Chapter 3

Path Planning in the Local-Level Frame for Small Unmanned Aircraft Systems

Laith R. Sahawneh and Randal W. Beard

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71895

Abstract

In this chapter, we propose a 3D path planning algorithm for small unmanned aircraft systems (UASs). We develop the path planning logic using a body fixed relative coordinate system which is the unrolled, unpitched body frame. In this relative coordinate system, the ownship is fixed at the center of the coordinate system, and the detected intruder is located at a relative position and moves with a relative velocity with respect to the ownship. This technique eliminates the need to translate the sensor's measurements from local coordinates to global coordinates, which saves computation cost and removes the error introduced by the transformation. We demonstrate and validate this approach using predesigned encounter scenarios in the Matlab/Simulink environment.

Keywords: small unmanned aircraft systems, path planning, collision avoidance, cell decomposition, Dijkstra's search algorithm

1. Introduction

The rapid growth of the unmanned aircraft systems (UASs) industry motivates the increasing demand to integrate UAS into the U.S. national airspace system (NAS). Most of the efforts have focused on integrating medium or larger UAS into the controlled airspace. However, small UASs weighing less than 55 pounds are particularly attractive, and their use is likely to grow more quickly in civil and commercial operations because of their versatility and relatively low initial cost and operating expense.

Currently, UASs face limitations on their access to the NAS because they do not have the ability to sense-and-avoid collisions with other air traffic [1]. Therefore, the Federal Aviation Administration (FAA) has mandated that UASs were capable of an equivalent level of safety to the see-and-avoid (SAA) required for manned aircraft [2, 3]. This sense-and-avoid (SAA)

mandate is similar to a pilot's ability to visually scan the surrounding airspace for possible intruding aircraft and take action to avoid a potential collision.

Typically, a complete functional sense-and-avoid system is comprised of sensors and associated trackers, collision detection, and collision avoidance algorithms. In this chapter, our main focus is on collision avoidance and path planning. Collision avoidance is an essential part of path planning that involves the computation of a collision-free path from a start point to a goal point while optimizing an objective function or performance metric. A robust collision avoidance logic considers the kinematic constraints of the host vehicle, the dynamics of the intruder's motion, and the uncertainty in the states estimate of the intruder. The subject of path planning is very broad, and in particular collision, avoidance has been the focus of a significant body of research especially in the field of robotics and autonomous systems. Kuchar and Yang [4] provided a detailed survey of conflict detection and resolution approaches. Albaker and Rahim [5] conducted a thorough survey of collision avoidance methods for UAS. The most common collision avoidance methods are geometric-based guidance methods [6–13], potential field methods [14, 15], sampling-based methods [16, 17], cell decomposition techniques, and graph-search algorithms [18–20].

Geometric approaches to collision avoidance are straightforward and intuitive. They lend themselves to fast analytical solutions based on the kinematics of the aircraft and the geometry of the encounter scenario. The approach utilizes the geometric relationship between the encountering aircraft along with intuitive reasoning [8, 21]. Generally, geometric approach assumes a straight-line projection to determine whether the intruder will penetrate a virtual zone surrounding an ownship. Then, the collision avoidance can be achieved by changing the velocity vector, assuming a constant velocity model. Typically, geometric approaches do not account for uncertainty in intruder flight plans and noisy sensor information.

The potential field method is another widely used approach for collision avoidance in robotics. A typical potential field works by exerting virtual forces on the aircraft, usually an attractive force from the goal and repelling forces from obstacles or nearby air traffic. Generally, the approach is very simple to describe and easy to implement. However, the potential field method has some fundamental issues [22]. One of these issues is that it is a greedy strategy that is subject to local minima. However, heuristic developments to escape the local minima are also proposed in the literature [23]. Another problem is that typical potential field approaches do not account for obstacle dynamics or uncertainly in observation or control. In the context of airborne path planning and collision avoidance, Bortoff presents a method for modeling a UAS path using a series of point masses connected by springs and dampers [24]. This algorithm generates a stealthy path through a set of enemy radar sites of known locations. McLain and Beard present a trajectory planning strategy suitable for coordinated timing for multiple UAS [25]. The paths to the target are modeled using a physical analogy of a chain. Similarly, Argyle et al. present a path planner based on a simulated chain of unit masses placed in a force field [26]. This planner tries to find paths that go through maxima of an underlying bounded differentiable reward function.

Sampling-based methods like probability road maps (PRM) [16] and rapidly exploring random trees (RRTs) [17] have shown considerable success for path planning and obstacle

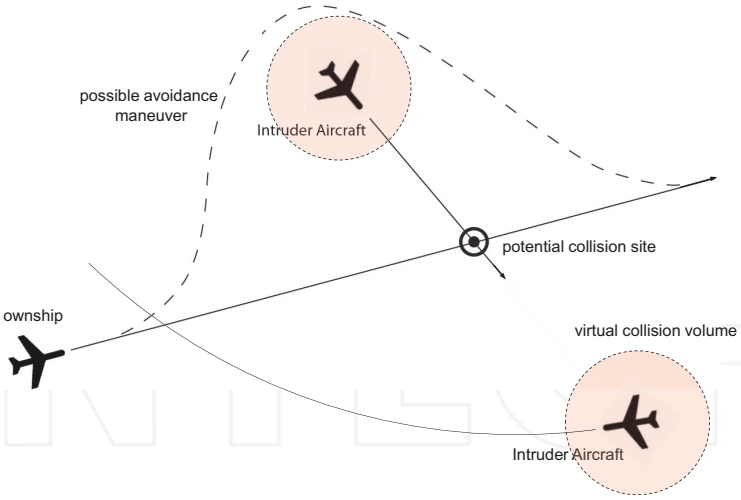

Figure 1. The geometry of an encounter scenario.

avoidance, especially for ground robots. They often require significant computation time for replanning paths, making them unsuitable for reactive avoidance. However, recent extensions to the basic RRT algorithm, such as chance-constrained RRT* [27] and close-loop RRT [28], show promising results for uncertain environments and nontrivial dynamics [28–30]. Cell decomposition is another widely used path planning approach that partitions the free area of the configuration space into cells, which are then connected to generate a graph [20]. Generally, cell decomposition techniques are considered to be global path planners that require a priori knowledge of the environment. A feasible path is found from the start node to the goal node by searching the connectivity graph using search algorithms like A* or Dijkstra's algorithm [18].

The proposed approach in this work will consider encounter scenarios such as the one depicted in **Figure 1**, where the ownship encounters one or more intruders. The primary focus of this work is to develop a collision avoidance framework for unmanned aircraft. The design, however, will be specifically tailored for small UAS. We assume that there exists a sensor(s) and tacking system that provide states estimate of the intruder's track.

2. Local-level path planning

A collision event occurs when two aircraft or more come within the minimum allowed distance between each other. The current manned aviation regulations do not provide an explicit value for the minimum allowed distance. However, it is generally understood that the minimum allowed or safe distance is required to be at least 500 ft. to 0.5 nautical miles (nmi) [21, 31]. For example, the near midair collision (NMAC) is defined as the proximity of less than 500 ft. between two or more aircraft [32]. Similarly and since the potential UAS and intruder aircraft

cover a wide range of vehicle sizes, designs, airframes, weights, etc., the choice of a virtual fixed volume boundary around the aircraft is a substitute for the actual dimensions of the intruder.

As shown in **Figure 2**, the choice for this volume is a *hockey-puck* of radius d_s and height h_s that commonly includes a horizontal distance of 500 ft. and a vertical range of 200 ft. [1, 33, 34]. Accordingly, a collision event is defined as an incident that occurs when two aircraft pass less than 500 ft. horizontally and 100 ft. vertically.

In this work, we develop the path planning logic using a body-centered relative coordinate system. In this body-centered coordinate system, the ownship is fixed at the center of the coordinate system, and the intruder is located at a relative position $\mathbf{p}_r$ and moves with a relative velocity $\mathbf{v}_r$ with respect to the ownship [35].

We call this body-centered coordinate frame the local-level frame because the environment is mapped to the unrolled, unpitched local coordinates, where the ownship is stationary at the center. As depicted in **Figure 3**, the origin of the local-level reference is the current position of the ownship. In this configuration, the x-axis points out the nose of the unpitched airframe, the y-axis points points out the right wing of the unrolled airframe, and the z-axis points down forming a right-handed coordinate system. In the following discussion, we assume that the collision volume is centered at the current location of the intruder. A collision occurs when the origin of the local-level frame penetrates the collision volume around the intruder.

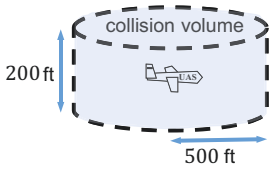

Figure 2. A typical collision volume or protection zone is a virtual fixed volume boundary around the aircraft.

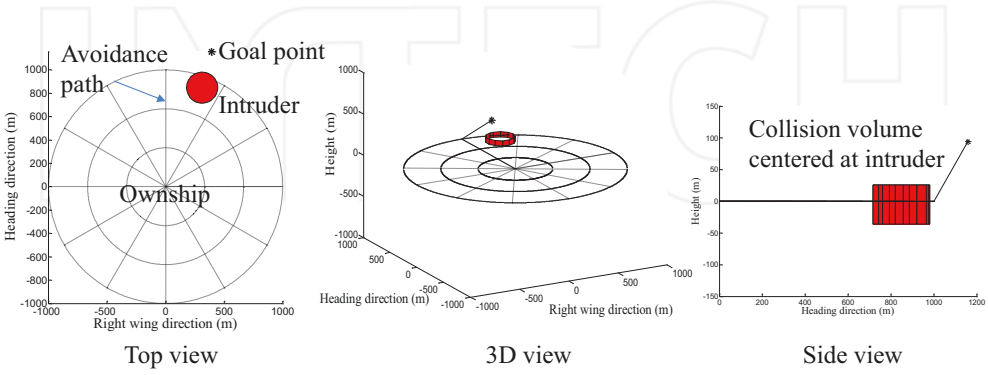

Figure 3. Local-level reference frame.

The detection region is divided into concentric circles that represent maneuvers points at increasing range from the ownship as shown in **Figure 4**, where the radius of the outmost circle can be thought of as the sensor detection range. Let the region in the space covered by the sensor be called the workspace. Then, this workspace is discretized using a cylindrical grid in which the ownship is commanded to move along the edges of the grid. The result is a directed weighted graph, where the edges represent potential maneuvers, and the associated weights represent the maneuver cost and collision risk. The graph can be described by the tuple $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{C})$, where $\mathcal{N}$ is a finite nonempty set of nodes, and $\mathcal{E}$ is a collection of ordered pairs of distinct nodes from $\mathcal{N}$ such that each pair of nodes in $\mathcal{E}$ is called a directed edge or link, and $\mathcal{C}$ is the cost associated with traversing each edge.

The path is then constructed from a sequence of nonrepeated nodes $(\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_N)$ such that each consecutive pair $(\mathbf{n}_i, \mathbf{n}_{i+1})$ is an edges in $\mathcal{G}$. Let the detection range d_r be the radius of the outermost circle, and r be the radius of the innermost circle so that $d_r = mr$. As shown in **Figure 6**, let $\mathcal{L}_l$, $l = 1, 2, \cdots, m$ be the lth level curve of the concentric circles. Assume that the level curves are equally partitioned by a number of points or nodes such that any node on the lth level curve, $\mathcal{L}_l$ connects to a predefined number of nodes k in the next level, that is, in the forward direction along the heading axis as depicted in **Figure 4**. The nodes on the graph can be thought of as predicted locations of the ownship over a look-ahead time window. Additionally, we assume that only nodes along the forward direction of the heading axis, that is, $x = 0$ connect to nodes in the vertical plane. This assumption allows to command the aircraft to climb or descend by connecting to nodes in the vertical plane as shown in **Figure 4**. Let the first level curve of the innermost circle be discretized into $|\mathcal{L}_1| = k + 2$ nodes including nodes in the vertical plane. Then, using the notation $|\mathcal{A}|$ to denote the cardinality of the discrete set $\mathcal{A}$, the number of nodes in the lth level curve is given by

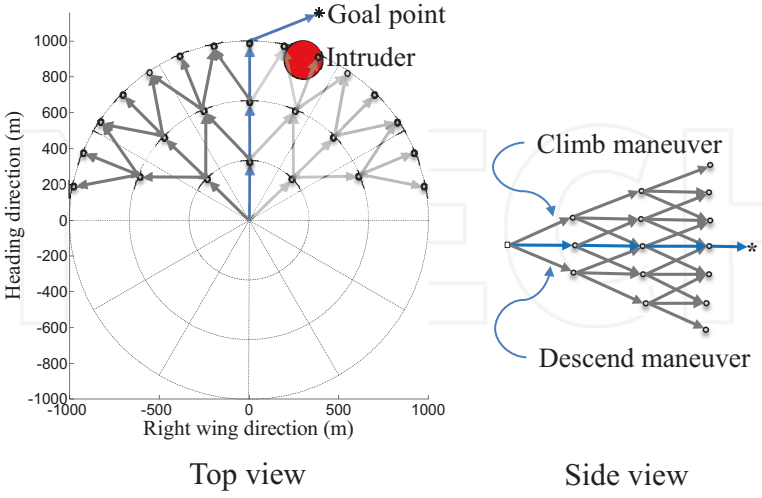

Figure 4. Discretized local-level reference workspace. The three concentric circles represent three maneuvers points.

$$|\mathcal{L}_l| = \begin{cases} k+2 & \text{if } l = 1, \\ 2|\mathcal{L}_{l-1}| + 2l + 1 & \text{if } l = 2, 3, \cdots, m, \end{cases} \tag{1}$$

where the total number of nodes is $|\mathcal{N}| = \sum_{l=1}^{m} |\mathcal{L}_l|$. For example, assuming that the start node is located at the origin of the reference map and given that $k = 3$, that is, allowing the ownship to fly straight or maneuver right or left. The total number of nodes in the graph including the start and destination node is given by

$$|\mathcal{N}| = \left(\sum_{l=1}^{m+1} 2^l + 2l - 3 \right) + 1. \tag{2}$$

Figure 5 shows an example of a discretized local-level map. In this example, $k = 3$ and $m = 3$, and the total number of nodes in the graph $|\mathcal{N}|$ is 39.

Assuming that the ownship travels between the nodes with constant velocity and climb rate, the location of the ith node at the lth level curve, and $\mathbf{n}_{i,l}$ in the horizontal plane of the graph is given by

$$\mathbf{n}_{i,l} = \left[lr \sin \psi_j^{\mathcal{L}_l}, lr \cos \psi_j^{\mathcal{L}_l}, 0 \right]^T, \tag{3}$$

where $\psi_j^l = \frac{j\psi^d}{2^{(l-1)}}$ and $j = \left\{ -\frac{|\mathcal{L}_l|-1}{2}, -\frac{|\mathcal{L}_l|-1}{2} + 1, \cdots, \frac{|\mathcal{L}_l|-1}{2} - 1, \frac{|\mathcal{L}_l|-1}{2} \right\}$ and ψ^d is the allowed heading. In the vertical plane, the location of nodes is $\mathbf{n}_{\bar{j},l} = \left[0, 0, \pm \bar{j} l h^d \right]^T$, where $\bar{j} = \{1, 2, \cdots, l\}$ and h^d are the altitude change at each step as shown in **Figure 6**.

For example, if $\psi^d = \pi/4$, $h^d = 50$ m, $r = 500$ m, $k = 3$, and $|\mathcal{L}_1| = 5$, then we have $j = \{-1, 0, 1\}$, $\bar{j} = \{-1, 1\}$, $\psi_j^1 = \{-\pi/4, 0, -\pi/4\}$, and the locations of nodes at $\mathcal{L}_1$ in the

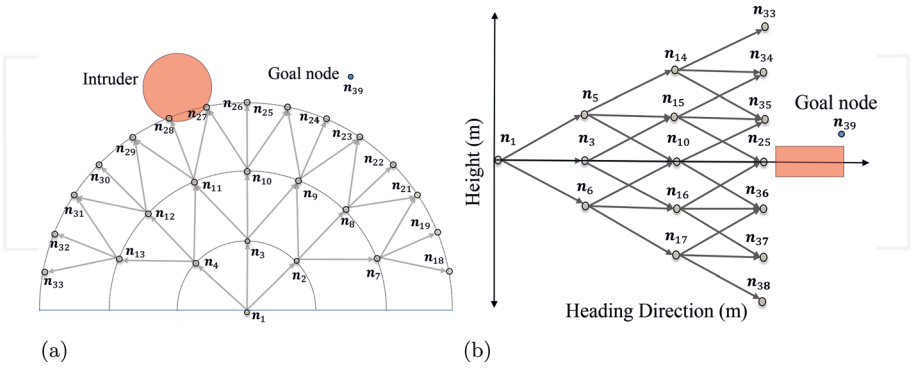

Figure 5. Example of discretized local-level map. (a) Top view: location and index of nodes and (b) side view: location and index of nodes.

http://dx.doi.org/10.5772/intechopen.71895

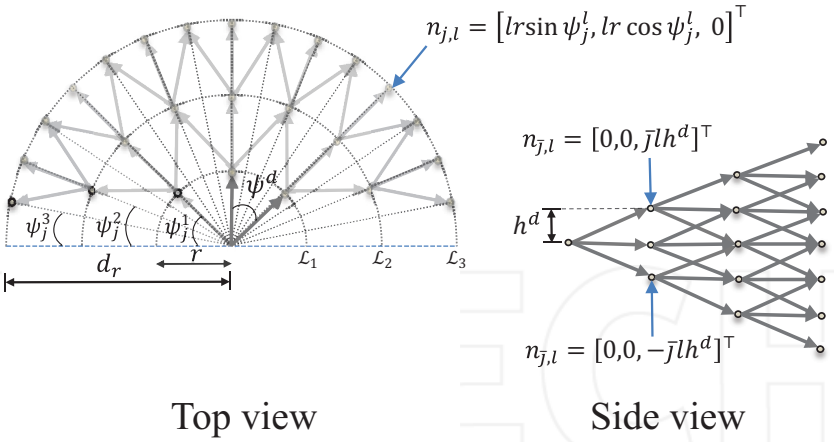

Figure 6. Nodes location in the local-level reference frame.

horizontal plane are $\left\{(-500\sin\pi/4, 500\cos\pi/4, 0)^T,\ (0,500,0)^T,\ (500\sin\pi/4, 500\cos\pi/4, 0)^T\right\}$, and in the vertical plane are $\left\{(0,0,50)^T, (0,0,-50)^T\right\}$.

The main priority of the ownship where it is under distress is to maneuver to avoid predicted collisions. This is an important note to consider when assigning a cost of each edge in the resulting graph. The cost associated with traveling along an edge is a function of the edge length and the collision risk. The cost associated with the length of the edge $e_{i,i+1}$ that connects between the consecutive pair nodes $(\mathbf{n}_i, \mathbf{n}_{i+1})$ is simply the Euclidean distance between the nodes $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$ expressed as

$$C_L(e_{i,i+1}) = \|\mathbf{n}_{i+1} - \mathbf{n}_i\|. \tag{4}$$

The collision cost for traveling along an edge is determined if at any future time instant, the future position of the ownship along that edge is inside the collision volume of the predicted location of an intruder. An exact collision cost computation would involve the integration of collision risk along each edge over the look-ahead time window $\tau \in [t, t+mT]$.

A simpler approach involves calculating the collision risk cost at several locations along each edge, taking into account the projected locations of the intruder over the time horizon τ. Assuming a constant velocity model, a linear extrapolation of the current position and velocity of the detected intruders are computed at evenly spaced time instants over the look-ahead time window. The look-ahead time interval is then divided into several discrete time instants. At each discrete time instant, all candidate locations of the ownship along each edge are checked to determine whether it is or will be colliding with the propagated locations of the intruders. For the simulation results presented in this chapter, the collision risk cost is calculated at three

points along each edge in $\mathcal{G}$. If v_o is the speed of the ownship, then the distance along an edge is given by $v_o T$, where $T = r/v_o$. The three points are computed as

$$\mathbf{p}_1 = \mathbf{n}_i + \mathbf{v}_o T_s \frac{\mathbf{n}_{i+1} - \mathbf{n}_i}{\|\mathbf{n}_{i+1} - \mathbf{n}_i\|}, \tag{5}$$

$$\mathbf{p}_2 = \mathbf{p}_1 + v_o T_s \frac{\mathbf{n}_{i+1} - \mathbf{n}_i}{\|\mathbf{n}_{i+1} - \mathbf{n}_i\|}, \tag{6}$$

$$\mathbf{p}_3 = \mathbf{p}_2 + v_o T_s \frac{\mathbf{n}_{i+1} - \mathbf{n}_i}{\|\mathbf{n}_{i+1} - \mathbf{n}_i\|}, \tag{7}$$

where $T_s = T/3$. Let the relative horizontal and vertical position of the intruder with respect to the ownship at the current time t be $\mathbf{p}_r(t)$ and $p_{r_z}(t)$, respectively. Define the collision volume as

$$\mathcal{C}(\mathbf{p}_r(t)) = \left\{ d \in \mathbb{R}^2 : \|\mathbf{p}_r(t)\| - d \leq d_s \text{ and } h \in \mathbb{R} : \left| p_{r_z} - h \right| \leq h_s/2 \right\}. \tag{8}$$

The predicted locations of each detected intruder over time horizon T at three discrete time samples T_s are

$$\mathbf{p}_{r_{3D}}(t + (1 + 3(l-1))T_s) = \mathbf{p}_{r_{3D}}(t) + \mathbf{v}_{r_{3D}}(t)(1 + 3(l-1))T_s, \tag{9}$$

$$\mathbf{p}_{r_{3D}}(t + (2 + 3(l-1))T_s) = \mathbf{p}_{r_{3D}}(t) + \mathbf{v}_{r_{3D}}(t)(2 + 3(l-1))T_s, \tag{10}$$

$$\mathbf{p}_{r_{3D}}(t + (3 + 3(l-1))T_s) = \mathbf{p}_{r_{3D}}(t) + \mathbf{v}_{r_{3D}}(t)(3 + 3(l-1))T_s, \tag{11}$$

where $\mathbf{p}_{r_{3D}}(t) = \left[\mathbf{p}_r(t), p_{r_z}(t) \right]^T \in \mathbb{R}^3$ and $\mathbf{v}_{r_{3D}}(t) = [v_r(t), \mathbf{v}_{r_z}(t)]^T \in \mathbb{R}^3$ be the 3D relative position and velocity of the intruder with respect to the ownship in the relative coordinate system, where $\mathbf{v}_r(t)$ and $v_{r_z}(t)$ are the relative horizontal velocity and vertical speed at the current time t.

In Eqs. (9)–(11), if $e_{i,i+1}$ is the current edge being evaluated, then the node $\mathbf{n}_{i+1}$ determines the value of l. In other words, if $\mathbf{n}_{i+1} \in \mathcal{L}_1$, then $l = 1$. For example, if we are to compute the three points along the edge $e_{1,2}$ in Eqs (5)–(7), then $\mathbf{n}_2 \in \mathcal{L}_1$ and $l = 1$. Using the definition of the binary cost function, the collision risk cost associated with the $e_{i,i+1}$ edge with respect to each detected intruder is given by the expression

$$C_{col}(\text{int}, e_{i,i+1}) = \begin{cases} \infty & \text{if any of } \mathbf{p}_1, \mathbf{p}_2, \text{ or } \mathbf{p}_3 \in \mathcal{C}\left(\mathbf{p}_{r_{3D}}(t + (\ell + 3(l-1))T_s)\right), \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $\ell = \{1, 2, 3\}$. In Eq. (12), the ∞ or the maximum allowable cost is assigned to any edge that leads to a collision, basically eliminating that edge and the path passing through it. The total collision risk associated with the ith edge is given by

$$C_{col}(e_{i,i+1}) = \sum_{\text{int}=1}^{M} C_{col}(\text{int}, e_{i,i+1}), \tag{13}$$

where M is the number of detected intruders.

A visual illustration of the collision risk computation is shown in **Figure 7**. The propagated collision volume of a detected intruder and the candidate locations of the ownship over the first-time interval $[t+T_s, t+3T_s]$ both in the horizontal and vertical plane is depicted in **Figure 7a** and **b**. Clearly, there is no intersection between these candidate points the ownship may occupy and the propagated locations of the collision volume over the same interval. Then, according to Eq. (13), the cost assigned to these edges is zero. Next, all candidate locations of the ownship along each edge over the second time interval $[t+4T_s, t+6T_s]$ are investigated. As shown in **Figure 7c**, edges $e_{2,7}$, $e_{2,8}$, and $e_{2,9}$ intersect with the predicted intruder location at time $t+4T_S$ and $t+5T_S$, respectively. Similarly, edges $e_{3,15}$ and $e_{3,16}$ in the horizontal plane intersect with the predicted intruder location at time $t+4T_S$ as shown in **Figure 7d**. Accordingly, the maximum allowable costs will be assigned to these edges, which eliminate these edges and the path passing through them. All the candidate locations of the ownship over the time interval $[t+7T_s, t+9T_s]$ do not intersect with the predicted locations of the intruder as shown in **Figure 7e** and **f**. Therefore, by the time, the ownship will reach these edges the detected intruder will be leaving the map, and consequently, a cost of zero is assigned to edges belonging to the third level curve $\mathcal{L}_3$.

To provide an increased level of robustness, an additional threat cost is added to penalize edges close to the propagated locations of the intruder even if they are not within the collision volume. At each discrete time instant, we compute the distances from the candidate locations of the ownship to all the propagated locations of the intruders at that time instant. The cost of collision threat along each edge is then given by the sum of the reciprocal of the associated distances to each intruder

$$C_{th}(\text{int}, e_{i,i+1}) = \frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3}. \tag{14}$$

where d_1, d_2, and d_3 are given by

$$d_1 = \left\| p_1 - \mathbf{p}_{r_{3D}}(t + (1 + 3(l-1))T_s) \right\|,$$

$$d_2 = \left\| p_2 - \mathbf{p}_{r_{3D}}(t + (2 + 3(l-1))T_s) \right\|,$$

$$d_3 = \left\| p_3 - \mathbf{p}_{r_{3D}}(t + (3 + 3(l-1))T_s) \right\|,$$

and the total collision risk cost associated with the ith edge with regard to all intruders is given by

$$C_{th}(e_{i,i+1}) = \sum_{\text{int}=1}^{M} C_{th}(\text{int}, e_{i,i+1}). \tag{15}$$

For example, the edges $e_{1,2}$, $e_{1,3}$, $e_{1,4}$, $e_{1,5}$, and $e_{1,6}$ shown in **Figure 7a** are not intersecting with the propagated collision volume locations over the first-time interval, yet they will be penalized based on their distances to the predicated locations of the intruder according to Eq. (15). Note that edge $e_{1,2}$ will have greater cost as it is the closest to the intruder among other candidate edges.

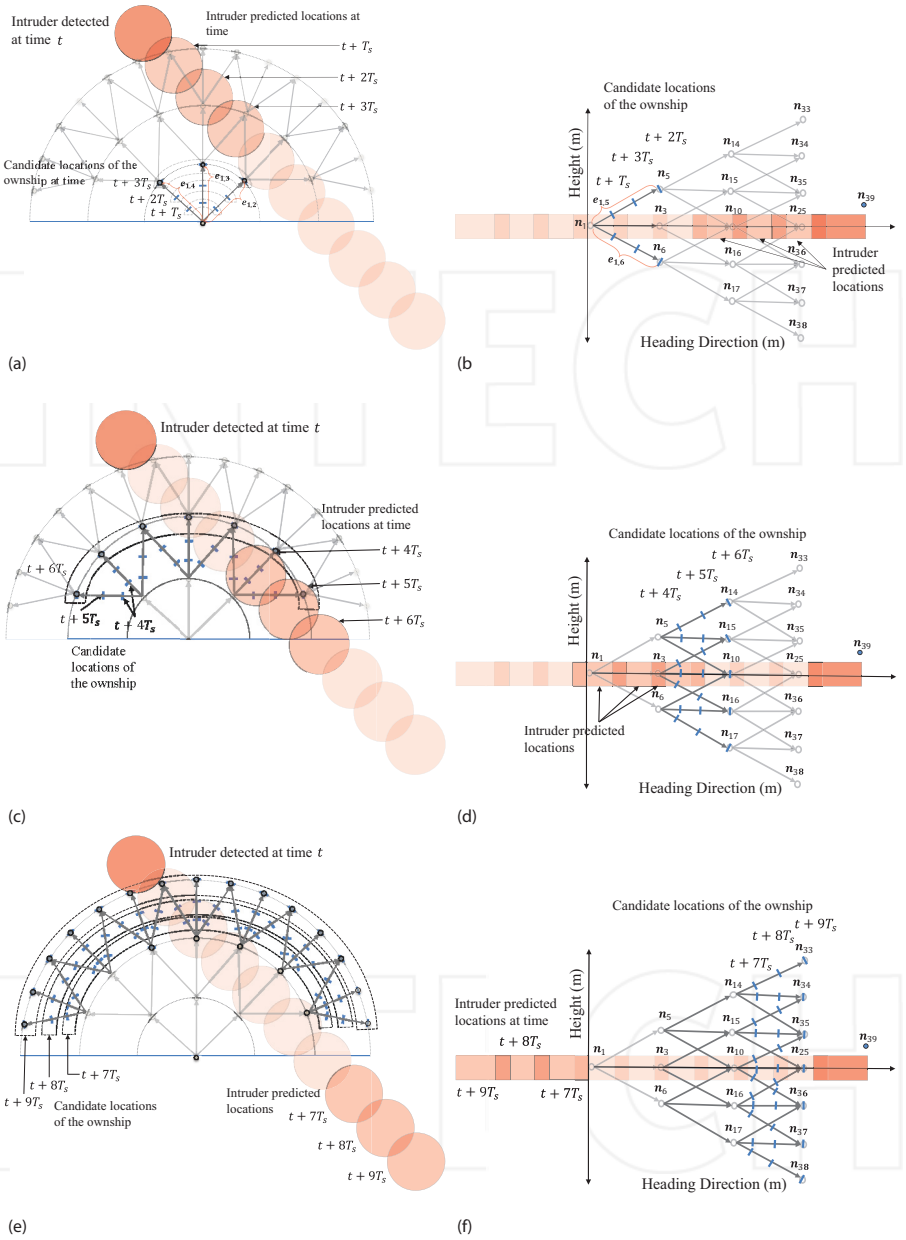

Figure 7. Example illustrating the steps to compute the collision risk. In this example, we have $k = 3$ and $m = 3$. (a) Top view: predicted locations of intruder (less transparent circles), and candidate locations of ownship; (b) side view: predicted locations of intruder (less transparent rectangles), and candidate locations of ownship; (c) predicted locations of intruder and candidate locations of ownship over time window ($t + 4Ts$, $t + 6Ts$); (d) time window ($t + 4Ts$, $t + 6Ts$); (e) time window ($t + 7Ts$, $t + 9Ts$); (f) time window ($t + 7Ts$, $t + 9Ts$).

Another objective of a path planning algorithm is to minimize the deviation from the original path, that is, the path the ownship was following before it detected a collision. Generally, the path is defined as an ordered sequence of waypoints $\mathcal{W} = \mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_f$, where $\mathbf{w}_i = (w_{n,i}, w_{e,i}, w_{d,i})^T \in \mathbb{R}^3$ is the north-east-down location of the ith waypoint in a globally known NED reference frame. The transformation from the global frame to the local-level frame is given by

$$\mathbf{w}_i^b = \mathcal{R}_g^b(\psi_o)\mathbf{w}_i, \tag{16}$$

where

$$\mathbf{R}_g^b(\psi_o) = \begin{pmatrix} \cos\psi_o & \sin\psi_o & 0 \\ -\sin\psi_o & \cos\psi_o & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where ψ_o is the heading angle of the ownship. Let $\mathbf{w}_s$ be the location waypoint of the ownship at the current time instant t and $\mathbf{w}_f \in \mathcal{W}$ be the next waypoint the ownship is required to follow. Assuming a straight-line segment between the waypoints $\mathbf{w}_s$ and $\mathbf{w}_f$, then any point on this segment can be described as $\mathcal{L}(\varrho) = (1-\varrho)\mathbf{w}_s + \boldsymbol{\rho} w_f$ where $\varrho \in [0,1]$, and the minimum distance between an arbitrary node $\mathbf{n}_i$ in $\mathcal{G}$ can be expressed by [36]

$$D(\mathbf{w}_s, \mathbf{w}_f, n_i) \triangleq \begin{cases} D(\varrho^*), & \text{if } \varrho^* \in [0,1], \\ \|\mathbf{n}_i - \mathbf{w}_s\|, & \text{if } \varrho^* < 0, \\ \|\mathbf{n}_i - \mathbf{w}_f\|, & \text{if } \varrho^* > 1, \end{cases} \tag{17}$$

where

$$D(\varrho^*) = \sqrt{\|\mathbf{n}_i - \mathbf{w}_s\|^2 - \frac{\left((\mathbf{w}_s - \mathbf{n}_i)^T(\mathbf{w}_s - \mathbf{w}_f)\right)^2}{\|\mathbf{w}_s - \mathbf{w}_f\|^2}},$$

and

$$\varrho^* = \frac{(\mathbf{w}_s - \mathbf{n}_i)^T(\mathbf{w}_s - \mathbf{w}_f)}{\|\mathbf{w}_s - \mathbf{w}_f\|^2}.$$

Then, the cost that penalizes the deviation of an edge in $\mathcal{G}$ from the nominal path is given by

$$C_{dev}(e_{i,i+1}) = D(\mathbf{w}_s, \mathbf{w}_f, n_i). \tag{18}$$

If small UASs are to be integrated seamlessly alongside manned aircraft, they may require to follow right-of-way rules. Therefore, an additional cost can be also added to penalize edges that violate right-of-way rules. In addition, this cost can be used to favor edges in the horizontal

plane over those in the vertical plane. Since the positive direction of the y-axis in the local-level frame is the right-wing direction, it is convenient to define right and left maneuvers as the positive and the negative directions along the right-wing axis, respectively. Let $\vec{\mathbf{e}}_i \triangleq \mathbf{n}_{i+1} - \mathbf{n}_i$ be the direction vector associated with the edge $e_{i,i+1}$ in $\mathcal{G}$, where $\mathbf{n}_i \triangleq \left(x_i, y_i, z_i\right)^T \in \mathbb{R}^3$ is the location of ith node in the local-level reference frame. Let the direction vector $\vec{\mathbf{e}}_i$ be expressed as $\vec{\mathbf{e}}_i = \left(e_{i_x}, e_{i_y} e_{i_z}\right)^T \in \mathbb{R}^3$. We define $\mathbf{E} \triangleq (e_{i_x}, L, R, e_{i_z})^T \in \mathbb{R}^4$, where e_{i_x} and e_{i_z} are the x and the z components of $\vec{\mathbf{e}}_i$. The y-component of $\vec{\mathbf{e}}_i$ is decomposed into two components: left L and right R, that are defined by

$$L, R \triangleq \begin{cases} L = e_{i_y}, R = 0 & \text{if } e_{i_y} \leq 0, \\ L = 0, R = e_{i_y} & \text{if } e_{i_y} > 0. \end{cases} \tag{19}$$

If we define the maneuvering design matrix to be $\mathbf{J} = \text{diag}([0, c_L, c_R, c_z])$, then the maneuvering cost associated with each edge is given by

$$C_m(e_{i,i+1}) = \sqrt{\mathbf{E}^T \mathbf{J} \mathbf{E}}, \tag{20}$$

The costs c_L and c_R allow the designer to place more or less cost on the left or right edges. Similarly, c_z allows the designer to penalize vertical maneuvers. Multiple values of these cost parameters may be saved in a look-up table, and the collision avoidance algorithm choses the appropriate value based on the geometry of the encounter.

The overall cost for traveling along an edge comes from the weighted sum of all costs given as [35]

$$C(e_{i,i+1}) = C_L(e_{i,i+1}) + C_{col}(e_{i,i+1}) + k_1 C_{th}(e_{i,i+1}) + k_2 C_{dev}(e_{i,i+1}) + k_3 C_m(e_{i,i+1}), \tag{21}$$

where k_1, k_2, and k_3 are positive design parameters that allow the designer to place weight on collision risk or deviation from path or maneuvering preferences depending on the encounter scenario. Once the cost is assigned to each edge in $\mathcal{G}$, then a graph-search method can be used to find the least cost path from a predefined start point to the destination point. In this work, we have used Dijkstra's algorithm.

Dijkstra's algorithm solves the problem of shortest path in a directed graph in polynomial time given that there are not any negative weights assigned to the edges. The main idea in Dijkstra's algorithm is to generate the nodes in the order of increasing value of the cost to reach them. It starts by assigning some initial values for the distances from the start node and to every other node in the graph. It operates in steps, where at each step, the algorithm updates the cost values of the edges. At each step, the least cost from one node to another node is determined and saved such that all nodes that can be reached from the start node are labeled with cost from the start node. The algorithm stops either when the node set is empty or when every node is examined exactly once. A naive implementation of Dijkstra's algorithm runs in a total time complexity of $O\left(|\mathcal{N}|^2\right)$. However, with suitable data structure implementation, the overall time complexity can be reduced to $O(|\mathcal{E}| + |\mathcal{N}| \log_2 |\mathcal{N}|)$ [23, 35].

http://dx.doi.org/10.5772/intechopen.71895

The local-level path planning algorithm generates an ordered sequence of waypoints $\mathcal{W}_c = \mathbf{w}_{c1}, \mathbf{w}_{c2}, \cdots, \mathbf{w}_{ci}$. Then, these waypoints are transformed from the relative reference frame to the global coordinate frame and added to the original waypoints path $\mathcal{W}$. When the ownship is avoiding a potential collision, the avoidance waypoints overwrite some or all of the original waypoints. Next, a path manager is required to follow the waypoints path and a smoother to make the generated path flyable by the ownship. One possible approach to follow waypoints path is to transit when the ownship enters a ball around the waypoint $\mathbf{W}_i$ or a better strategy is to use the half-plane switching criteria that is not sensitive to tracking error [36]. Flyable or smoothed transition between the waypoints can be achieved by implementing the fillet maneuver or using Dubins paths. For further analysis on these topics, we refer the interested reader to Ref. [36].

3. Simulation results

To demonstrate the performance of the proposed path planning algorithm, we simulate an encounter scenario similar to the planner geometry shown in **Figure 8**. The aircraft dynamics are simulated using a simplified model that captures the flight characteristics of an autopilot-controlled UAS. The kinematic guidance model that we considered assumes that the autopilot controls airspeed, v_a, altitude, h, and heading angle, ψ. Under zero-wind conditions, the corresponding equations of motion are given by

$$\dot{p}_n = v_a \cos\psi, \tag{22}$$

$$\dot{p}_e = v_a \sin\psi, \tag{23}$$

$$\dot{\psi} = \frac{g}{v_a}\phi, \tag{24}$$

$$\dot{v}_a = b_v\left(v_a^c - v_a\right) \tag{25}$$

$$\dot{\phi} = b_\phi\left(\phi^c - \phi\right) \tag{26}$$

$$\ddot{h} = b_{\dot{h}}\left(\dot{h}^c - \dot{h}\right) + b_h(h^c - h), \tag{27}$$

where p_n, p_e are the north-east position of the aircraft. The inputs are the commanded altitude, h^c, the commanded airspeed, v_a^c, and the commanded roll angel, ϕ^c. The parameters b_v, b_ϕ, b_h, and $b\dot{h}$ are positive constants that depend on the implementation of the autopilot and the state estimation scheme. For further analysis on the kinematic and dynamic guidance models for UAS, we refer the interested reader to [36]. In the following simulation, the ownship starts at $(0, 0, -200)^T$ in the NED coordinate system, with an initial heading of 0 deg. measured from north and follows a straight-line path at a constant speed of 22 m/s to reach the next waypoint located at $(1500, 0, -200)^T$. The encounter geometry includes three intruders flying at different altitudes: the first is approaching head-on, the second is converging from the right, and the third is overtaking from the left. We chose the intruders's speed similar to the known cruise speed of ScanEagle UAS, Cessna SkyHawk 172R, and Raven RQ-11B UAS. The speed of the

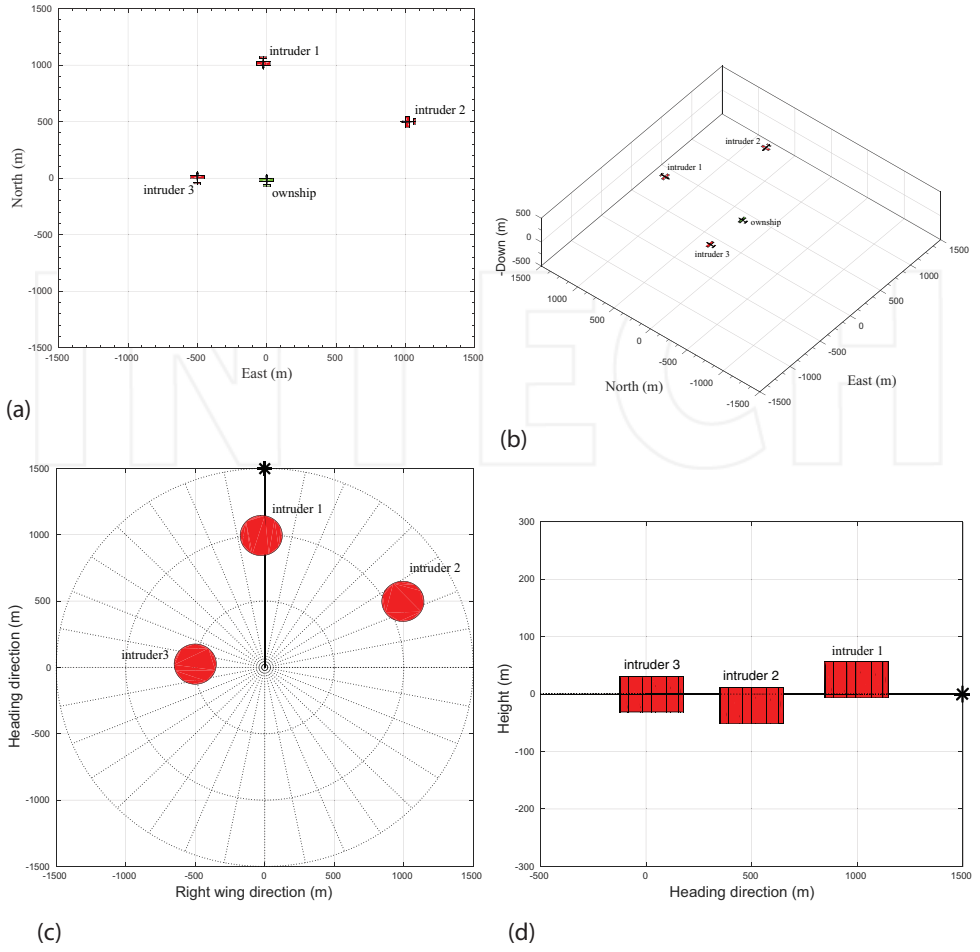

Figure 8. Encounter geometry for the ownship and three intruders at t = 0.1 s. (a) Overhead view of initial locations of aircraft; (b) 3D view of initial locations of aircraft; (c) overhead view of reference frame; (d) side view of relative reference frame.

intruders is 41, 65, and 22 m/s, respectively. In addition, the intruders are assumed to fly at a constant speed the entire simulation period. As shown in **Figure 8**, the initial locations of intruders in the NED coordinate system are $(-25, 1000, -225)^T$, $(500, 1000, -180)^T$, and $(25, -500, -200)^T$, respectively, with initial heading of 180, -90, and $0°$, respectively.

In the following simulation, our choice of the collision volume is a cylinder of radius d_s = 152.4 m (500 ft) and height h_s = 61 m (200 ft) centered on each of the intruders. A collision incident occurs when the horizontal relative range and altitude to the ownship are simultaneously below horizontal and vertical minimum safe distances d_s and $h_s/2$. We assume that there exists a sensor and tracking system that provides the states of the detected intruders.

However, not every aircraft that is observed by the sensing system presents a collision threat. Therefore, we implemented a geometric-based collision detection algorithm to determine whether an approaching intruder aircraft is on a collision course. The collision detection approach is beyond the scope of this work, and we refer the interested reader to [37].

At the beginning of simulation, the predicted relative range and altitude at the closest point of approach (CPA) are shown in **Table 1**. Imminent collisions are expected to occur with the first and second intruders as their relative range and altitude with respect to the ownship are below the defined horizontal and vertical safe distances. The time remaining to the closest point of approach t_{CPA} with respect to the first and second intruders is 15.77 and 16.56 s, respectively. The scenario requires that the ownship plans and executes an avoidance maneuver well before the t_{CPA}. This example demonstrates the need for an efficient and computationally fast avoidance planning algorithm. **Table 2** shows the total time required to run the avoidance algorithm, and the maximum and average time required to execute one cycle. The results show that the proposed algorithm takes a significantly reduced time in computation with an average and maximum time to execute one cycle of the code of 20 ms and 0.1326 s, respectively, and a total time of 0.3703 s to resolve the collision conflict.

Figure 9 shows the planned avoidance path by the ownship. These results show that the avoidance path safely maneuvers the ownship without any collisions with the intruders. In addition, the ownship should plan an avoidance maneuver that does not lead to a collision with intruders that were not on a collision course initially such as the case with the third intruder. Initially, the third intruder and the ownship are flying on near parallel courses. The relative range and altitude at CPA with respect to the third intruder are 437.14 and 4361.07 m, respectively, and the time remaining to the CPA is 1982.25 s. Obviously, both aircrafts are not on a collision course. However, the third intruder is descending and changing its heading toward the ownship. The path planner, however, accounts for predicted locations of the detected intruder over the look-ahead time window, allowing the ownship to maintain a safe distance from the third intruder. This example demonstrates that the proposed path planner can handle unanticipated maneuvering intruders. Once collisions are resolved the path planner returns the ownship to the next waypoint of its initial path.

Intruder	$\lVert \mathbf{p}_r(t_{\mathrm{CPA}}) \rVert$ (m)	$\lvert p_{r_z}(t_{\mathrm{CPA}}) \rvert$ (m)	t_{CPA} (s)
1	24.90	25	15.77
2	141.33	20	16.56
3	437.14	4361.07	1982.25

Table 1. Relative range and altitude, and the time remaining to the closest point of approach.

Total run time (s)	Max. run time (one cycle) (s)	Average run time (one cycle) (s)
0.3703	0.1326	0.0206

Table 2. Collision avoidance algorithm run time.

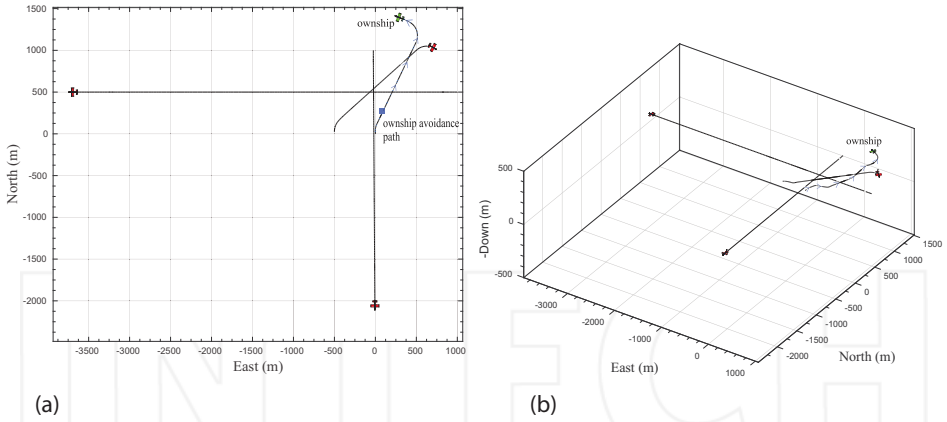

Figure 9. Avoidance path followed by the ownship and path tracks of the intruders at t = 75 s. (a) Overhead view of avoidance path and (b) 3D view of of avoidance path.

The relative range between the ownship and the intruders is shown in **Figure 10**. The results show that no collisions have occurred, and that the ownship successfully planned an avoidance maneuver. The avoidance planner ensures that when the relative horizontal range is less than d_s, the relative altitude is greater than $h_s/2$. For example, as shown in **Figure 10b**, the relative range to the first intruder over time interval [16.2, 18] s is below d_s. However, over the same time interval, the relative altitude is above $h_s/2$.

Another important aspect to evaluate the performance of the proposed algorithm is its ability to reduce the length of the avoidance path while avoiding the intruders. This is important because it reduces the amount of deviation from the original path and ultimately the flight time, which is of critical importance for the small UAS with limited power resources. **Table 3** shows that the length of the avoidance paths is fairly acceptable compared to the initial path length.

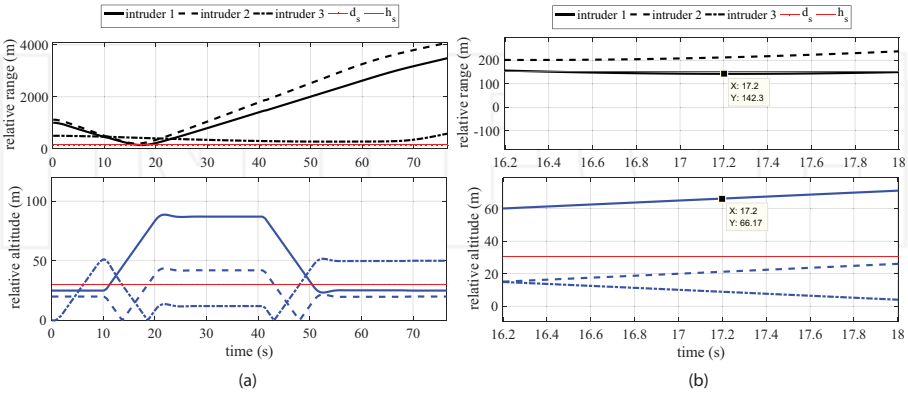

Figure 10. Relative horizontal range and altitude between the ownship and intruders. (a) Horizontal range and relative altitude to intruders and (b) a close up view of Figure 10a.

Scenario number	Initial path length (m)	Avoidance path length (m)
1	1500	1955

Table 3. Length of the avoidance path.

4. Conclusions

In this chapter, we have presented a path planning approach suitable for small UAS. We have developed a collision avoidance logic using an ownship-centered coordinate system. The technique builds a maneuver graph in the local-level frame and use Dijkstra's algorithm to find the path with the least cost.

A key feature of the proposed approach is that the future motion of the ownship is constrained to follow nodes on the map that are spaced by a constant time. Since the path is represented using waypoints that are at fixed time instants, it is easy to determine roughly where the ownship will be at any given time. This timing information is used when assigning cost to edges to better plan paths and prevent collisions.

An advantage of this approach is that collision avoidance is inherently a local phenomenon and can be more naturally represented in local coordinates than global coordinates. In addition, the algorithm accounts for multiple intruders and unanticipated maneuvering in various encounter scenarios. The proposed algorithm runs in near real time in Matlab. Considering the small runtime shown in the simulation results, we expect that implementing these algorithms in a compiled language, such as C or C++, will show that real-time execution is feasible using hardware. That makes the proposed approach a tractable solution in particular for small UAS.

An important step forward to move toward a deployable UAS is to test and evaluate the performance of the close-loop of sensor, tracker, collision detection, path planning, and collision avoidance. Practically, the deployment of any UAS requires a lengthy and comprehensive development process followed by a rigorous certification process and further analysis including using higher fidelity models of encounter airspace, representative number of simulations, and hardware-in-the-loop simulation. Unlike existing collision manned aviation collision detection and avoidance systems, an encounter model cannot be constructed solely from observed data, as UASs are not yet integrated in the airspace system and good data do not exist. An interesting research problem would be to design encounter models similar to those developed to support the evaluation and certification of manned aviation traffic alert and collision avoidance system (TCAS).

Acknowledgements

This research was supported by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation-sponsored industry/university cooperative research center (I/UCRC) under NSF Award No. IIP-1161036 along with significant contributions from C-UAS industry members.

Author details

Laith R. Sahawneh[1]* and Randal W. Beard[2]

*Address all correspondence to: lsahawneh@ufl.edu

1 Department of Mechanical and Aerospace Engineering, University of Florida, Florida, USA

2 Department of Electrical and Computer Engineering, Brigham Young University, Utah, USA

References

[1] George S. FAA Workshop on Sense and Avoid (SAA) for Unmanned Aircraft Systems (UAS). 2009

[2] Hottman SB, Hansen KR, Berry M. Literature review on detect, sense, and avoid technology for Unmanned Aircraft Systems. In: Technical Report. 2009

[3] Federal Aviation Administration. Subchapter F-Air Traffic and General Operating Rules. 2015

[4] Kuchar JK, Yang LC. A review of conflict detection and resolution modeling methods. IEEE Transactions on Intelligent Transportation Systems. Dec. 2000;**1**(4):179-189

[5] Albaker BM, Rahim NA. A survey of collision avoidance approaches for unmanned aerial vehicles. International Conference for Technical Postgraduates (TECHPOS). 2009:1-7

[6] Hyunjin YK. Reactive collision avoidance of unmanned aerial vehicles using a single vision sensor. AIAA Guidance, Control, and Dynamics. 2013;**36**(4):1234-1240

[7] Rajnikant S, Saunders JB, Randal Beard W. Reactive path planning for micro air vehicles using bearing-only measurements. International Robotic Systems. 2012;**65**(1–4):409-416

[8] White BA, Antonios HS. UAV obstacle avoidance using differential geometry concepts. In: 18th IFAC World Congress; Milano, Italy; 2011. Vol. 3. pp. 6325-6330

[9] Saunders J, Beard RW. Vision-based reactive multiple obstacle avoidance for micro air vehicles. In: IEEE American Control Conference ACC'09; St. Louis, MO, June 10–12; 2009, pp. 5253-5258

[10] George J, Ghose D. A reactive inverse PN algorithm for collision avoidance among multiple unmanned aerial vehicles. In: American Control Conference; St. Louis, MO; June 10–12; IEEE. pp. 3890-3895

[11] Bilimoria KD. A geometric optimization approach to aircraft conflict resolution. In: Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit; 2010

[12] Fiorini P, Shiller Z. Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research. 1998;**17**(7):760-772

[13] Chakravarthy A, Ghose D. Obstical avoidance in a dynamic environment: A collision cone approach. IEEE Transactions on System, Man and Cybernitics, Part A: Systems and Humans. 1998;**28**(5):562-572

[14] Lam TM, Mulder M, Van Paassen M, Mulder JA, Van Der FC. Force-stiffness feedback in uninhabited aerial vehicle teleoperation with time delay. AIAA Guidance, Control, and Dynamics. 2009;**32**(3):821-835

[15] Sahawneh LR, Beard RW, Avadhanamz S, He B. Chain-based collision avoidance for UAS sense and avoid systems. In: AIAA Guidance, Navigation, and Control (GNC) Conference; Boston, MA; 2013

[16] Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation. 1996;**12**(4):566-580

[17] LaValle SM. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98–11. Computer Science Department, Iowa State University; October 1998

[18] Dijkstra EW. A note on two problems in connection with graphs. Numerische Mathematik. 1959;**1**:269-271

[19] Dechter R, Pearl J. Generalized best-first search strategies and the optimality of a*. Journal of the ACM (JACM). 1985;**32**(3):505-536

[20] Mirolo C, Pagello E. A cell decomposition approach to motion planning based on collision detection. In: Proceedings of the 1995 International Conference on Advanced Robotics. 1995. pp. 481-488

[21] Angelov P. Sense and Avoid in UAS: Research and Applications. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Ltd; 2012

[22] Koren Y, Borenstein J. Potential field methods and their inherent limitations for mobile robot navigation. In IEEE International Conference On Robotics And Automation; IEEE. 1991;**2**:1398-1404

[23] LaValle SM. Planning Algorithms. Cambridge University Press; 2006

[24] Bortoff SA. Path planning for UAVs. In: Proceedings of the American Control Conference. Chicago, Illinois; June 2000. pp. 364-368

[25] McLain TW, Beard RW. Trajectory planning for coordinated rendezvous of unmanned air vehicles. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference. AIAA Reston, VA; 2000. Vol. 4369. pp. 1-8

[26] Argyle ME, Chamberlain C, Beard RW. Chain-based path planning for multiple UAVs. In: 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA. Dec. 2011

[27] Luders BD, Karaman S, How JP. Robust sampling-based motion planning with asymptotic optimality guarantees. In: AIAA Guidance, Navigation, and Control Conference (GNC), Boston, MA. 2013

[28] Luders BD, Karaman S, Frazzoli E, How JP. Bounds on tracking error using closed-loop rapidly-exploring random trees. In: American Control Conference (ACC). IEEE; 2010. pp. 5406-5412

[29] Luders B, Karaman S, How JP. Robust sampling-based motion planning with asymptotic optimality guarantees. In Guidance, Navigation, and Control (GNC) Conference, Boston, MA. 2013. AIAA

[30] Kothari M, Postlethwaite I. A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. International Robotic Systems. 2013;**71**:231-253

[31] Standard Specification for Design and Performance of an Airborne Sense-and-Avoid System. Tech. Rep. TR F2411-07. West Conshohocken, PA: ASTM International; 2007

[32] US Department of Transportation and Federal Aviation Adminstration. Aeronautical Information Manual Official Guide to Basic Flight Information and ATC Procedures

[33] Lee SM, Park C, Johnson MA, Mueller ER. Investigating effects of well clear definitions on UAS sense-and-avoid operations. In: Aviation Technology, Integration, and Operations Conference, Los Angeles, CA. AIAA. 2013

[34] Consiglio M, Chamberlain J, Munoz C, and Hoffler K. Concept of integration for UAS operations in the NAS. In: 28th International Congress of the Aeronautical Sciences (ICAS); Brisbane, Australia; 2012

[35] Sahawneh LR, Airborne Collision Detection and Avoidance for Small UAS Sense and Avoid Systems [PhD Thesis] Brigham Young University; 2016

[36] Beard RW, McLain TW. Small Unmanned Aircraft: Theory and Practice. New Jersey, USA: Princeton University Press; 2012

[37] Sahawneh LR, Argyle ME, Beard RW. 3D path planning for small UAS operating in low-altitude airspace. In International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2016. pp. 413-419