which a vehicle can remain indefinitely without colliding with static and moving obstacles or breaching constraints, assuming a constant heading by moving obstacles. In terms of safety, both control and obstacle avoidance systems must prove that they can reach the goal node in all possible vehicular and environmental variations.[14] Without these capabilities the use of unmanned vehicular system will be highly restricted to time–invariant, *a priori* defined environments.[14]

For realistic path planning the UAV must be equipped with an advanced sensory system for environmental sensing that shall be able to detect new COTs as they appear in the scenario space.[12,13,16] The sampling rate of such system is inversely proportional to the distance at which COTs must be detected.[25] Furthermore, the path planning system must be provided with appropriate COT representation and mapping through mathematical models that incorporate kinematic and dynamic characteristics of obstacles with minimal computational demand and a fast update rate to immediately detect changes in the environment in harmonisation with the onboard environmental characteristic database.[21,34,35,40] This stresses the need for dynamic resolution that will be able to automatically adjust to handle different COTs at different distances from the onboard sensing system so as to make optimal use of computational resources while accurately defining obstacles eliminating the risk of leaving small passages between obstacle available.[41] Moreover, COT representation shall make use of approximations to mitigate with memory and computational power limitations.[8] The control algorithm must also react simultaneously with the path planning algorithm to avert collisions, stressing the need for an online implementation.[36] These requirements highlights the difficulty for a UAV to operate in unknown areas. In fact, Dittrich *et. al.*[12] remarked that the best solutions have yet to be found.

D. Environmental Assumptions

It is assumed that the environmental space is made up of two disjoint subsets of either free space or obstacle space.[8] Different research works consider either that the environmental characteristics including obstacle

geometry and locations are known *a priori*[15,42] or oppositely no *a priori* knowledge of the environment is provided to the path planning algorithm[13,24,43] or a combination of both of the above.[16,35,44]

In the first case, COT information is provided as the exploration area expands due to moving target.[15] In the second case, the path planner can initially assume an obstacle–free environment with obstacles known only within a finite receding–horizon, re–planning a new non–colliding path when a new COT is detected from sensory systems that violates the previously generated path.[13,21,24,43] This option is ideal for surveillance, moving targets and high time–variant environments[24] In the third case, paths are generated offline based on known environmental COT, triggering reactive re–planning in the eventuality of randomly–generated COT pop–ups that will result in a collision or non–optimal paths if the offline generated path is not updated.[12,16,35,41,44–46] In high time–variant dynamic environments the planner has to re–plan frequently demanding more computational power consequently limiting the ability of the path planning algorithm to be applied online.

E. Constraints, Obstacle and Threats (COT) Sensing and Modelling Systems

Computational resources are very limited in real–time implementations and their efficient use is key for realistic implementations.[35] Therefore, COT modelling for dynamic re-planning in dynamic environments require efficient storage, fast addition and deletion and fast and efficient collision checks.[40] These requirements depend also on obstacle density which is dependent upon the environment into which the UAV will operate. For example, Koenig *et. al.*[47] considered an obstacle density of 30%. Amin *et. al.*[35] remarked that COT representations for unmanned vehicles in 2006 were in the majority not suitable for real–time implementations such as[48,49] and suggested that systems developed for the video gaming industry shall be considered.

Obstacle representation formats can be classified into 3 main categories: (1) Raw Data (e.g. vertices-edges sets); (2) Bounding Volumes (e.g. Oriented Bounding Boxes (OBB) and their variants) and (3) Spatial Partitioning (e.g. Grids, Quadtrees and Octrees).[35,50]

1. Raw Data

Stereo vision was used to build 3D occupancy maps that allowed dynamic re–planning to be realised.[51] Dittrich *et. al.*[12] utilised also a stereo camera based system to construct polygonal lines or circles with a minimal safety distance from each detected obstacle. Besides stereo vision, Scherer *et. al.*[52] and Shim *et. al.*[53] utilised laser range finders for obstacle detection and re-planning. Owing to the weight of laser range finders, Hrabar[51] remarked that stereo vision offer the best solution for small UAVs although such obstacle detection system shall be capable of defining absolute range measurements.

Similarly, a depth map can be constructed using computer vision algorithms to define the time to collision estimates.[43] Consequently, an obstacle map can be built in local frame of reference as a depth map can provide the range to the obstacle derived from the airspeed and bearing to obstacle from the UAV current position.[43] Furthermore, Ya *et. al.*[19] constructed a 500x500 cell environment utilising a variable probability function (0% to 50%) that randomly places obstacles within each cell.

2. Bounding Volumes

Gottschalk *et. al.*[54] modelled COTs using OBBs for fast and reliable dynamic collision identification in a 2D environment. Similarly, Gros *et. al.*[21] utilised the same obstacle modelling technique with a time-invariant COTs of fixed radius of 150m. Results in a high density COT environment showed that real–time can be achieved with a Finite Receding-Horizon Incremental-Sampling Tree (RHIST). Similarly, Ögren *et. al.*[55] considered a constant obstacle margin of 15m was considered although in sparse environments this margin was increased to 50m.

Foo *et. al.*,[7] defined a threat zone as a sphere for above ground and a hemisphere for a ground vehicle of user–defined radius surrounding an obstacle. A radius of 20,000ft. was arbitrary selected for such threats. The radius tolerance was reduced by a factor of 4 times for non-enemy UAVs.[7] Similarly, Srikanthakumar *et. al.*[14] considered spherical obstacles with a safe radius greater than the actual radius by 25% and an variable influence range with a repulsive potential to divert the UAV from obstacles. Also, collision cones are constructed for every obstacle which is also surrounded by a safety ball of 5m–20m radius.[36] Similarly,

Ok *et. al.*[56] defined a low-level planner that adds repulsive forces to obstacles governed by a higher level Generalized Vonoroi Diagram to construct a collision–free path in the presence of uncertainty.

Bollino *et. al.*[15,57] defined obstacles by simple shapes including squares, diamonds, circles, ellipses and rectangles using the p-norm formulation. Similarly, to Foo *et. al.*[7] and Srikanthakumar *et. al.*,[14] Bollino *et. al.* defined a distance–dependent buffer to mitigate with uncertainty in obstacle modelling.[15] Such buffer increases with increase in distance to the obstacle from the UAV so as to emulate the uncertainty increases as the UAV moves away from obstacles.[15,58] Similarly, Likhachev *et. al.*[44] considered a buffer zone around obstacles with high costs that required the planner to stop and conduct 90 degree turns to avoid obstacles.

Adolf *et. al.*[13] considered an incremental heuristic path planner for *a priori* known COTs and a reactive anytime path planning algorithm for incremental obstacle pop–ups which were defined by 3D non–convex polyhedral surfaces. A 3D voxel grid was used to index and update polygonal pop-up COTs. The helicopter model was considered to be holonomic and the 3D path planning algorithm exhibited exponential runtime complexity.[13] Consequently, anytime planners may lack in constantly generating new global paths whenever the environment changes.[44] Similarly, Lee *et. al.*[16] designed static obstacles using polygons, moving obstacles using rectangular no–fly zones and pop–up threats as circles with the latter designed using the Markov Chain method. Through a Model Predicative Control method the agent was able to manoeuvre in a 9 pop–up threat environment with 5 targets and 5 static obstacles. Also a first order Markov Chain was developed in[45,59] to generate the sequence of pop–ups.

3. Spatial Partitioning

Visibility maps builds lines with all possible vertices from the agent position by considering that the shortest path touches polygonal obstacles at their vertices.[8] The Edge–Sampled Visibility Graph, a variant of the primary method, defines a minimum edge length to build a visibility graph by assigning multiple vertices along edges of polyhedral obstacles.[8] Kim *et. al.*[60] proposed the Quantized Visibility Graph (QVG) to model polygon shaped obstacles. In a hybrid approach, the Freeway Method used bounding cylinders around obstacles to build a map of lines. Results show that it is not limited to 2D but is incomplete and non–optimal.[8] Oppositely, the Silhoutte Method is complete for any obstacle geometry with any dimension.[8]

The visibility method in conjunction with a sparse uniform space sampling algorithm was developed by Tsardoulias *et. al.*[61] to model COTs. Sampling–based methods does not require explicit construction of obstacles when opposed to more deterministic approaches reducing computational time.[62] Results showed that the A* algorithm for 3D path planning in combination with visibility graph for polygonal obstacles modelling did not guarantee the shortest paths.[63] Moreover, Visibility and Voronoi diagrams can only generate shortest paths with optimal clearances in low–dimensional spaces with only polygonal obstacles.[64,65] Voronoi diagrams were mainly used for static obstacles although Roos *et. al.*[66] added bounds to Voronoi channels to mitigate with dynamic COTs in 2D environments. Such approach can be extended for 3D environments.[67]

A Quadtree data structure was utilised by Amin *et. al.*[35] for obstacle representation. This was integrated with a modified RRT algorithm for path planning. Another variant is the grid map decomposition technique that offline defines obstacle–free rectangles and replaces all nodes within with edges so as to improve path optimisation.[68]

F. Solutions

Numerous researchers have tried to use classical approaches such as classical algorithms such as cell decomposition, potential field, sampling–based and sub-goal networks to achieve real–time path planning that can then be applied for dynamic re-planning.[27,69–71]

Graph–based approaches such as the Sparse A* Search path planning algorithm is also a potential candidate for path planning in dynamic environments.[72–74] To mitigate with the static nature of the A* algorithm, a mechanism named the Virtual Force was proposed for dynamic re–planning.[75] Likhachev *et. al.*[44] successfully implemented an A* based anytime algorithm in a time–variant obstacle where targets are randomly generated. The trajectory planner was tested in a 500mx500m with a constant speed of 5m/s. Trajectories included both parking, reverse manoeuvring in a dense environment.

Sampling–based methods were also proposed by a number of researchers for dynamic re-planning. Such methods are considered due to their asymptotic optimality, efficiency although they cannot guarantee an optimal solution.[27,76,77] Hsu *et. al.*[39] developed an online Probabilistic Roadmap (PRM) that generates

a new path in a predefined time window when obstacle motion differs from estimated during execution. Similarly, Otte *et. al.*[78] developed an asymptotically optimal re–planning algorithm (RRT^X) that updates a goal rooted tree when new obstacles are detected. During re–planning the environment is assumed static.

Heuristic methods, such as Artificial Neural Network, Fuzzy Logic and Nature–inspired Methods can also generate optimal path planning in uncertain, partially unknown and dynamic environments.[27] However, these methods require a learning phase and high computational demand. The latter is highly limited in real–time applications especially for small UAVs.[27]

Different researchers model various different static and dynamic environments with different degrees of complexity. Yu *et. al.*[43] defined a 40mx10mx6m environmental space furnished with three poles of 3, 4m poles, a 1.5m wall and a rectangular obstacle. It was concluded from this study that on average 0.15s were required to re-plan a new non-colliding path when new obstacles were detected. Bollino *et. al.*[57] considered a 32 obstacle 2D environment into which the planner was allowed to travel backwards, solving problems that otherwise would lead to no solutions. U-turns and backwards movements were also considered in.[25]

A different approach is the velocity obstacle concept that assumes known velocities of obstacles and considers a range of possible agent velocities based upon a predefined maximum acceleration was used to generate collision paths offline.[12,79,80] This concept assigns discretises velocities and assigns associated costs based on the vicinity to nearby velocity obstacles.[12,79] Similarly, Ögren *et. al.*[55] considered a variable agent speed with a maximum acceleration of $30m/s^2$ and a maximum speed of 100m/s with moving obstacles at constant direction and speed of 30m/s. Results show that by increasing horizon lengths performance is enhanced for both static and moving obstacles. Others developed the Directional Priority Sequential Selection[81] and Predictive Trajectory Planning algorithms[38] for 2D reactive trajectory planning in dynamic environments.

In the majority of the studies constant speed scenarios are considered.[34] For inspection purposes, UAV typically fly at low speed, lower than 5m/s or just hover in situations when they are acquiring images.[13,51] Such situation differs from high altitude problems as the agent is operating close to ground.[12]

The pre–defined time window requirement of reactive obstacle avoidance was defined by Chawla *et. al.*[36] at 4s to 8s using a partial integrated guidance and control approach using a real six–DOF model that executed in a single loop. It must be pointed out that such parameter increases significantly from 2D to 3D in fact, path planning in 2D requires polynomial time while in 3D the solution is NP–hard for polygonal and polyhedral obstacles respectively.[82,83]

Bohren *et. al.*[25] developed a sensory system that is able to provide a sensing range varying from 4 to 60m using 90 degree field of view sensors at rate of 10Hz. Results in a 300x300m map showed that vehicles were detected out to 60m with an accuracy of 1m/s and ground points within a 30m range in good conditions and depending on ground reflectivity. Furthermore, Benjamin *et. al.*[84] implemented wireless communication to provide the agent with real–time 2D obstacle information for real–time obstacle avoidance in marine Unmanned Surface Vehicles (USVs). In the same field, Larson *et. al.*[42] developed real–time obstacle avoidance using the projected area method.

Different performance measures are used to assess the validity of different path planning algorithms. In our previous work, path length and computational time were considered.[1–3] Besides these two parameters in dynamic environment the clearance criterion i.e. the minimum distance from the UAV to the COT was considered as a performance measure.[14,36] Simulations show that at least five times the radius of a surrounding ball around the obstacle is required for safe operation.[36] Furthermore, another performance measure is the deviation to the goal point for the mission to be successful. In this regard, Chawla *et. al.*[36] considered a maximum deviation of 0.5m radius around the goal position.

G. Conclusion

This extensive literature review initiated with an overview of what different researchers considered as a dynamic environment. Such environment constituted time–invariant and time–variant COTs that are either known, partially known or unknown to the path planner *a priori*. The need to generate a reactive real–time path in such an environment was discussed in view of realistic UAV applications. This highlighted the importance of the UAV to be equipped with all the reliable sensory systems and adequate computational power. Then the review discussed the different COT sensing and modelling systems considered by different researchers to best represent the dynamic environment in view of computational demand limitations and efficient path planning. Finally, the most promising path planning solutions proposed by different researchers in 2D/3D dynamic environments in different dynamic environments emulating reality even with USVs and

robots as agents were reviewed and discussed. This review will form the basis of our 3D real–time path planning in a dynamic environment that will be discussed in the next sections.

III. A*, RRT, Smoothing and Real–time Algorithms

A. Introduction

This section will briefly describe the two most utilised graph–based and sampling–based methods, namely the A* and RRT algorithms. Then the smoothing algorithm employed to smoothen the non–optimal path generated by the RRT algorithm will be explained. This section will conclude with a resume of the implementation of both path planning algorithms emulating real–time situations.

B. The A* algorithm

Graph–based methods define the state space into an occupancy grid defining obstacles residing in grid points as inaccessible points. Based on the free grid points, the graph–based algorithms check whether a path connecting the start and goal position exists.[85] Graph–based algorithms only offer a guarantee of solution if an adequate resolution is selected.[77]

The standard A* algorithm uses a heuristic evaluation function ($f(n)$) to determine the cost of neighbouring grid points.[22] This evaluation function sums the cost from the current position to a prospective future position and the cost from the latter to its goal node.[22,86] For a detailed explanation of this algorithm refer to our previous work.[1–3]

C. The Rapidly–Exploring Random Tree (RRT) Algorithm

Sampling–based methods generate a path by connecting unevenly selected obstacle–free points in the configuration space.[37,77] As opposed to graph–based methods, such algorithms can generate a path within infinite time provided that a path exists.[77]

The standard Rapidly–Exploring Random Tree (RRT) constructs a unidirectional tree by randomly selecting obstacle–free points. A new tree branch is defined a predefined distance from the nearest point on the tree if a direct path to the latter does not collide with an obstacle. A path is formed when one of the tree branches reaches the goal node and another connects to the start point.[87–89] Such algorithm is efficient in complex high–dimensional environments although the non-optimal path generated by this algorithm may require smoothing.[64,89,90] As for A* refer to our previous work for a more detailed explanation of the RRT algorithm and its variants.[1–3]

D. The Smoothing Algorithm

The smoothing algorithm randomly selects two path points and then randomly defines two points on the path segment connecting the formerly selected points and their respective next path points. If the interconnection of the latter two points results in an obstacle–free line, then all points between these two points are neglected from the path. This process is repeated until the percentage path length reduction over the last 20 iterations is less than 1% and for at least 20 iterations. Please refer to our previous work for a more detailed explanation and for assessing the algorithm's effect on path planning performance.[2,3]

The smoothing algorithm is developed to target the non–optimality of paths generated by the RRT algorithm. Oppositely, the A* algorithm generates the shortest available path based on the considered resolution. Therefore the smoothing algorithm can only improve the path by eliminating the grid and assumes that each point in the free space can be used to smoothen the path. From our previous work[1,2] it was concluded that the improvement is marginal. Furthermore, the implementation of the A* algorithm in real–time situations can lead to non–colliding smoothed path points which are very near to obstacles or reside on obstacles due to different frame of reference in the next iterate when the start point is moved further into the path. Such situation can result in a collision.

E. The Real–time Algorithm

As remarked earlier, real–time path planning is fundamental for a UAV to reach the final goal position in a dynamic environment.[57,91,92] In this light, an algorithm to emulate real–time behaviour was developed in

our previous work.[3]

In a nutshell, this algorithm defines an obstacle–free intermediate goal point in the direction of the final goal a predefined distance from the current UAV position based upon the sensory system's Field of View (FOV). An intermediate path, if possible, is generated by the A* or the RRT algorithm from the current position to the intermediate goal point. Consequently, the UAV's new position is defined a predefined distance along the generated path. This distance is selected based upon the UAV speed and maximum allocated intermediate time and assuming that actuator systems are defined with high fidelity and the UAV is not effected by external factors. Owing that the time needed for the UAV to travel this arbitrary distance is very low it was assumed that the environment will remain static in this time frame. The algorithm has a two layer time limitation one to generate an intermediate path and another for the total duration of the mission. Review our previous work[3] for a thorough explanation, parameter definition and assignment and performance results with both A* and RRT algorithms.

F. Conclusion

This section provided an overview of the algorithms that are utilised to generate a real–time path in a dynamic environment. The A* and RRT algorithms formed the backbone of the path planning algorithms. The smoothing algorithm is considered to mitigate with the non–optimality of the RRT algorithm. Finally, the real–time algorithm provided a platform to assess the performance of both path planning algorithms in situations derived from real life UAV applications.

IV. The obstacle generation algorithm

A. Introduction

The scope of this algorithm is to serve as a platform where different obstacles derived from real–life situations but not only are modelled with custom user–defined fixed and time–variant characteristics to assess the validity of the holistic real–time path planning algorithm. Although in this case the A* and RRT algorithms are considered, this algorithm is modular enough to test any path planning algorithm using either the previously developed real–time algorithm or any other platform.

The initial and future environments are generated prior the initiation of the real–time path planning tests. In real–time path planning the environment is generated by the sensing and modelling system which is independent of the path planning system. The computational demand required by the sensing and modelling system is beyond the remit of this work and may vary depending upon the software and hardware utilised. The maximum computational time to generate an intermediate path was selected in view of the UAV speed, resolution and environmental size.

B. Theoretical Rationale

As derived from literature, in real–life situations every obstacle can be approximated by a regular shape.[7,12,14,21] The algorithm initiates with retrieving from a predefined file the shape and size of each obstacle that will be present initially and at a future time in the environment. The obstacle position in the obstacle characteristic file was initially set at the middle of the environmental space. Each shape is modelled through a finite number of planes which are interconnected with each other to form closed or open shapes. This shape is then replicated for a predetermined number of times and each copy is randomly placed in the environmental space. In case that also rotation is considered, besides the random placement, a rotation by a random value different for all 3 dimensions about a random line is performed for each replica.

The next step is to differently shift and if requested rotate each of the generated replicas by a random distance in a random direction for a finite number of times. In real terms, this implies variable speed, roll, pitch and yaw obstacles. Each time all obstacles within the environmental space are shifted implies that a predetermined amount of time has elapsed. This time is equal to the time required by the UAV to traverse from the current point to a new point on the intermediate path. The distance moved by each obstacle in the mentioned time is less than the distance to be covered by the UAV in the same amount of time as otherwise the UAV will not be able to avoid the moving obstacle. Each environmental space is allocated a time stamp to harmonise with the real–time algorithm. Obstacle are allowed to move in and out of the environmental space and to combine into one another so as to emulate real–life situations.

C. Implementation

In the implementation three shapes were considered, namely a cube, the V obstacle and 2D planes with small windows. The characteristics of each are tabulated in Table 1. Four different scenarios with increasing complexity are considered, namely:

- Scenario 1. 10 cubes of 0.1x0.1x0.1 with no rotation;
- Scenario 2. 10 cubes the same size as Scenario 1 but with random rotation at definition stage and with changing independent rotation with time iterates;
- Scenario 3. 10 V obstacles constructed by adjoining one side of each of the two planes with an angle of 53° between the planes. Each plane has a size of 0.1x0.112 and is randomly rotated as in Scenario 2. This plane size was considered so that it fits exactly into the considered cube; and
- Scenario 4. Two planes on the Y–Z axis separated by 0.4 each with a window of 0.2x0.2 as well as the obstacles in Scenarios 2 and 3 combined.

Figure 1 illustrates pictorially each scenario for a random time iterate. The positions of each obstacle for each scenario will change in every time iterate as described earlier.

Table 1. Obstacle shapes

Shape	Size	Number of Planes	Closed/Open
Cube	0.05x0.05	6	Closed
V obstacle	0.05x0.05	2; (90^o with each other)	Open
Plane with window	1x1	1; window (0.05x0.05)	Open

D. Conclusion

This section provided an overview of the rationale and implementation of the dynamic obstacle generation algorithm. This modular algorithm was designed to emulate real–life situations and the user can design the environment based on the already considered obstacles or new ones. Furthermore, the number, characteristics, rotation and movement can be individually defined in a time–variant nature. In flight pop-ups and real–time obstacle elimination can also be modelled.

V. Enhancements to the real–time path planning algorithm

The real–time algorithm thoroughly explained in[3] was adapted to integrate the obstacle generation algorithm described in Section IV. Table 2 adapted from our previous work[3] tabulates the nominal parameter values that are considered as constant for the scope of this paper. Each parameter is defined in a modular way based on the rationale described in our previous work.[3] All parameter assignments considered a 1x1x1 environment. Furthermore, for A* the environment and start and goal points were shifted by a random distance between 0 and half the distance between grid points to eliminate path length ripple as thoroughly explained in.[2]

The resolution is defined as the amount of grid points per dimension. The RRT is a sampling–based method and therefore every point in the available space is available. The limitation is the distance the planner can move during intermediate path construction in the generated path direction. The distance to travel per iterate and the distance between the current UAV position and an intermediate goal point are denoted by d_{s_step} and d_{int_goal}, respectively. The maximum time to generate an intermediate path and the maximum time to reach the goal from start position are denoted by $t_{iterate\_max}$ and $t_{path\_gen\_max}$, respectively. Finally, the distance reduction factor (d_{factor}) is considered to reduce the distance to the intermediate goal point and the new UAV position as the latter may reside on an obstacle.

The UAV speed which was considered to be constant during the traversal of the path was considered as variable for both algorithms and all scenarios. For the analysis this modular parameter between 0.01[-]/s and

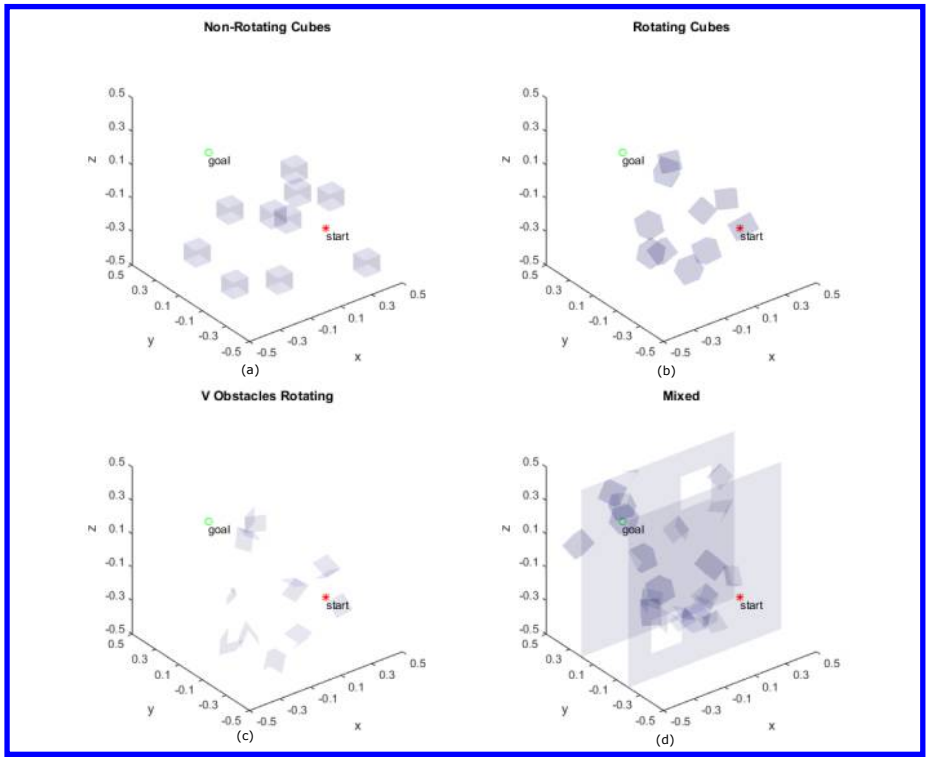

Figure 1. Environmental scenarios: (a) Non-rotating cubes scenario, (b) Rotating cube scenarios, (c) V obstacle rotating scenario and (d) Mixed scenario. These scenarios incorporate cubes, V obstacles and obstacle planes in the Y-Z with windows as openings.

0.1[-]/s in steps of 0.01[-]/s, where [-] represent modular distance units. A $5000 \times 5000 \times 5000$ distance unit environment was considered. As described in,[3] these values were determined based on the nominal speeds for exploration situations in a nominal environment.

Table 2. Real–time algorithm parameter definition

Parameter	Nominal Value	Units
Resolution (res)	21	[-]
Step size RRT (d_{step_RRT})	$\frac{1}{21-1} = 0.05$	[-]
Distance to travel per iterate (d_{s_step})	$\frac{2}{res-1}$	[-]
Distance between current UAV position and prospective new intermediate goal point (d_{int_goal})	0.4 and 0.6 for Mixed case scenario	[-]
Maximum time to generate path segment ($t_{iterate\_max}$)	$\frac{d_{s\_step} \times 60 \times 60}{100 \times v_{UAV}}$	s
Maximum time to generate path ($t_{path\_gen\_max}$)	$10 \times t_{iterate\_max}$	s
Distance reduction factor (d_{factor})	0.8	[-]

Once the above parameters are defined, the environment is generated *apriori* for a predefined number of times based on the expected amount of iterates required which varies significantly for different path planning algorithms, environmental scenarios and random sequence that is different for different iterates but initiates using the same random seed for both A* and RRT tests. Therefore, the number of environmental generations was defined with a large margin.

Unless the intermediate and total path generation time is less than $t_{iterate\_max}$ and $t_{path\_gen\_max}$, respectively the real–time algorithm can continue searching for a path to intermediate goal otherwise no path is possible. In a nutshell, during this searching process, the algorithm loads the respective environment, assigns

a new intermediate goal point, generate a path from the current UAV position to goal if possible and moves the UAV to a new current position on the generated path.

The new intermediate goal point is determined by d_{int_goal} in the direction of the final goal provided that the selected point does not reside on an obstacle otherwise d_{int_goal} is reduced by d_{factor}. If although an intermediate goal point is available but a path cannot be constructed, the algorithm reduces d_{int_goal} by d_{factor} to increases the chances of generating a path. This process is repeated until the intermediate time exceeds the maximum allowable intermediate time. Another option that is considered is to wait at the current UAV position until the intermediate goal point is available. This waiting process is halted if no solution to the intermediate goal point results in the allowable maximum intermediate time. Both solutions will be assessed and the results discussed in Section VI.

A similar approach was considered in defining a new UAV position based on the constructed path using either the A* or RRT algorithms to the intermediate goal point. Although the intermediate path was obstacle free when constructed, in the next iteration an obstacle may have moved in the path line. Therefore, the algorithm must re–check that the path line d_{s_step} distance from the current UAV position have remained obstacle–free. Otherwise, the algorithm will need to move the UAV $d_{s\_step} \times d_{factor}$ distance on the path. In case, the path segment in consideration remained non–obstacle–free the expression $d_{s\_step} \times d_{factor}$ is further re–multiplied by d_{factor}. This process is repeated until a non–colliding path segment to new UAV position is found or the maximum intermediate time has been exceeded. Only in the case of A*, a movement less than the resolution can yield a no movement whatsoever, resulting in a waiting phase for the UAV.

In contrast with our previous implementation[3] that only considered 2D planes, this implementation considered also open and closed 3D obstacles. Closed obstacles present a new situation that makes it more difficult to determine whether the obstacle is closed and therefore points within the obstacle are unavailable or opened from part of the 3D obstacle, implying that points within are freely available. This issue was mitigated by checking that each point is not smaller than the maximum and not larger than the minimum of coordinate in each plane for every dimension. In this case, the point will reside inside the closed 3D obstacle. Besides this, for A*, a safety margin of half the distance between grid position was also considered.

VI. Results

A. Introduction

The whole algorithm described in the previous sections was implemented in MATLAB and simulations were computed using an Intel Xeon ES–1650, 3.2GHz. The path length, computational time and success rate are the performance measures considered. Each constant parameter tabulated in Table 2 was assigned as in the same table whilst the UAV speed was varied as described in Section V.

B. A* results

Figure 2 illustrates the performance results of the A* algorithm for the two cases described earlier that consider two contrasting rationale when a path to the intermediate goal point is not available. In the first case, the real–time path planning system waits in its current UAV position until a path to the intermediate goal point is available (A* waiting). In the second case, the real–time path planning system defines a new intermediate goal point a shorter distance (governed by d_{factor}) towards the assigned UAV position (A* moving).

The path lengths for A* waiting and A* moving are similar for the first three scenarios for all speeds considered although the mean of the A* waiting instances results in a longer length of 0.6%, 0.1% and 0.3% as compared to A* moving for the first three scenarios, respectively. This shows that by moving nearer to the obstacle there is the possibility of finding shorter paths for the same environment. Another interesting point is that the path length for the cube with no rotation is always less than the rotation case for the moving algorithm. Oppositely, the waiting algorithm shows situations were the two lengths are equal or even the rotating case is shorter with respect to the non–rotating case. By nearing in the vicinity of obstacles, the planner can make optimal use of the non–rotating factor creating a shorter path. By waiting for the obstacle to clear for the prospective intermediate goal position, being a rotating or a non–rotating obstacle will make lesser of a difference in terms of path length.

Overall, the the shortest path length was recorded for the non–rotating cubes followed by the rotating cube and the V–obstacle cases. Rotating objects virtually occupy a larger volume than their actual size as

opposed to non–rotating equivalent objects resulting in longer paths as their effect on the generated path can be larger once their orientation and position changes. Although the V–obstacle, being an open obstacle with 2, 2D planes occupies a lower volume with respect to the cube cases it may result in shorter paths, but this is not the case as confirmed from the results. The definition of the V–obstacles in a graph–based environment is dependent upon the resolution. For the considered resolution, inside the V–obstacle the distance between the planes is at some parts (more than 50%) lower than the distance between grid positions (a buffer of half the distance between grid positions is considered for all planes). Also, the V–obstacle is rotating with each plane larger than each side of the obstacle. These two factors combined consequently increase the path length with respect to the cube cases. Speed for both cases considered had no effect on the path length. This is attributed to the fact that irrespective of speed the planner allocates the shortest sub–path. With higher speeds the UAV will travel this sub–path faster consequently not effecting path length at all.

For the Mixed cases, the waiting algorithm resulted in 0% success rate for the majority of the speeds considered with a maximum of 2% at 4 other different speeds. So although from the successful runs when can deduce that the path length increases by approximately 1.5 times with respect to the first 3 scenarios, statistically the successful sample is low to draw conclusions. From the moving case results, with an average success rate of over 50%, the same conclusion as for the waiting case can be drawn. Owing to the difficulty and lack of free space in the mixed scenario case, the planner has to traverse a larger portion of the environmental space to reach the goal position resulting in 1.5 times path length with respect to the other scenarios. With increase in speed a drop in path length is recorded for the moving situation. This does not imply that at higher speeds the path length decreases but owing that the higher the speed the lower the success rate due to less intermediate time allocation. Therefore the best performing iterations as speed increases are considered in the path length analysis since the others lead to the maximum intermediate time violation.

The computational time for A* moving for all scenarios for all speed considered is longer by 2.31, 2.27, 6.32 and 7.19 times with respect to A* waiting for the non–rotating, rotating, V–obstacle and Mixed cases respectively. The speed has no effect on computational time for both A* waiting and A* moving for the reasons described earlier. If the mixed case scenarios are neglected due to low success rate for the A* waiting algorithm, results show that by waiting for the obstacle to clear the UAV will reach to a solution faster. So if computational demand is the bottleneck and the scenario complexity is in line with the first three scenario, it is not worth the risk of nearing in the vicinity of an obstacle as the success rate is in line with A* moving.

The maximum intermediate time allocated for the first three scenarios is absolutely more than enough to generate an intermediate path. The lowest computational time is recorded for the V–obstacle. It is only 8.0% and 21.8% of the non–rotating cube case for the A* waiting and moving respectively. The V–obstacle scenario constitute the lowest obstacle restricted case since each obstacle consist of only two planes, therefore it is easier to define the obstacle nodes in the graph space and check for collisions. The obstacle avoidance sub–routines are the most computationally demanding parts of the algorithm. Therefore, the inclusion of 3D obstacles will increase the computational burden since besides check for collisions since besides a larger number of planes from 20 (V–obstacle) to 60 (cubes) the algorithm need to model and check for collisions inside the cubes. Also from Figure 2 it can be deduced that obstacle rotation increases computational time by 5.1% and 3.3% for A* waiting and moving, respectively. This is attributed to the rationale described earlier that with rotation the unavailable space changes in orientation between one iterate and another causing more adaptions in the construction of the new intermediate path. For the mixed case in the A* moving algorithm, a larger mean and variance is recorded for the low speed situations. This is attributed to the higher success rate for low speed situation due to the higher maximum intermediate time allocation.

For A* waiting the average success rate is 100%, 99.8%, 97.4% and 0.6% while for A* moving the success rate is 99.7%, 100%, 98.7% and 66.2% both results for the four consecutive scenarios. The results show that A* moving outperforming A* waiting in the mixed case scenario with an almost equal success rate close to 100% for the other scenarios. The difference in success rate between the variants increases with the complexity of the scenario. This can be attributed to the fact that the A* moving algorithm takes more risks in approaching an obstacle with a greater chance of finding an available grid position behind the obstacle while the A* waiting case is more vigilant and waits until a path is available within its safe zone. Although the difference in success rate for the first three scenarios is minimal this does not imply that by adding rotation the chances of finding a path to goal remain the same. Results of success rate only show that the allocated maximum intermediate and total times are more than required for both. So the difference is not visible. The rotating factor success rate reduction cannot be neglected if more stringent times are applied, as is deduced in the computational time analysis.

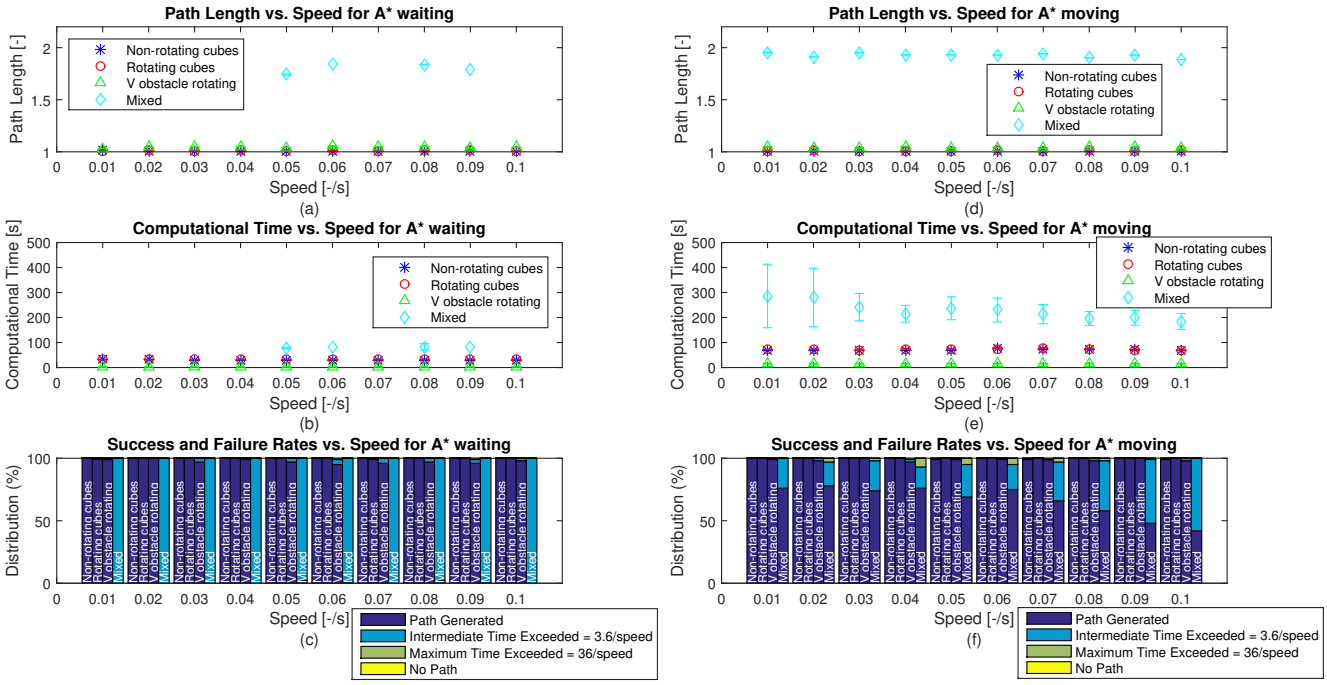

Figure 2. Performance parameters vs. speed: (a) Path Length for A* waiting, (b) Computational Time for A* waiting, (c) Success and Failure rates for A* waiting, (d) Path Length for A* moving, (e) Computational Time for A* moving and (f) Success and Failure rates for A* moving for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{int\_goal} = 0.2, d_{factor} = 0.8$ **and** $t_{iterate\_max}, t_{path\_gen\_max}$ **are a function of the UAV speed).**

Furthermore, the success rate is independent of speed for both variants for all scenarios except for the mixed case in A* moving. With lower speeds the planner is allocated more time to find a path and therefore the chance of finding one is higher. This trend is not shown in all cases except for the A* moving in the mixed case since the success rate is saturated. For the A* moving mixed case a drop in success rate is visible mainly due to the lower maximum intermediate time allocation. The main cause of unsuccessful runs in both A* waiting and moving is the lack of intermediate time. A* moving success rate results also so some total time violations for the mixed case as the path length is longest for this scenario. This violation never triggered in A* waiting. Oppositely, the No path violation is recorded for A* waiting. This implies that there are 2 instances out of 4000 where the planner waited but by waiting the obstacle moved towards her causing a collision. This rare situation is very dangerous especially in combat applications. So the use of A* waiting should be used with caution to eliminate this possibility. Otherwise it would be better to risk rather than get crashed into by an enemy or moving obstacle.

In conclusion, the path length and computational time for both A* waiting and A* moving are similar although the overall success rate is equal for the first three scenarios but better by 65.6% for A* moving with respect to A* waiting. This shows that there exist a better chance of reaching the final goal if the planner identifies a closer intermediate goal point that will allow the UAV to extend its visual line of sight at the expense of being nearer to the obstacle. As the speed of the UAV is assumed to be equal or higher than that of an obstacle in the environmental space, the UAV will always be safe to fly away from the obstacle. Nearing an obstacle can result in putting the UAV at a larger risk especially if the obstacle if an armed enemy. Speed have negligible effect on path length and computational time for the considered speed range. Finally, obstacle density and rotation have adverse affect on path length, computational time and success rate.

C. RRT results

Figure 3 illustrates the performance results for the RRT algorithm for the waiting and moving cases for the scenarios described in the A* algorithm analysis. The lowest path length was exhibited for the cube case with no rotation, followed by the rotating cube and V–obstacle scenarios with the mixed case exhibiting a multiple times larger path length due to the multiple times higher scenario complexity with respect to the other scenarios. Although the V–obstacle occupies less space than the cube, its two planes are larger than each plane of the cube. Furthermore, since a path can be constructed multiple times faster for the V–obstacle scenario than for both cube cases due to lower obstacle density, the constructed path will not be as optimal since the requirement in our path construction algorithm is to construct a non–colliding path in the minimum time possible and the amount of restrictions to tree propagation in the V–obstacle case will be less restrictive with respect to the cube cases. Although the smoothing algorithm will reduce sharp turns in the path it will not achieve the same result as if the path was smoother from the beginning.

The results in terms of mean and standard deviation for both RRT variants are equal for all speeds considered for all scenarios except for the mixed case scenarios. The relatively large path length difference for the moving configuration in the mixed case is attributed to the higher obstacle density. Results show that path length is independent of speed for all scenarios. This implies that irrespective of speed the path length is invariant although the path followed may differ depending on the random seed sequence. Rotation adds to path length as can be deduced from the non–rotating and rotating scenarios. The addition is less than 0.1% for both RRT waiting and moving configurations. This conclusions were also derived in the A* case.

The computational time for RRT waiting is 3.0%, 2.4% and 1.2% times longer with respect to RRT moving for non–rotating cubes, rotating cubes and V–obstacle rotating, respectively. For the mixed scenario, comparison cannot be made since the success rate for the waiting configuration is 0% for all tests. Result for the first three scenarios shows that in terms of computational time, the RRT moving option is better than the RRT waiting option for all cases. As for A*, the lowest computational time is recorded for the V–obstacle case, 5.0 and 4.9 times lower than the second best performing scenario i.e. non–rotating cube case for RRT waiting and moving, respectively. By introducing rotation to the cubes the computational time increased by 32.3% and 33.0% for RRT waiting and moving, respectively. The difference in computational time between the mixed case and the other scenarios is 10s of multiple times longer for the moving case.

The low obstacle density of the V–obstacle case (open obstacle) is advantageous for the RRT planner as the environment is almost free except for two planes per obstacle constituting the V–obstacle. Computationally it is easier to check for collisions in tree branches, path segments and new intermediate and UAV positions if

only obstacles are made of only 2D planes rather than closed obstacles. In closed obstacles, besides checking for collisions with obstacle surfaces, the collision avoidance sub–algorithm must check if any part of the path segment or the point lies inside the 3D obstacle. This explains the lower computational time of the V-obstacle case with respect to the non–rotating cubes. Zooming in on Figures 3 (b) and (e), shows that as the speed increases the computational time remains constant for all scenarios except for the Mixed case in the moving case configuration. Only the maximum intermediate and total times are a function of speed. The maximum intermediate and total times only determine the success rate. So independent of speed the planner is required to construct a path a constant distance from the current UAV position to an intermediate goal point and determine the new UAV position also a constant distance from the current UAV position on the constructed path segment. This explains why for the different speeds considered the computational time is invariant. For the mixed case in the moving configuration the reduction in computational time with increase in speed is a consequence of the drop in success rate due to lower maximum intermediate time allocation which is inversely proportional with speed. The computational time to generate an intermediate path is mainly dependent upon scenario difficulty as can be deduced from Figure 3. Therefore, it can be concluded that computational time is independent of speed for both RRT variants.

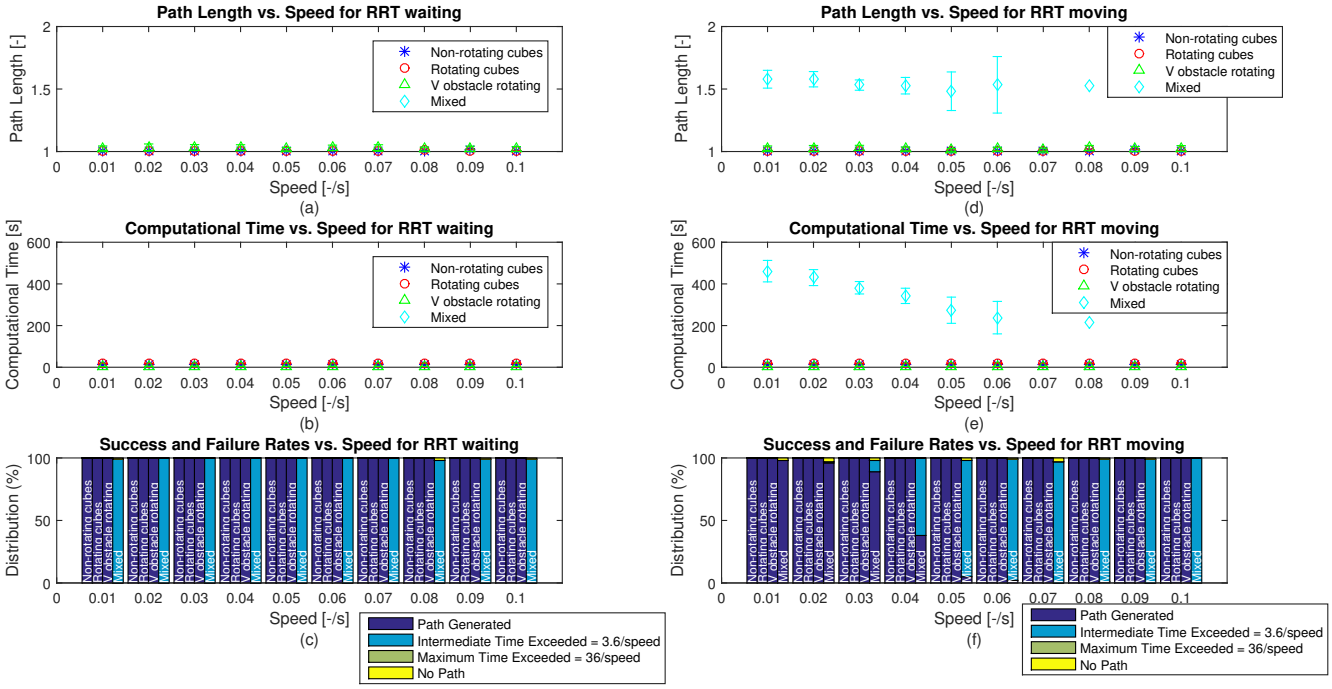

Figure 3. Performance parameters vs. speed: (a) Path Length for RRT waiting, (b) Computational Time for RRT waiting, (c) Success and Failure rates for RRT waiting, (d) Path Length for RRT moving, (e) Computational Time for RRT moving and (f) Success and Failure rates for RRT moving for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($d_{s\_step} = 0.05, d_{int\_goal} = 0.1, d_{factor} = 0.8$ **and** $t_{iterate\_max}, t_{path\_gen\_max}$ **are a function of the UAV speed).**

Figures 3 (c) and (f) illustrates the success rate for RRT waiting and moving, respectively. The figure show a 100% success rate for both configurations for the cube with and without rotation and the V–obstacle scenarios. The result show the robustness of both RRT variants and the look–ahead distance algorithm that is able to mitigate with moving obstacles. For the mixed case, the RRT waiting case showed no successful runs while for the moving case a success rate of 98% was achieved for low speed reducing to 0 as the speed increases. This is attributed to the fact that in the waiting case the planner waits until the intermediate goal point a predetermined distance from the current UAV position is free from obstacles and a path to this intermediate goal point is possible. The maximum intermediate time which is equal to the waiting time is not enough for the obstacle to move away from the intermediate goal position. For the moving case, the planner reduces this distance by the distance reduction factor until an intermediate path to an intermediate goal point is possible. This effect is only visible for the mixed case since in this situation the environment is

crowded with obstacles and so the above mentioned condition that an intermediate path is not possible is more frequent.

As described earlier, speed defines the maximum allowable intermediate and total time for the planner to generate a path. For the first three scenarios, time was never the bottleneck and the planner was able to construct the path in a fraction of the maximum intermediate and total time. The planner in all situations will require a range of times to generate intermediate paths as obstacles' initial and time–variant positions change during path construction from one run to another although the start and goal points, environmental space and amount, size and shape of obstacles is constant for a particular scenario. Therefore as speed increases and therefore the maximum allowable intermediate and maximum time reduces a higher percentage of runs for the Mixed case will exceed the maximum allowable intermediate time at one instance, resulting in a maximum intermediate time violation. In fact, the majority of unsuccessful runs result due to insufficient intermediate time. This explains the drop in success rate with increase in speed.

Another aspect is that the RRT waiting case resulted in less No Path situations than RRT moving. This is attributed to the fact that in the moving case the UAV is nearer to moving obstacles than in the waiting case that waits until a path a predetermined distance from the current UAV position is possible. This increases the risk of collision as the obstacles can move into the new or current UAV position for the moving case with respect to the waiting case.

In conclusion results of both A* and RRT show that exploring further into the vicinity of an obstacle in the direction of the goal increases the chances of finding a viable path around the obstacle $vis - a - vis$ waiting at the current position until an intermediate goal point is available and a path to it the intermediate point is possible. The moving tactic increases the risk of collision due to a number of factors including sensor inaccuracy and unknown future obstacle state. On the other hand, waiting allow the UAV to stay at a safer position from the obstacle in its vicinity, but the UAV cannot stop forever and staying stationary will make the UAV an easy prey. Therefore, for the scope of comparison between A* and RRT both moving cases will be considered as for both A* and RRT the moving variant resulted in better success rates.

D. A* vs. RRT

Figure 4 illustrates the path length, computational time and success rate for the A* and RRT algorithms for the moving variant. In terms of path length the RRT algorithm constructed shorter paths with respect to the A* algorithm, although the mean difference is less than 1% for the cube cases increasing to 3.8% for the V-obstacle cases. For the mixed case, the success rate at speed higher than 0.06[-]/s is 1% or less if we consider the path length for speeds from 0.01[-]/s to 0.05[-]/s the RRT algorithm constructed paths 23.3% shorter with respect to A*. In terms of variance, RRT also exhibits lower variance for all scenarios for all speeds with respect to A*, implying that the constructed path is less effected by the movement of obstacles. The shorter path length and larger variance range is mainly attributed to the graph–based nature of the A* algorithm. In A*, obstacles are modelled as unavailable grid positions with a buffer equal to half the distance between grid positions as otherwise during re–assignment a node which resides a distance lower than the buffer from an obstacle may be unavailable in the next iterate even if the environment remains static both due to quantisation and also due to the ripple reduction algorithm.[2] The smoothing algorithm which was only applied for the RRT algorithm cannot be applied to the A* algorithm since this will near the path to a distance less than the buffer distance to the obstacles possibly resulting in collisions in the next iteration. Refer to[1–3] for further details. Furthermore, for both algorithms the path length is independent of speed as speed only effects the maximum allocated time to construct the path.

The computational time characteristics illustrated in Figures 4 (b) and (e) for A* and RRT, respectively show that a longer mean computational time with a higher variance is recorded for A* with respect to RRT. For both algorithms, the V–obstacle results in the shortest path followed by the cube without and with rotation and the mixed case. Also for both algorithms speed has no effect on performance. The mean computational time for A* is 6.6, 5.1 and 7.0 times longer with respect to RRT for the first three scenarios respectively and 17.9% shorter for A* with respect to RRT for speeds from 0.01[-]/s to 0.05[-]/s as success rate is very low for RRT at larger speeds. The variance is similar for the first three scenarios but is larger for A* in the Mixed case with respect to RRT for small speeds. This increase in variance is the result of the quantisation of the graph–based nature of the A* algorithm.

From the computational time results it can be concluded that the RRT algorithm performed better than the A* for the first three scenarios deteriorating at a faster rate than A* as the obstacle complexity increases. In our previous analysis[1,2] it was showed that the A* algorithm was faster than RRT in finding a goal offline

in the presence of 2D obstacle. With the inclusion of 3D obstacles the computational demand to check whether each node is within or residing at a distance lesser than half the distance between grid positions has increased significantly. For RRT the difference in computational demand is less as the planner is not required to model the environment within the sensory system field of view but only check for collision once tree nodes and branches are to be constructed. Since the obstacle density in the first three scenarios is low this explains why the RRT performed better than A*. For the highly–obstacle Mixed case the RRT the chance of collisions with obstacle is much larger and therefore more time is required to find a sub–path. For A* the difference is minimal as in all cases the planner need to check each node in the environment.

Figure 4 (c) and (f) show that the success rate is almost 100% for A* and 100% for RRT for the first three scenarios for all speeds considered for both A* and RRT algorithms. From the results it can be concluded that RRT will be able to generate a path in low complexity environments while A* may not be able to offer this guarantee. The computational response shown in Figure 4 (b) and (e) show that the allocated time is more than enough for the majority of the situations and the non successful runs are a consequence that no path could be constructed with the considered FOV and resolution in A* in few situations for the first three scenarios irrespective of the allowable time.

For the Mixed case a drop in success rate is shown as speed increases with the major drop exhibited for RRT. The Mixed case is at least three times more complex than the other three scenarios. As scenario complexity increases the computational requirements for RRT increases at a higher rate for RRT with respect to A* resulting in maximum intermediate time violations. This is attributed to the fact that A* must model all obstacles within its FOV irrespective of the complexity of the environment while the RRT only checks for collisions as described earlier. Therefore it can be concluded that the RRT is more vulnerable to increase in obstacle complexity with respect to RRT. The RRT algorithm was able to achieve a success rate of 95% or better for the lowest three speeds considered. This high success rate was never recorded for A*. From this it can be concluded that A* will not achieve the same level of success rate as RRT if the latter is given appropriate time to construct sub–paths. More computational time can be achieved by reducing the speed and/or increasing the computational processing power. Another point is that the RRT issued No Path violations for certain mixed case scenarios while A* never issued this violation. This implies that a violation occurred since the UAV cannot move as no path is possible to an intermediate goal point or that an obstacle collided into the UAV. This situation is very dangerous as it could lead to the loss of the vehicle. For A* this situation never resulted mainly due to the buffer distance considered for quantisation purposes and the UAV although not reaching the path never collided or is at risk of colliding.

E. Conclusion

Throughout this section, the performance results of the A* and RRT algorithms using both waiting and moving variants were illustrated, analysed and compared to select the best option in view of 3D UAV path planning with moving obstacles. Results showed that the A* moving variant produced better results with respect to A* waiting variant while similar results were recorded for RRT for both variants. Overall, RRT results in better or equal success rate for all scenarios at all speeds with respect to A*, except for high speeds in the Mixed case scenario. This shows that RRT is better suited for 3D real–time applications in dynamic obstacle–rich environments provided that appropriate time is allocated for obstacle–rich environments. This does not exclude the A* algorithm from dynamic environments especially if safety is a paramount requirement and the computational power in complex environments is limited. In such case the A* would lead to better results.

VII. Conclusion and future work

This paper implemented 3D real–time path planning A* and RRT algorithms in the presence of dynamic constraints, obstacles and threats. Literature confirms that dynamic path planning is essential as UAVs are sometimes required to operate in time-variant environments. In this regard, 4 different scenarios were considered with increasing difficulty consisting of V–obstacles, enclosed cubes and 2D planes moving at different time–varying speeds, direction and orientation. Only, obstacle characteristics a look–ahead distance away from the current UAV position are known with certainty by the planner with updates being provided as the UAV moves. This was done to emulate sensory systems onboard UAVs. The planner was time–limited both in the construction of individual path segments and in the overall path as the UAV is not allowed

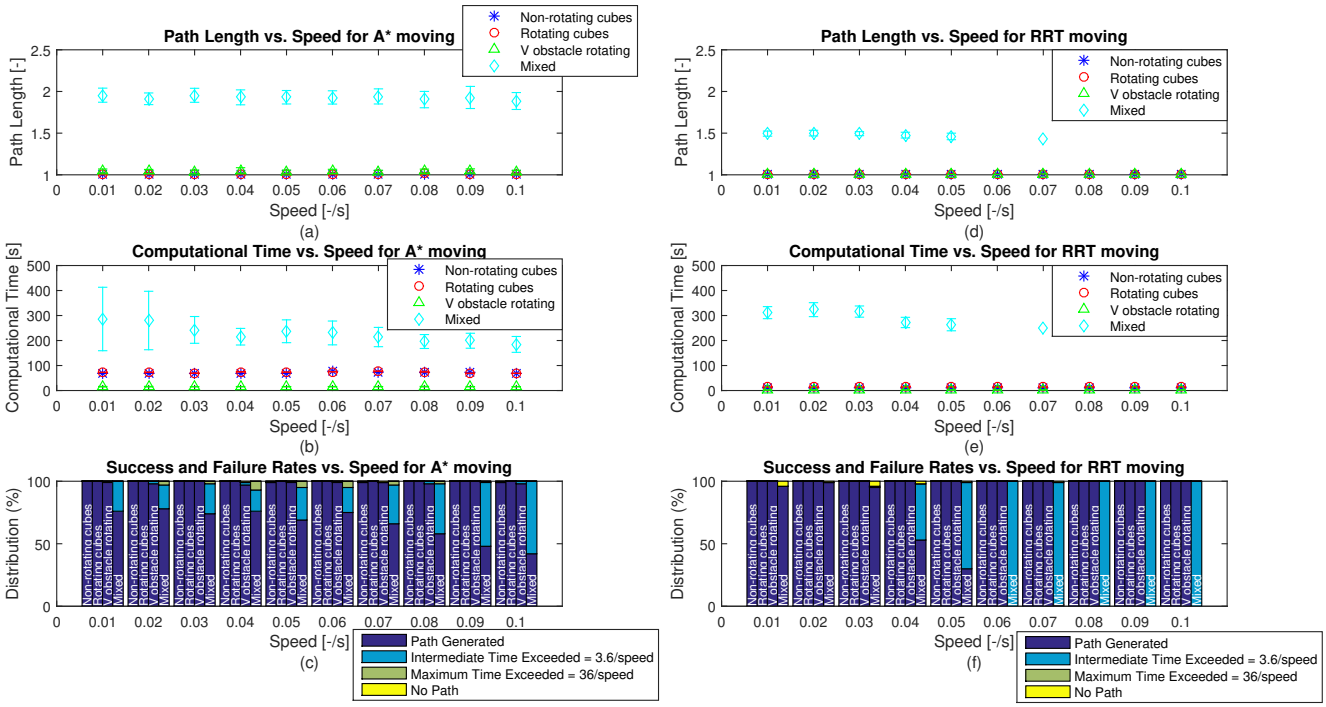

Figure 4. Performance parameters vs. speed: (a) Path Length for A*, (b) Computational Time for A*, (c) Success and Failure rates for A*, (d) Path Length for RRT, (e) Computational Time for RRT and (f) Success and Failure rates for RRT for 100 iterates for each considered situation (speed and scenario) with 95% confidence interval. ($res = 21, d_{s\_step} = 0.05, d_{int\_goal} = 0.4$ (for Scenarios 1,2 and 3) and $d_{int\_goal} = 0.6$ (for Scenario 4), $d_{factor} = 0.8$ and $t_{iterate\_max}, t_{path\_gen\_max}$ are a function of the UAV speed).

to stop and is required to always have available the next position prior moving from the current position. Two rationale were considered and implemented for both A* and RRT planners in case an intermediate goal point was not available, namely waiting until the defined intermediate goal point becomes available or moving to the intermediate goal point nearer to the current UAV position consequently increasing the chance of moving further towards the final goal. Path length, computational time and success rate were considered as the performance measures. These measures were assessed with UAV speed which was varied between 0.01 and 0.1[-/s].

Results show that the moving option yielded better overall results in terms of path length, computational time and success rate for A* and RRT with respect to the waiting option especially for the Mixed case which recorded an almost 0% success rate for the waiting case for both variants. UAV speed determines the maximum intermediate and total time, reducing as speed increases. Results show that as speed increases success rate drop due to lack of path planning time for the Mixed case scenarios in both A* and RRT as in these situations the required path planning time is near the maximum while for the other scenarios the allocated time is much more than required. Overall, both A* and RRT produced similar results for the first three scenarios with RRT recording slightly better results in path length, computational time and success rate. For the Mixed case, the RRT algorithm performed better at low speeds but worst than A* for high speeds. Also, A* was only limited but time while the RRT resulted in No Path situations that can potentially lead to collisions. The results show that both algorithms can be applied in low obstacle density environments in real–time in the presence of moving obstacles. For complex scenarios the RRT is better suited if time is not limited while the A* algorithm is less susceptible to time constraints. Finally, this work shows that 3D real–time path planning in low and high obstacle density, moving obstacle environments is possible.

Future enhancement shall include the analysis of the effects of other parameters on the performance of both path planners with both waiting and moving rationale. These parameters shall include, resolution and step–size in case of A* and RRT respectively, look–ahead distance, distance to travel per iterate, intermediate and total time and the distance factor which defines the reduction factor of look–ahead distance and distance to travel per iterate in case the intermediate goal point or the new UAV position reside on an obstacle. This analysis will flag the best configuration for both algorithms in the considered test scenarios which offer a range of path construction difficulties. Furthermore, real sensory systems sense the environment within a certain degree of uncertainty that changes with a number of factors including: distance to the object, shape and light levels. This uncertainty may have an effect on the performance of the 3D real–time path planners as paths are generated only with a certain degree of non–collision. This can generate unexpected collisions with obstacles that can ultimately result in the loss of the UAV.

References

[1]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.

[2]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.

[3]Zammit, C. and van Kampen, E. J., "Comparison of A* and RRT in real–time 3D path planning of UAVs", *AIAA Guidance, Navigation and Control Conference*, AIAA SciTech Forum, Kissimmee, FL, 6–10 Jan., 2020.

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

[5]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.

[6]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.

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

[8]Goerzen, C., Kong, Z. and Mettler, B. "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance", *Journal of Intelligent & Robotic Systems*, Vol. 57, pp. 65-100, 2010.

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

[10]Maza, I., and Ollero, A., "Multiple UAV cooperative searciteching operation using polygon area decomposition and efficient coverage algorithms", *Jistributed autonomous robotic systems*, in R. Alami, R. Chatila, and H. Asama (Eds.), Vol. 6, Tokyo, Japan: Springer, pp. 221-230, 2012.

[11]Jiao, Y.-S.,Wang, X.-M., Chen, H. and Li, Y., "Multiple Research on the coverage path planning of UAVs for polygon

areas", *5th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Taichung, Taiwan, 15–17 Jun., 2010, pp. 1467-1472.

[12]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.

[13]Adolf, F.–M., Andert, F. and Rocha, J. G. F., "Rapid Online Path Planning Onboard A VTOL UAV", *AIAA Infotech, Aerospace Conference*, Atlanta, GA, Apr. 20–22, 2010.

[14]Srikanthakumar, S., Liu, C. and Chen W. H. "Optimization-Based Safety Analysis of Obstacle Avoidance Systems for Unmanned Aerial Vehicles", *Journal of Intelligent Robot Systems*, Berlin, Germany: Springer Science & Business Media, 2012.

[15]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.

[16]Lee, J.-W., Walker, B. and Cohen, K. "Path Planning of Unmanned Aerial Vehicles in a Dynamic Environment", *Infotech@Aerospace*, St. Louis, Missuori, 29–31 Mar., pp. 1–19.

[17]Swartzentruber, L., Foo, J. L. and Winer, E., H. "Three-dimensional multi-objective uav path planner using terrain information", *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Palm Springs, California, 4–7 May 2009, pp. 1–19.

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

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

[20]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.

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

[22]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.

[23]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.

[24]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.

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

[26]Fujimura, K. *Motion planning in dynamic environments*, Berlin, Germany: Springer Science & Business Media, 2012.

[27]Mac, T. T., Copot, C., Tran, D. T. and Keyser, R. D. "Heuristic approaches in robot path planning: A survey", *Journal of Robotics and Autonomous Systems*, Vol. 86, Aug. 2016, pp. 13–28.

[28]Qin, Y. Q., Sun, D. B., Li, N. and Cen, Y. G. "Path Planning for mobile robot using the particle swarm optimization with mutation operator", *International conference on Machine learning and Cybernetics*, Shangai, China, 26–29 Aug. 2004, pp. 2473–2478.

[29]Mac, S. P. "Optimal path palnning for mobile robots: A review", *International Journal of Physical Science*, Vol. 7, No. 9, Aug. 2012, pp. 1314–1320.

[30]"A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments, *IEEE Transactions on Neural Networks*, Vol. 19, No. 7, pp. 1279–1298, 2008.

[31]Zhang, H., Butzke, M. and Likhachev, M. "Combining global and local planning with guarantees on completeness", *IEEE International conference on Robotics and Automation*, St. Paul, MN, 14–18 May 2012, pp. 2500–2506.

[32]Wang, L. C., Yong, L. S. and Ang J. M. H. "Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment", *IEEE International Symposium on Intelligent Control*, Vancouver, British Columbia, Canada, 27–30 Oct. 2012, pp. 821–826.

[33]Bi, Z. and Yimin, Y. "Hierarchical path planning approach for mobile robot navigation under the dynamic environment", *IEEE International Symposium on Industrial Informatics*, Daejeon, Korea, 13–16 Jul. 2008, pp. 372–376.

[34]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.

[35]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.

[36]Chawla, C. and Pahdi, R., "Neuro–Adaptive Augmented Dynamic Inversion Based PIGC Design for Reactive Obstacle Avoidance of UAVs", *AIAA Guidance, Navigation, and Control Conference*, Portland, OR, 08–11 Aug., 2011, pp. 1–25.

[37]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.

[38]Svec, P., Shah, B.C., Bertaska, I.R., Alvarez, J., Sinisterra, A.J., Von Ellenrieder, K., Dhanak, M., and Gupta, S. K. "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic", *IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 3–7 Nov., 2013, pp.3871–3878.

[39]Hsu, D., Kindel, R., Latombe, J.–C., and Rock, S. "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, Vol. 21, No. 3, pp. 233–255, 2002.

[40]Howlett, Schulein, G. and Mansour, M. H., "A Practical Approach to Obstacle Field Route Planning for Unmanned Rotorcraft", *60th Annual Forum at the American Helicopter Society*, Baltimore, MD, Jun. 7–10, 2004.

[41]Oomkens, W. "UAV Aviating Automation", Master Thesis, TU Delft, 2007.

[42]Larson, J., Bruch, M., and Ebken, J., "Autonomous navigation and obstacle avoidance for unmanned surface vehicles, *Proceedings of SPIE*, Vol. 8, pp. 17–20, 2006.

[43]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.

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

[45]Liu, Y., Cruz, J. B. and Sparks, A. G., "Coordinating Networked Uninhabited Air Vehicles for Persistent Area Denial", *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec., 2004, pp. 3351–3356.

[46]Koenig, S. and Smirnov, Y. "Sensor–based planning with the freespace assumption", *International Conference on Robotics and Automation (ICRA)*, Albuquerque, New Mexico, 20–25 Apr., 1997, pp. 3540-3545.

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

[48]Guivant, J. and Nebot, E."Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementation", *IEEE Transactions of Robotics and Automation*, Vol. 17, No. 3, pp. 242-257, 2001.

[49]Thrun, S., Diel, M. and Hahnel, D. "Scan Alignment and 3-D Surface Modeling with a Helicopter Platform", *4th International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 14–16 Jul., 2003, pp.1–6.

[50]Ericson, C. "Real-Time Collision Detection", *Real–Time Collision Detection*, 1st ed., Vol. 1, Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[51]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.

[52]Scherer, S., Singh, S., Chamberlain, L., and Saripalli, S."Flying fast and low among obstacles", *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 11–13 Apr., 2007, pp.2023–2029.

[53]Shim, D., Chung, H., Kim, H. J., and Sastry, S. "Autonomous exploration in unknown urban environments for unmanned aerial vehicle", *AIAA Ground, Navigation and Control Conference*, San Francisco, CA, 15–18 Aug., 2005, pp.1–8.

[54]Gottschalk S., Lin M. C. and Manocha D., "OBB–Tree: A Hierarchical Structure for Rapid Interference Detection," *Proceedings of 23rd ACM Siggraph Structure for Rapid Interference Detection*, ACM, NY, pp. 171–180.

[55]Ögren, P. and Robinson, J. W. C. "Receding Horizon Control of UAVs using Gradual Dense–Sparse Discretizations," *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario, Canada, 2–5 Aug. 2010, pp. 1–11.

[56]Ok, K.-C., Ansari, S., Gallagher, B., Sica, W., Dellaert, F., and Stilman, M."Path planning with uncertainty: Voronoi uncertainty fields", *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 6–10 May, 2011, pp.4596–4601.

[57]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.

[58]Lewis, L.R. " Rapid Motion Planning and Autonomous Obstacle Avoidance for Unmanned Vehicles", Master Thesis, Naval Postgraduate School, Monterey, CA, 2006.

[59]Subramanian, S. K. and Cruz, J. B., "Adaptive Models of Pop-Up Threats for Multi-Agent Persistent Area Denial," *42nd IEEE Conference on Decision and Control*, Maui, Hawaii, 9–12 Dec. 2003, pp. 510–515.

[60]Kim, J., Kim, M. and Kim, D. "Variants of the quantized visibility graph for efficient path planning", *Journal of Advanced Robotics* , Vol. 25, No. 18, pp. 2341-2360, 2011.

[61]Tsardoulias, E. G.,Iliakopoulou, A., Kargakos, A. and Petrou, L. "A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density", *Journal of Intelligent & Robotic Systems*, Vol. 84, No. 1–4 pp. 829-858, 2016.

[62]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.

[63]Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. and Thrun, S., "Principles of Robot Motion: Theory, Algorithms, and Implementations", Massachusetts: MIT Press, 2005.

[64]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.

[65]Latombe, J.–C., "Robot Motion Planning", *The Springer International Series in Engineering and Computer Science*, Berlin, Germany: Springer Science & Business Media, 1991.

[66]Roos, T. and Noltemeier, H. "Dynamic Voronoi Diagrams in Motion Planning *Lecture Notes in Computer Science*, Vol. 553, No. 17, Berlin, Germany: Springer Science & Business Media, 1991.

[67]Yang, L., Qi, J., Song, D., Xiao, J., Han, J. and Xia, Y. "Survey of Robot 3D Path Planning Algorithms *Journal of Control Science and Engineering*, Vol. 2016, No. 5, pp. 1–22, 2016.

[68]Harabor D. and Botea, A. "Breaking Path Symmetries on 4-Connected Grid Maps, *Proceedings of the Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*, Stanford, CA, 11–13 Oct., 2010.

[69]Lau, B., Sprunk, C. and Burgard, W. "Efficient grid–based spatial representations for robot navigation in dynamic environments *Robotic Automation Systems*, Vol. 61, pp. 1116–1130, 2013.

[70]Park, B., Choi, J. and Chung, W. K. "An efficient mobile robot path planning using hierarchical roadmap representation in indoor environment *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, 14–18 May, 2012, pp. 180–186.

[71]Aoude, G. S., Luders, B.D., Levine, D. S. and How, J.P. "Decentralized path planning for multi–agent teams in complex environment using rapidly–exploring random trees *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 9–13 May, 2011, pp. 4956–4961.

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

[73]Szczerba, R.J., Galkowski, P., Glicktein, I. and Ternullo, N., "Robust algorithm for real-time route planning", *IEEE Transactions on Aerospace Electronic Systems*, Vol. 36, 2000, pp. 869–878.

[74]Zheng, C., Xu, F., Hu, X., Sun, F. and Yan, P., "Online route planner for unmanned air vehicle navigation in unknown battlefield environment," *IMACS Multiconference on Computational Engineering in Systems Applications*, Beijing, China, 4–6 Oct. 2006, pp. 814–818.

[75]Dong, Z., Chen, Z., Zhou, R. and Zhang, R. "A Hybrid Approach of Virtual Force and A* Search Algorithm for UAV Path Re-Planning, *Proceedings of the 6th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Beijing, 21–23 Jun., 2011, pp. 1140-1145.

[76]Elbanhawi, M. and Simic, M. "Sampling-Based Robot Motion Planning: A Review *IEEE Access*, Vol. 2, pp. 56–77, 2014.

[77]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.

[78]Otte, M. and Frazzoli, E. "RRTX: Real-time motion planning/replanning for environments with unpredictable obstacles, *11th Algorithmic Foundations of Robotics*, Berlin, Germany: Springer Science & Business Media, 2015, pp. 461478.

[79]Fiorini, P. and Shiller, Z., "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, Vol. 7, No. 7, 1998, pp. 760–772.

[80]Zhuang, J.Y., Su, Y.M., Liao, Y.L. and Lei, S. "Motion planning of USV based on Marine rules, *Procedia Engineering*, Vol. 15, pp. 269–276, 2011.

[81]Naeem, W., Irwin, G.W., and Yang, A., "COLREGs-based collision avoidance strategies for unmanned surface vehicles, *Mechatronics*, Vol. 22, pp. 669–678, 2012.

[82]Canny, J. and Reif, J., "New lower bound techniques for robot motion planning problems," *Proceedings of the Symposium on the Foundations of Computer Science*, Los Angeles, CA, 12–14 Oct. 1987, pp. 49–60.

[83]Canny, J., Donald, B. and Reif, J., "On the complexity of kinodynamic planning," *29th Annual IEEE Symposium on Foundations of Computer Science*, White Plains, NY, 24–26 Oct. 1988, pp. 306–316.

[84]Benjamin, M. R., and Curcio, J. A. "COLREGS-based navigation of autonomous marine vehicles, *IEEE/OES Autonomous Underwater Vehicles Conference*, Sebasco, ME, 17–18 Jun., 2004, pp. 32–39.

[85]Gonzàlez, D., Pèrez, 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.

[86]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.

[87]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.

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

[89]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.

[90]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.

[91]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.

[92]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.

[93]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.