Acceleration of Heuristic Motion Planning for Unmanned Aerial Vehicles

Neelanga Thelasingha
thelan@rpi.edu
PhD student
Rensselaer Polytechnic Institute (RPI)
Troy, NY, USA

Kaushik Nallan
nallak@rpi.edu
PhD student
RPI
Troy, NY, USA

A. Agung Julius
agung@ecse.rpi.edu
Professor
RPI
Troy, NY, USA

Sandipan Mishra
mishrs2@rpi.edu
Associate Professor
RPI
Troy, NY, USA

ABSTRACT

In this paper, we propose a heuristic-based fast motion planning framework which can be readily incorporated by the on-board path planner of Unmanned Aerial Vehicles (UAVs) to generate safe and efficient trajectories while traversing through challenging environments cluttered with obstacles. The proposed planning technique is effective for the scenarios where the exact obstacle locations need to be detected during flight and the obstacle detection range is limited by degraded environmental conditions like fog. Unlike many kinematic based planning strategies, the generated planned trajectories can be tracked effectively as they preserve the dynamics of the UAV. The planning problem is graphically represented by discretizing input and state spaces to facilitate usage of discrete search algorithms. We also propose a heuristic calculation strategy based on dynamics relaxation to accurately encode the obstacle. The Bellman optimality condition is used to modify the heuristic to facilitate faster search. This faster planning contributes to requiring a reduced minimum obstacle detection range for receding horizon planning. The proposed algorithm has been compared against an off-the-shelf nonlinear program solver and the proposed method produced superior planning times and feasible trajectories avoiding collisions. Further, we analyzed the sub-optimality of the planned trajectories and the minimum obstacle detection range required for the receding horizon planning framework.

NOTATION

$\mathbf{F}$ - continuous chains of integrators for the differentially flat dynamics
f_s - differentially flat dynamics
g - gravity
J - cost function
k - rotational inertia in ψ
m - mass of the UAV
$\mathscr{O}_i$ - state of the obstacle of interest 'i'
$\mathbf{p}$ - differentially flat state of the UAV (excluding yaw)
$\mathbf{p}_0$ - initial state of the UAV
$\mathbb{P}_D$ - set of allowable final states of the UAV
$\mathbf{q}$ - states of UAV in the differentially flat dynamics
$\mathbb{R}_i$ - 'Region of Influence' around Obstacle $\mathscr{O}_i$
T_p - total available propeller thrust
t_f - final time of single obstacle planner
$\mathbb{U}$ - feasible input set of the differentially flat model
$\mathbf{u}$ - inputs to the differentially flat model
u_ψ - yaw moment
$\mathbf{v}$ - inputs to the differentially flat model excluding yaw moment
V_{cruise} - Cruise velocity of UAV
$\mathbb{W}$ - feasible set of lattice inputs
$\mathbf{w}$ - discrete lattice input
x, y, z - planned position of the UAV in the inertial frame
$\phi/\theta/\psi$ - roll/pitch/yaw angles
G - directed tree structure
$\mathbb{S}$ - set of vertices
$\mathbb{E}$ - set of Edges
T - length of a timestep s_0 - root vertex
s - a vertex of G
$\mathbb{S}_g$ - set of goal vertices
s_g - goal vertex
$\mu(s)$ - mapping that returns underlying position co-ordinates of a given vertex
$d(s,s')$ - Step cost from vertex s to s'
$g(s)$ - Path cost from s_0 to s
$h(s)$ - Heuristic - Estimated optimal cost-to-go to the goal from a vertex s
$f(s)$ is the estimated optimal total cost of a path to any goal vertex through the vertex s
$[v_x,\ v_y,\ v_z]$ - velocities in the reduced order system
$\bar{\mathbf{p}}$ - reduced order state space
$\bar{G}$ - reduced order graph
$\bar{\mathbb{S}}$ - set of vertices in reduced order graph
$\bar{\mathbb{E}}$ - set of edges in reduced order graph
$\bar{s}$ - vertex in reduced order graph
$h_1(\bar{s})$ - heuristic for reduced order graph
$g_1^*(\bar{s}_g)$ - optimal path cost to goal in reduced order graph
$f(s)_{correct}$ - corrected total cost of the optimal path from root to goal through s
$child(s)$ - set of vertices with underlying state that is achieved by applying any lattice input to the underlying state of s
$h(s)_{correct}$ - corrected heuristic of s
$h^*(s)$ - optimal cost to go
G' - explored sub-tree of G
t_{plan} - computation time of planning algorithm

Presented at the Vertical Flight Society's 78th Annual Forum & Technology Display, Ft. Worth, Texas, USA, May 10–12, 2022. Copyright © 2022 by the Vertical Flight Society. All rights reserved.

$[x_0, y_0, z_0]$ - initial position co-ordinates
$D_{od}^{contract}$ - minimum obstacle detection range to satisfy safety contract
ΔD_{od} - increment in obstacle detection range due to planning time

INTRODUCTION

In order to ensure safe and efficient performance of Unmanned Aerial Vehicles (UAVs) while they execute complex mission tasks, it is important that their navigation framework is fully autonomous with minimal human intervention. Autonomous navigation becomes especially important when the mission involves traversing through an unknown environment cluttered with obstacles, whose sizes and positions are not known before-hand. Here, the UAV is expected to detect the obstacles and plan safe and efficient trajectories on-the-fly. Since the obstacle detection range can be limited by the environmental conditions like fog, one particular approach as discussed in (Ref. 1), involves avoiding obstacles sequentially, that is, one at a time. However, using conventional optimization-based planners are ineffective for on-board path planning due to their high solution time, as discussed in (Ref. 2) and (Ref. 3), which would restrict the UAV's performance by demanding a longer obstacle detection range and/or restricting the maximum cruise velocity.

The main objective of this work is to develop a fast motion planning method which can be readily incorporated and implemented in an online path-planning setting discussed in (Ref. 1). Several approaches in literature including (Ref. 4) and (Ref. 5) describe how graphical representation of the planning problems can be exploited to efficiently generate motion planning solutions for aerial vehicles. The authors in (Ref. 6) investigate flight path planning under weather uncertainty on discrete airways graphs and model this problem as a shortest path problem. Extending this approach, we formulate a graphical representation of the planning problem which allows the use of fast-discrete planning algorithms for the trajectory generation task while ensuring that the solution is feasible.

Since it is important to discretize the input space to achieve faster planning as discussed in (Ref. 7), the original feasible input set is converted into its equivalent lattice input set which is represented as edges of a tree structure as in (Ref. 8). Further, the corresponding UAV system states are represented as vertices in the aforementioned directed tree. Now, off-the-shelf path planners can be directly deployed onto this framework for optimal path search towards the goal. A^* algorithm is one of the best-first search methods which uses a heuristic function to guide the search (Ref. 9). The work in (Ref. 10) adapts a well-known reachability analysis techniques based on a delete relaxation of the problem. (Ref. 11) presents a different approach that computes inadmissible heuristics by learning expansion Delay for transitions in the state space. Since incorporating common heuristics like Euclidean distance (Ref. 12) in the state space would lead to higher planning time with unnecessary vertex visits for the 2^{nd} order system in focus, we propose an efficient heuristic calculation technique through model reduction by relaxing the dynamics. Further, we store the known heuristic values in a data structure which enables trading off memory to achieve even shorter planning times. Although the designed heuristic estimation technique was fast enough to achieve sub-second planning times, we observed that corrections could be made to the heuristic function to drive it towards the optimal cost. To this end, we present a heuristic learning technique that adjusts the heuristic function with each new vertex visit towards the goal. We also include the proof of optimality, admissibility and consistency properties for the proposed correction of the heuristic. Moreover, We formulate a receding horizon planning framework to handle multiple obstacle avoidance scenarios. The faster planning time contributes to having a shorter required obstacle detection range for receding horizon planning. Case studies were conducted by implementing the proposed algorithm to compare against an off-the-shelf nonlinear program solver. The results validate the developed method by producing sub-second planning times and feasible trajectories avoiding collisions. The proposed technique was able to reduce the planning time by a factor of 15 when compared to solving the optimization problem with an off-the-shelf solver. Further, We have included an analysis on the optimality of the planned trajectories and the minimum required obstacle detection range for the receding horizon planning framework.

PROBLEM DESCRIPTION

In this work, we consider the scenario as shown Fig. 1, where a UAV cruising towards its target (direction) must pass through an environment cluttered with obstacles, whose exact positions are sizes are not known beforehand. For the UAV to be able to plan and execute safe trajectories inside the obstacle field, the parameters of the UAV (maximum cruise velocity, control input bounds and obstacle detection range) and guarantees environment (maximum obstacle size and minimum separation) should the satisfy single-obstacle safety contract as well as the multiple-obstacle safety contract as discussed in (Ref. 13) and (Ref. 1). The contracts construction involves developing Regions of Influence (ROI) denoted by R_i around each obstacle $\mathscr{O}_i$ as shown in Fig. 2a such that the UAV can avoid O_i while remaining inside R_i before cruising again. If no two *ROI*s in the obstacle field intersect, the UAV can sequentially plan and avoid obstacles safely. This way , multiple obstacle avoidance scenario in an unknown environment can be handled by planning on-board for each (single) obstacle, as long as the obstacle is detected before the UAV enters it's Region of Influence.

Single Obstacle Planner

For the planned trajectories to be executed with minimum control tracking error, the mathematical model based trajectory planners consider the dynamics (and kinematics) of the UAV as well as the environment as hard constraints and optimize a desired cost function such as duration of flight, path

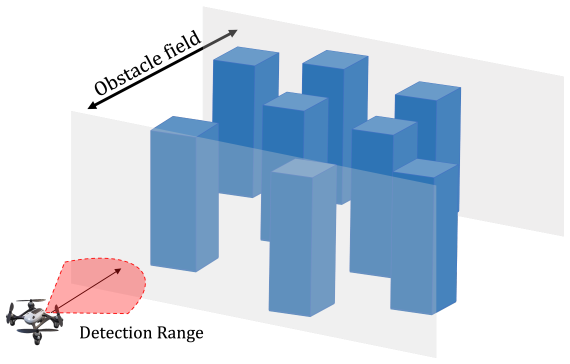
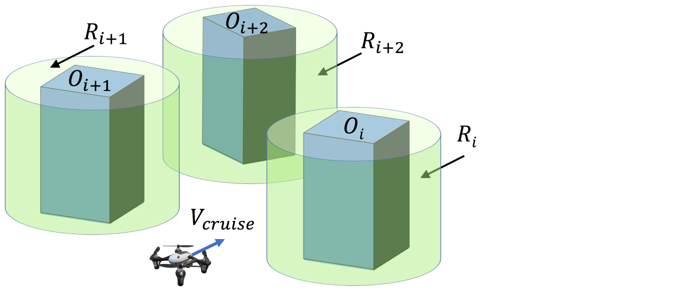
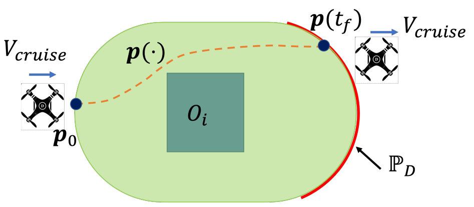


Figure 1: Schematic of UAV online path planning scenario in the presence of multiple obstacle



(a) Schematic of the sequential planning approach inside the obstacle field



(b) Top view schematic of single-obstacle avoidance

Figure 2: Schematic of sequentially avoiding multiple obstacles

length or fuel usage (Ref. 2). For sequential avoidance, this constrained-optimization problem (1) is solved for a single obstacle O_i. Here, we denote the state and control input at time t as $\mathbf{p}(t)$ and $\mathbf{u}(t)$, respectively. The objective (cost) function J in this work is chosen to be the total planned path length. The constraints of the optimization problem are due to the differentially flat dynamics f_s, initial and feasible terminal states $(\mathbf{p}_0, \mathbb{P}_D)$, obstacle O_i and the Region of Influence R_i around the obstacle. Here, $\mathbf{p}_0$ is the state of the UAV when it enters R_i with cruise velocity (V_{cruise}) and $\mathbf{P}_D$ is the set of states on the other side of R_i such that the UAV leaves R_i with the same cruise velocity. Each obstacle O_i is assumed to be a closed, convex set and the feasible control input set is assumed to have a box-like structure, where each input has an upper and lower bound. The Region of Influence constraint enforces the planned trajectory to not only avoid the obstacle, but also stay inside R_i before reaching the terminal state in $\mathbb{P}_D$.

Although implementing the framework in (Ref. 1) guarantees the feasibility of (1), solving the optimization problem is nontrivial since it is non-convex with a large number of decision variables. Further, using an off-the shelf solver for numerically calculating the optimal solution can be time consuming, making it unsuitable for online path planning. Therefore, we develop a discretized graph based solution whose search strategy is driven by a heuristic. We reduce the implementation time even further by designing a heuristic that takes the constraints in (1) into consideration.

$$\begin{aligned}
\min_{\mathbf{p}(\cdot),\mathbf{u}(\cdot),t_f} \quad & J(\mathbf{p}(\cdot),t_f), \qquad (1)\\
& \textit{cost function (path length)},\\
s.t. \quad & \dot{\mathbf{p}}(t) = f_s(\mathbf{p}(t),\mathbf{u}(t)),\\
& \textit{differentially flat dynamics of the UAV}\\
& \mathbf{p}(0) = \mathbf{p}_0, \quad \mathbf{p}(t_f) \in \mathbb{P}_d,\\
& \textit{initial and terminal state constraints}\\
& \mathbf{p}(t) \notin O_i, \quad \mathbf{u}(t) \in \mathbb{U} \ \ \forall t \in [0,t_f],\\
& \textit{obstacle avoidance, path and input constraints}\\
& \mathbf{p}(t) \in R_i \ \ \forall t \in [0,t_f],\\
& \textit{Region of Influence constraint}
\end{aligned}$$

GRAPH SEARCH BASED SOLUTION

In this section, we look at how the previously defined constrained optimization problem can be interpreted as an equivalent graph search problem with a discrete action space. We first describe the dynamics model of the UAV and the discretize the feasible control input set. We then explain how the constraints in (1) are incorporated and how A^* (path planning algorithm) can be used for receding horizon planning.

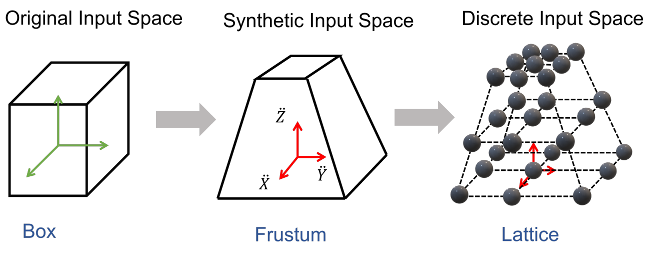
UAV Dynamics Model

As discussed in (Ref. 13), the differentially flat UAV dynamics are shown in Eq. (2). This approach enables us to exploit the equivalent linear double integrator dynamics where the differentially flat state is $\mathbf{q}$ and synthetic control input is $\mathbf{v}$. The relationship between the actual control inputs (total propeller thrust (T_P), roll (ϕ), pitch θ and yawing moment u_ψ) and the synthetic inputs (linear accelerations $\ddot{x}$,$\ddot{y}$,$\ddot{z}$ and yawing acceleration $\ddot{\psi}$) is shown below in Eq. (3).

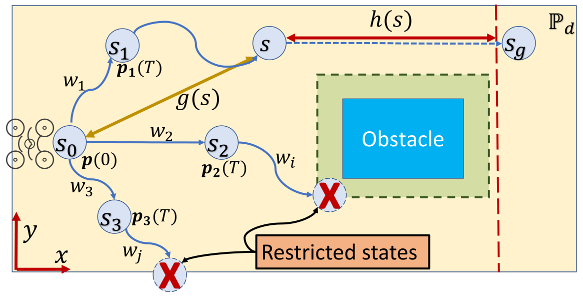
$$\begin{aligned}
\dot{\mathbf{q}} &= f_s(\mathbf{q},\mathbf{v}) = \mathbf{F}\mathbf{q} + \mathbf{G}\mathbf{v}, \qquad (2)\\
\mathbf{q} &= [x,\dot{x},y,\dot{y},z,\dot{z},\psi,\dot{\psi}]^T,\\
\mathbf{v} &= [\ddot{x},\ddot{y},\ddot{z},\ddot{\psi}]^T
\end{aligned}$$

$$\phi = \tan^{-1}\left(\frac{-\ddot{x}\sin\psi + \ddot{y}\cos\psi}{\sqrt{(g-\ddot{z})^2 + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2}}\right), \quad (3)$$
$$\theta = -\tan^{-1}\left(\frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{g-\ddot{z}}\right),$$
$$T_p = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g-\ddot{z})^2},$$
$$u_\psi = \frac{1}{k}\ddot{\psi}$$

Since, constraints on yaw are independent of the rest of the states, we ignore the yaw dynamics. Assuming box-constraints on the original inputs, we define the state, control input excluding yaw as $\mathbf{p} = [x,\dot{x},y,\dot{y},z,\dot{z}]^T$, $\mathbf{u} = [\ddot{x},\ddot{y},\ddot{z}]^T$ respectively and determine the equivalent feasible synthetic input set $\mathbb{U} \subset \mathbb{R}^3$ as discussed in (Ref. 1). This synthetic input set which is independent of yaw, takes the shape of a frustum as shown in Fig. 3a. Later, $\mathbb{U}$ is discretized to obtain lattice representation $\mathbb{W}$. These lattice inputs ($\mathbf{w} \in \mathbb{W}$), when applied for a defined time step T, can be interpreted as primitive actions. The lattice input set $\mathbb{W}$ is as shown in Fig. 3a.



(a) Input space discretization



(b) 2D view of graphical representation

Figure 3: Discretization for graphical representation

Equivalent Graph Search Problem

To define the analogous graph search problem, we represent the state space in a tree structure. A 2D view of this graphical representation is shown in Fig. 3b. First, let us define directed tree $G(\mathbb{S},\mathbb{E})$ with $s_0 \in \mathbb{S}$ as the root vertex. Here, $\mathbb{S},\mathbb{E}$ denotes the set of vertices and the edges of the tree, respectively. Each of these vertices are assigned an specific state of the UAV dynamics. We start with setting the initial state, $\mathbf{p}(0)$ as the underlying state of the root vertex s_0. We apply the lattice inputs to the system for a predefined duration T and the final state is set as the underlying state of a new child vertex of s_0 connected by an edge corresponding to the applied lattice input. When any lattice input $\mathbf{w}$ is applied at the underlying state p_0 of the root vertex s_0 at time t_0, the discretized UAV system dynamics (with zero-order hold) are evaluated for T duration from $t = t_0$ to $t = t_0 + T$ and the resulting system state $\mathbf{p}(t_0+T)$ is identified as,

$$\mathbf{p}(t_0+T) = \begin{bmatrix} 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & T \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{p}(t_0) + \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 \\ 0 & T & 0 \\ 0 & 0 & \frac{1}{2}T^2 \\ 0 & 0 & T \end{bmatrix} \mathbf{w}. \quad (4)$$

We assign $\mathbf{p}(t_0+T)$ to the child vertex of s_0 that is connected to s_0 by the edge corresponding to the applied input. This process is repeated for all child vertices of current vertices by branching out through all inputs $\mathbf{w} \in \mathbb{W}$. This formulation allows both the dynamical constraints and the input constraints to be embedded in the graphical representation itself. Now, the required constraints in the state space such as the initial and final states, Region of Influence and the obstacles can be directly imposed as restrictions on the graph. We do this by restricting access to the vertices reachable from trajectories that do not satisfy the aforementioned constraints as shown in Fig. 3b. Further, all states that satisfy terminal state condition $\mathbf{p}(t_f)$ are defined as the underlying UAV states of the set of goal vertices, $\mathbb{S}_g$. Note that each vertex in G can be traversed from the root by applying a specific sequence of lattice actions. Therefore, the underlying UAV state of each vertex is reachable from the root and for any path that start from the root that exists in G, there is a corresponding trajectory that follows the UAV dynamics. In the graphical representation, the original optimization problem is represented as an equivalent graph search problem. Hence, the solution is to find a path to any goal state s_g from the root. We look at how this solution is developed using $\mathbf{A}^*$ algorithm in the next section.

Shortest Path Solution using A*

In the defined graphical representation, the optimization problem is now represented as a shortest path problem from s_0 to any $s_g \in \mathbb{S}_g$. We propose to use a well-known informed search algorithm A^* (Ref. 9) to find the optimal path. Let $\mu : s \rightarrow \mathbf{p}$ be a mapping that returns the position co-ordinates of the underlying UAV state for a given graph vertex s. Then, we define $d(s,s')$ as the step cost from a vertex s to its child vertex s'. In line with the cost function of the optimization problem in Eq. (1), we define this as the Euclidean distance between the spatial co-ordinates of the underlying UAV states. We can then accumulate the step cost from the start vertex to find the path cost, $g(s)$ from the start s_0 to s. A^* also uses an estimate of the optimal cost-to-go to the goal from a vertex s as the heuristic $h(s)$. The method to compute the heuristic is discussed in the sections ahead. Further, A^* keeps track of two specific costs $g(s)$ and $f(s)$ to conduct selective vertex expansion as defined in Eq. (5). Here, $f(s)$ is the estimated optimal total cost of a path to any $s_g \in \mathbb{S}_g$ through the vertex s. With the costs defined, A^* iteratively compute $f(s)$ for each child (immediate descendant) vertex of the current vertex and always chooses the child vertex with the lowest $f(s)$ for exploration. This process continues until any of the goal states $s_g \in \mathbb{S}_g$ is reached. In multiple obstacle scenarios, we propose that similar search can be conducted in a receding horizon fashion. For A^* algorithm's best-first search to be faster, the heuristic function must correctly reflect the optimal cost-to-go. Hence, heuristic formulation is a critical component in the planning algorithm.

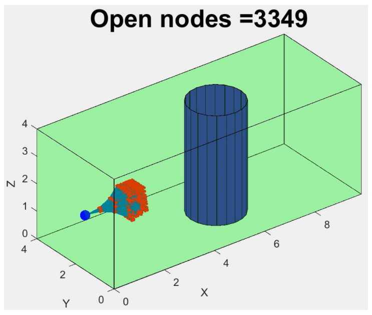
$$\begin{aligned} d(s,s') &\triangleq ||\mu(s') - \mu(s)||, \\ g(s') &= g(s) + d(s,s'), \\ f(s) &\triangleq g(s) + h(s). \end{aligned} \tag{5}$$

HEURISTIC FORMULATION

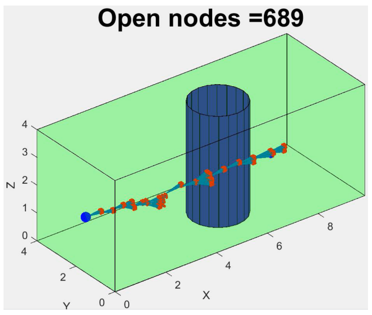
In this section, we discuss the proposed heuristic formulation in detail. The heuristic $h(s)$ provides an estimate of the optimal cost-to-go to any of the goal vertices $s_g \in \mathbb{S}_g$ from a vertex s. A common heuristic formulation is to use the Euclidean distance between spatial co-ordinates of s and s_g i.e. $h(s) = ||\mu(s_g) - \mu(s)||]$ (Ref. 12). However, this estimation fails to encapsulate the obstacle presence in the Region of Influence. Therefore, A^* method takes longer time to provide a solution and engages in unnecessary vertex expansions. This is evident in the planning run shown in Fig. 4a. However, estimating better heuristics that encode the obstacles and take the system dynamics into account would require a significant computation time. This is not feasible when the planner operates online as it has to plan the trajectory before the next Region of Influence is encountered. As a solution, we propose a method for heuristic calculation through rapid planning on a reduced system model.

Heuristic Calculation through Dynamics Relaxation

In this approach, the 2^{nd} order dynamics of the differentially flat UAV system is relaxed to formulate a reduced order system as in $[\dot{x},\ \dot{y},\ \dot{z}]^T = [v_x,\ v_y,\ v_z]^T$. Here, $\bar{\mathbf{p}} = [x,\ y,\ z]^T$ denotes the spatial co-ordinates of the UAV and we assume that the velocities $[v_x,\ v_y,\ v_z]$ can be directly applied as inputs to the system, i.e. we assume accelerations are unbounded and instant. Similar to the previous graphical representation, we define another graph $\bar{G}(\bar{\mathbb{S}}, \bar{\mathbb{E}})$. Following the previously defined process the reduced order system and the states $\bar{\mathbf{p}}$ are encoded to $\bar{G}$ with vertices $\bar{s}$. Now, A^* can be deployed on $\bar{G}$ to find the shortest path to a goal vertex and the Euclidean distance can be used as the heuristic $h_1(\bar{s})$ to estimate the cost-to-go. This heuristic function is suitable as the reduced order system dynamics can be easily evaluated. Therefore, planning can be completed faster. When a solution is found for shortest path in $\bar{G}$, the optimal cost to go to the goal $g_1^*(\bar{s}_g)$ is accumulated by evaluating the action sequence in the shortest path. In this approach, we propose to use $g_1^*(\bar{s}_g)$ as the estimated heuristic for the full order system i.e. $h(s) = g_1^*(\bar{s}_g)$. This estimation provides the A^* method with a better heuristic that encodes the obstacle presence in the Region of Influence for graph G. Therefore, this results in fewer vertex explorations during the planning process as shown in Fig. 4a.



(a) Euclidean Heuristic



(b) Proposed Heuristic

Figure 4: Path search evolution under different heuristics for cylindrical obstacle (radius= $1m$, height= $4m$, centroid= $(4.59, 2, 2)$)

The heuristic formulation is further extended by the maintenance of a tabular memory structure that is used to store the heuristic values for known vertices of the exploration process. This approach leverages storage capacity to gain an advantage in computational time. The memory block is saved at the end of each planning run and then reloaded when the next planning session begins.

Learning the Heuristic Function

It is evident that the proposed formulation enables faster estimation of a more accurate heuristic function. However, it does not reflect the optimal cost yet. We propose to apply a correction to the heuristic that is saved in the memory, allowing the heuristic to be improved towards the optimal cost-to-go at each step of planning. This is conducted based on the following result which is a consequence of acyclic properties of tree structures as in (Ref. 14),

Result 1 Let, s' be any child vertex of a vertex s of a tree structure G. Any path from root to a goal vertex that passes through a child s' must have passed through its parent s. Therefore, if a path from root vertex to a goal vertex exists through s', same path exists through s.

Now, we can use the path cost through s' to correct the estimate of the path cost $f(s)$ for s. To this end, we state the Proposition 1.

Proposition 1 If $f(s') < f(s)$ for any child vertex s' of a parent s, $f(s)$ can be corrected towards the optimal total cost as $f(s)_{correct} = \min_{s' \in child(s)} f(s')$.

Proof of Proposition 1 From Result 1 we know that any path from root to a goal vertex through any child vertex s' exists for its parent s. Therefore, if $f(s') < f(s)$ a better path is available through the child s'. When all children of s' are considered, the best path is the one with minimum cost. Thus, we find the correct path cost for s as $f(s)_{correct} = \min_{s' \in child(s)} f(s')$.

From Proposition 1 and Eq. (5), we can derive the corrected heuristic,

$$h(s)_{correct} = \min_{s' \in child(s)} d(s,s') + h(s'). \tag{6}$$

For each vertex in the explored sub-graph, a corrected heuristic is found and included in the memory. This correction can be shown to converge to the optimal cost-to-go for the explored sub-graph. Further, it can be proven that the Bellman optimality conditions (Ref. 15) holds for this heuristic update. Moreover, the admissibility property of a heuristic function states that the estimated cost-to-go $h(s)$ is an under approximation of the optimal cost to go $h^*(s)$.

$$h(s) \leq h^*(s). \tag{7}$$

Admissibility guarantees an optimal solution by A^* search. Further, a heuristic $h(s)$ is consistent if and only if,

$$h(s) \leq d(s,s') + h(s'), \tag{8}$$

for any parent - child pair s, s'. The consistent property dictates that the search will not do unnecessary vertex visits. The proposed heuristic correction can be proven to preserve the admissibility and it imposes consistency on the corrected heuristic. At each step of exploration, the correction to the heuristic at each vertex is propagated backwards along a path on the tree from the leaves at the frontier to the root. When any goal state is reached this back-propagation corrects the heuristic on the optimal path to optimal cost-to-go which is the ideal heuristic function. When reused in the next planning run, the corrected heuristic greatly reduces the planning time by avoiding unnecessary vertex visits.

Optimality of the Heuristic Update

Let $G' \subset G$ be the explored sub-tree of the tree G and $h^*(s)$ be the optimal cost-to-go to a goal vertex from any vertex s. The optimal cost-to-go is also called the *Value function* in literature. For any child vertex $s' \in G'$ of a parent vertex $s \in G'$, let the optimal cost-to-go be denoted as $h^*(s')$. Then, we have the Bellman optimality condition stated for the parent vertex s as,

$$h^*(s) = \min_{s' \in child(s)} d(s,s') + h^*(s'), \tag{9}$$

with $d(s,s')$ as the step cost.

In the proposed heuristic learning method, the corrected heuristic at a parent vertex $s \in G'$ of child vertices $s' \in G'$ is as follows.

$$h(s)_{correct} = \min_{s' \in child(s)} d(s,s') + h(s'), \tag{10}$$

with $d(s,s')$ as the step cost.

We see that $h(s)_{correct}$ follows the Bellman optimality condition. Therefore, the corrected heuristic is optimal for the explored sub-tree G'. Thus, the proposed heuristic correction preserves the Bellman optimality within the explored sub-tree.

Admissibility Preservation of the Heuristic Update

Admissibility states that any heuristic does not over estimate the cost of reaching the goal. Thus, if the initial heuristic $h(s)$ is admissible, we have $h(s) \leq h^*(s)$ from (7). In the proposed heuristic update in Proposition 1 for any vertex s,

$$f(s)_{correct} \leq f(s), \tag{11}$$
$$g(s) + h(s)_{correct} \leq g(s) + h(s), \tag{12}$$
$$h(s)_{correct} \leq h(s). \tag{13}$$

If the initial heuristic is admissible, then $h(s)_{correct} \leq h^*(s)$. The heuristic update preserves admissibility if started with an admissible heuristic.

Consistency of the Heuristic Update

Consistency states that the heuristic is monotone and satisfies (8). From the heuristic update in (6), we have that,

$$h(s)_{correct} \leq d(s,s') + h(s'). \tag{14}$$

Therefore, the heuristic update imposes the consistency on the heuristic in the explored sub-tree.

RESULTS

Receding Horizon Implementation

The proposed solution is implemented as a receding horizon planner to enable planning for multiple obstacles. The satisfaction of the safety contract ensures that ROIs do not intersect. Thus, a plan is computed for the next ROI, right after the obstacle has been detected while executing the plan for the current ROI. However, the safety contract imposes a requirement on the minimum obstacle detection range denoted

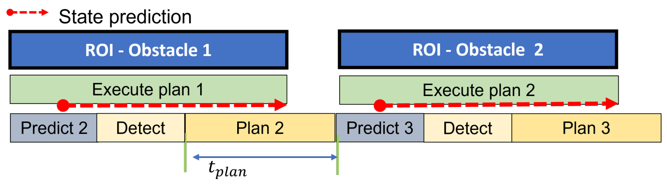


Figure 5: Proposed architecture for receding horizon implementation. Next obstacle must be detected by at least t_{plan} before the next ROI.

as $D_{od}^{contract}$. Previous work in (Ref. 1) evaluated $D_{od}^{contract}$ to be $4m$ analytically. However, it does not take the planning time t_{plan} in to consideration. Therefore, we consider the effect of t_{plan} as it plays a critical role in the implementation the proposed receding horizon planning framework.

As in Fig. 5, suppose a plan is available for the ROI of obstacle 1. While the plan is being executed, the system model is used to predict the exit state. If another obstacle is detected, a plan can be then computed to navigate the next ROI. From the experiments described in the next section, we have empirically evaluated the planning time t_{plan} under varying cruise velocities and initial positions. This requires that the obstacle must be detected at least t_{plan} duration before entering the next ROI. The detection requirement translates in to a required increment to the obstacle detection range by a distance of $\Delta D_{od} = v_{cruise} \times t_{plan}$. Therefore, the minimum required detection range is $D_{od} = D_{od}^{contract} + v_{cruise} \times t_{plan}$. Given that the required obstacle detection range is satisfied, the proposed planning architecture can be implemented as a receding horizon planner for multiple obstacle filled environment. We have analyzed and compared the performance including the required increase of the obstacle detection range for the proposed approach against that of an off-the-shelf optimization problem solver in the next section.

Performance comparison

We present outcomes of the proposed strategy implementation in comparison with an off-the-shelf optimization solver. This simulated case studies was carried out to evaluate the planning time of the proposed approach and the trade off of the optimality of the trajectory. Further, we have evaluated the proposed planning algorithm on the minimum required obstacle detection range. The goal of the proposed approach is to develop a faster and feasible solution by trading-off the optimality of the solution. We compare the proposed approach with CasADi which is an off-the-shelf solver for non-linear optimization problems.

Experiment setup In this analysis, we used a work space of pillar type cylindrical obstacle as in Fig. 10b. We observed that under the variations in the initial z co-ordinate, the problem set up is invariant due to the obstacle shape. We varied the initial y co-ordinate such that $y_0 \in (1m, 3m)$ and the cruise velocity in positive x direction such that $v_{cruise} \in (1m/s, 2.8m/s)$ to create 1000 initial states. In the tested scenario, the goal

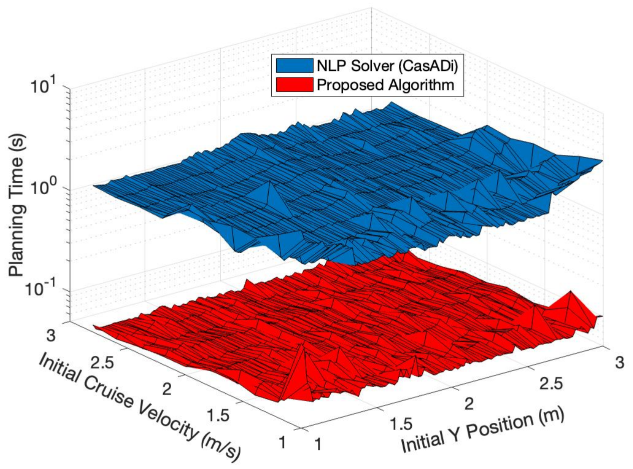


Figure 6: Comparison of planning time between the proposed algorithm (max $= 0.3531s$) and Non-Linear Program (NLP) solver (max $= 6.38s$) for $y_0 \in (1m, 3m)$ and $v_{cruise} \in (1m/s, 2.8m/s)$ for a cylindrical obstacle (radius$=$ $1m$, height$= 4m$, centroid$= (4.59, 2, 2)$).

was to get past $8m$ in x direction. This experiment was conducted using MATLAB ver. R2021a on a workstation with an Intel core i7-8700 CPU running at $3.2GHz$ with $16GB$ of volatile memory.

Planning time We evaluate the planning time for each initial state using CasADi solver and the proposed approach. Included Fig. 6 shows the variation of the planning time against initial cruise velocity and y position. We observe that the proposed approach has more than 15 times faster planning times on average when compared against the optimization problem solver. The run time for each of the planning runs are within sub-second levels for the proposed algorithm. Further, faster planning times allow a reduction in the required expansion of the obstacle detection range. This is critical in implementing the receding horizon planner. Therefore, the proposed approach is suitable for a realtime implementation in an online receding horizon planning framework when multiple obstacles are present as evident through the presented results.

Sub-optimality The discretization of the inputs and the state space to construct the graphical representation contributes to a sub-optimality in the resulting trajectories of the proposed approach. We present this trade off in Fig. 7 as a comparison of resulting trajectory length of the proposed approach against the optimization problem solver. We observe a $< 20\%$ sub-optimality in the proposed approach. However, this trade off is tolerable as the resulting trajectory is feasible and the plan is computed with sub-second planning times allowing receding horizon implementation with a lower obstacle detection range even at higher velocities.

Obstacle detection range As described in the Receding horizon implementation, planning for multiple obstacles requires a minimum obstacle detection range of $(D_{od}^{contract} +$

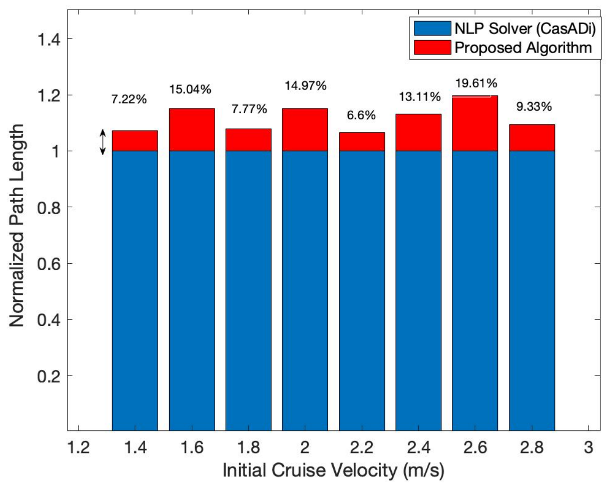


Figure 7: Sub-optimality in path length of trajectories from the proposed algorithm compared with the of path length of trajectories from Non-Linear Program (NLP) solver for $[x_0, y_0, z_0] = [0, 2, 2]$ and $v_{cruise} \in (1.4m/s, 2.8m/s)$ for a cylindrical obstacle (radius$= 1m$, height$= 4m$, centroid$= (4.59, 2, 2)$).

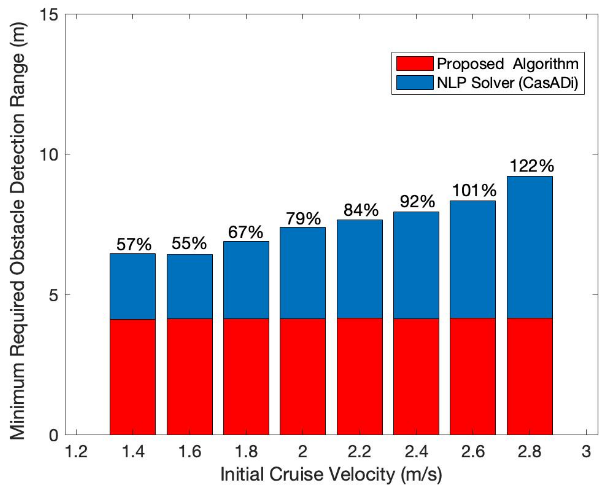


Figure 8: Minimum required obstacle detection range, $(D_{od}^{contract} + \Delta D_{od})$ where $D_{od}^{contract} = 4m$, $\Delta D_{od} = v_{cruise} \times t_{plan}$ for the proposed algorithm compared with the Non-Linear Program (NLP) solver for $[x_0, y_0, z_0] = [0, 2, 2]$ and $v_{cruise} \in (1.4m/s, 2.8m/s)$ for a cylindrical obstacle (radius$= 1m$, height$= 4m$, centroid$= (4.59, 2, 2)$).

$\Delta D_{od})$ where $D_{od}^{contract} = 4m$, $\Delta D_{od} = v_{cruise} \times t_{plan}$ as a plan must be produced before the UAV reaches the ROI. The minimum required obstacle detection range for the proposed algorithm and the Non-linear program solver are compared in Fig. 8. We see that the using the optimization solver on-board in the receding horizon routine would require the UAV to have at least 50% longer obstacle detection range than what the proposed algorithm would require. Further, we observe that this increase would be even more (up to 120%) for higher cruise velocities. As the required increment of obstacle detection range is smaller in the proposed approach, we visualize it in Fig. 9 in comparison with $D_{od}^{contract}$. We observe that the proposed method only requires an increment less than 4% of $D_{od}^{contract}$ even with higher cruise velocities. Therefore, due to the faster planning times, incorporating the proposed approach into the planner would enable the UAV to fly with higher cruise velocities inside the obstacle field where visibility is limited.

Further, a resulting planned trajectory from the proposed approach for initial position of $[x_0, y_0, z_0] = [0, 2, 2]$ is shown in Fig. 10b and Fig. 10c. It is observed that the planned trajectories avoid the obstacle successfully and they are the feasible trajectories for the utilized UAV model as the system dynamics are not violated in the planning process.

CONCLUSIONS

In this paper, we have proposed and demonstrated a heuristic based motion planning strategy for UAVs with fast planning times. Based on previous work, multiple obstacle avoidance problem was decoupled to formulate single obstacle avoidance problem using the idea of region of influence. The planning is then represented as optimization problem constrained by the UAV dynamics, ROI bounds and the obstacle. We proposed a graphical representation of the planning problem where the state, dynamics and the inputs are incorporated in to the graph structure. This allows the use of heuristic based graph search approaches like A^* to compute feasible solutions faster. In such approaches, the design of the heuristic function plays a critical role. We have proposed a heuristic calculation based on dynamics relaxation of the original problem. When the exploration is conducted, the calculated heuristic is dynamically corrected to satisfy Bellman optimality of the explored sub-graph. Further, we maintain a memory of the previous heuristic values to help fast search. We also propose a receding horizon planning framework for multi obstacle avoidance.

The proposed approach showed superior planning times when compared against an off-the-shelf nonlinear program solver with minor sub-optimality. Also, the studies validated that the proposed approach is implementable in a receding horizon framework with minimal required obstacle detection range when compared against the NLP solver.

In conclusion, the proposed heuristic based planner was able to achieve accelerated motion planning supporting a receding horizon implementation framework. In addition, the proposed heuristic calculation and correction method allows it to be extended towards complex planning scenarios. Further, we find that it is important to evaluate the proposed method in the presence of disturbances and model uncertainties in the future.

ACKNOWLEDGMENTS

This work was carried out at the Rensselaer Polytechnic Institute under the Army/Navy/NASA Vertical Lift Research

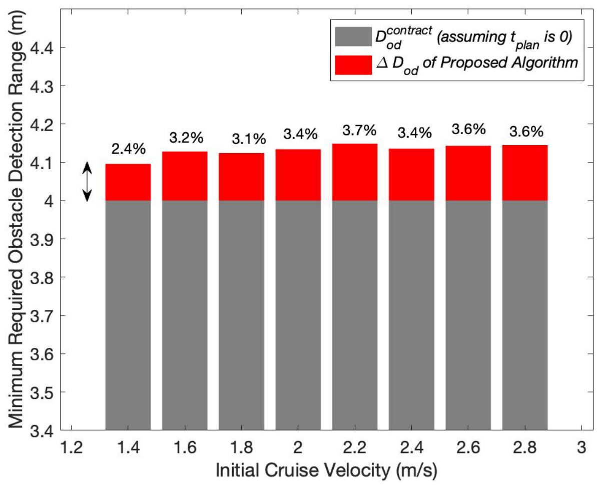


Figure 9: Minimum required obstacle detection range, $(D_{od}^{contract} + \Delta D_{od})$ where $D_{od}^{contract} = 4m$, $\Delta D_{od} = v_{cruise} \times t_{plan}$, for the proposed algorithm for $[x_0, y_0, z_0] = [0, 2, 2]$ and $v_{cruise} \in (1.4m/s, 2.8m/s)$ for a cylindrical obstacle (radius$= 1m$, height$= 4m$, centroid$= (4.59, 2, 2)$).

Center of Excellence (VLRCOE) program, grant number W911W61120012, with Dr. Mahendra Bhagawat as Technical Monitor.

REFERENCES

1. Nallan, K., Mishra, S., and Julius, A., "An Assume-Guarantee Framework for Multiple-Obstacle Collision Avoidance," *Vertical Flight Society 77th Annual Forum*, held online, 2021.

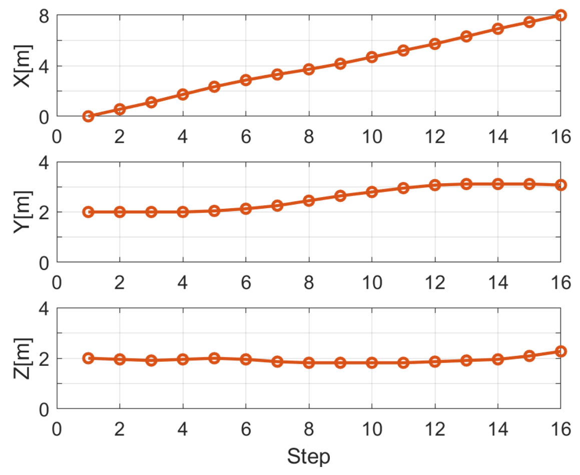
2. Yang, L., Qi, J., Xiao, J., and Yong, X., "A literature review of UAV 3D path planning," Proceeding of the 11th World Congress on Intelligent Control and Automation, 2014. DOI: 10.1109/WCICA.2014.7053093

3. Shahid, N., Abrar, M., Ajmal, U., Masroor, R., Amjad, S., and Jeelani, M., "Path planning in unmanned aerial vehicles: An optimistic overview," *International Journal of Communication Systems*, Vol. 35, (6), 2022, pp. e5090. DOI: https://doi.org/10.1002/dac.5090

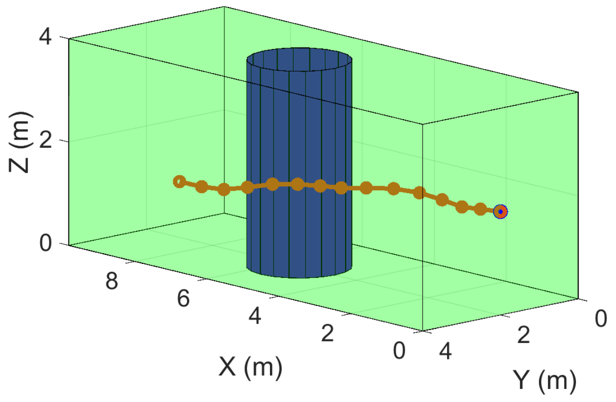
4. Pivtoraiko, M., Mellinger, D., and Kumar, V., "Incremental micro-UAV motion replanning for exploring unknown environments," 2013 IEEE International Conference on Robotics and Automation, 2013. DOI: 10.1109/ICRA.2013.6630910

5. Goerzen, C., Kong, Z., and Mettler, B., "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, (1-4), 2009, pp. 65–100. DOI: 10.1007/s10846-009-9383-1

6. Geißer, F., Povéda, G., Trevizan, F., Bondouy, M., Teichteil-Königsbuch, F., and Thiébaux, S., "Optimal



(a) Planned trajectory for $x_0 = 0$, $y_0 = z_0 = 2$



(b) Planned trajectory in the work space

Figure 10: Results of the proposed planning algorithm for a cylindrical obstacle (radius$= 1m$, height$= 4m$, centroid$= (4.59, 2, 2)$) initial point$= (0, 2, 2)$.

and Heuristic Approaches for Constrained Flight Planning under Weather Uncertainty," *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 30, (1), Jun. 2020, pp. 384–393.

7. Cohen, B. J., Chitta, S., and Likhachev, M., "Search-based planning for manipulation with motion primitives," 2010 IEEE International Conference on Robotics and Automation, 2010. DOI: 10.1109/ROBOT.2010.5509685

8. Pivtoraiko, M., and Kelly, A., "Kinodynamic motion planning with state lattice motion primitives," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011. DOI: 10.1109/IROS.2011.6094900

9. Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Sci-*

ence and Cybernetics, Vol. 4, (2), 1968, pp. 100–107. DOI: 10.1109/TSSC.1968.300136

10. Corrêa, A. B., Francès, G., Pommerening, F., and Helmert, M., “Delete-Relaxation Heuristics for Lifted Classical Planning,” *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 31, (1), May 2021, pp. 94–102.

11. Kaur, J., Chatterjee, I., and Likhachev, M., “Speeding Up Search-Based Motion Planning using Expansion Delay Heuristics,” *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 31, (1), May 2021, pp. 528–532.

12. Ferguson, D., Likhachev, M., and Stentz, A., “A guide to heuristic-based path planning,” Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS), 2005.

13. Alimbayev, T., Moy, N., Nallan, K., Mishra, S., and Julius, A., “A contract based approach to colllision avoidance for UAVs,” *Vertical Flight Society 76th Annual Forum*, held online, 2020.

14. Reinhard, D., *Trees and Forests*, Springer Graduate Texts in Mathematics, Springer-Verlag, fifth edition, 2017, p. 12–20.

15. Bellman, R., “On the Theory of Dynamic Programming,” *Proceedings of the National Academy of Sciences*, Vol. 38, (8), 1952, pp. 716–719. DOI: 10.1073/pnas.38.8.716