Autonomous navigation for multiple unmanned aerial vehicles (UAVs) in urban environments

Merrick Stuart Macqueen Hughes[1*], and *Jacobus Adriaan Albertus* Engelbrecht[1]

[1]Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa

Abstract. This paper presents an autonomous navigation system for multiple unmanned aerial vehicles (UAVs) in urban environments. The navigation system comprises a long-term route planner and a short-term cooperative collision avoidance function. A 3D model of a real urban terrain is used to represent the environment within which the UAVs operate, and the system takes the motion constraints of the UAVs into account when performing planning. The navigation system enables multiple independent UAVs to travel from their departure points to their destinations while predicting and avoiding collisions with one another and with static terrain. Monte Carlo simulations on the system indicate a worst-case route planner replan rate of 11.65 %, route planner failure rate of 9.82×10^{-4} %, and collision resolution failure rate of 0.12 % for the collision avoidance function.

1 Introduction

This section begins by providing background for autonomous navigation of unmanned aerial vehicles (UAVs). This is followed by a brief literature review of terrain modelling methods, UAV path planning, and collision avoidance for both UAVs and manned aircraft.

1.1 Background

Several exciting applications are emerging for UAVs in urban environments including delivery services, monitoring and surveillance, disaster management, wireless coverage, infrastructure inspection, and road traffic monitoring [1-2]. Basic implementations of these applications can be achieved using UAVs as remotely piloted aircraft systems (RPASs). However, having UAVs be remotely piloted severely limits the potential of these applications seeing as each UAV would typically require a dedicated human operator for large portions of its mission. Furthermore, the capabilities of each UAV would be limited by that of their human operator. For example, the ability of a UAV to avoid collisions is completely dependent on the speed and accuracy of its operator's reactions.

To maximise the potential of emerging UAV applications, UAVs will be required to perform autonomous navigation through their environment while avoiding collisions with

* Corresponding author: 21712662@sun.ac.za

other UAVs and static obstacles. Urban environments pose a formidable challenge for UAV autonomous navigation due to their complex arrangement of obstacles and confined spaces. This makes it difficult for a UAV to initially determine a flyable path between its start and goal location. Furthermore, UAVs will sometimes be required to replan their intended trajectories if an upcoming collision is detected, with dense urban terrain often restricting the number of possible alternative trajectories.

Consequently, this paper proposes an autonomous navigation system for multiple UAVs in urban environments capable of planning long-term flyable routes that UAVs can follow to reach their destination, as well as performing short-term cooperative collision avoidance to prevent upcoming collisions with other UAVs and terrain.

1.2 Literature review

Autonomous navigation of UAVs in urban environments is an extensively researched problem. As a result, literature contains several different approaches that have been used to model urban terrain for use in simulation. Overwhelmingly, 3D urban terrain in literature is modelled by scattering simple, randomly generated 3D objects (often rectangular prisms or cylinders) of varying sizes [3-7]. This is an appealing way to model urban terrain seeing as these objects are easy to generate and perform collision queries against, require little memory, and provide a fairly accurate approximation of the geometry of real urban terrain. It is also an especially attractive option when autonomous UAV navigation is not the primary focus of the research [7] seeing as detailed or irregular terrain can often inhibit other processes. Like that of 3D urban terrain, literature involving 2D urban terrain often uses regular polygons (such as rectangles [8-9]), circles [6], as well as maze-like obstacle regions [10] to represent 2D urban terrain. Occasionally, 2D urban terrain is modelled by creating polygon obstacles from the outlines of building footprints which are extracted from a satellite image of an urban area [10-11]. Only rarely are accurate 3D terrain models of real urban environments used in UAV autonomous navigation systems [3], and often the research in which these models are used is not focused primarily on autonomous navigation [12]. In manned commercial aircraft, the *Enhanced Ground Proximity Warning System* (EGPWS) uses a digital elevation model of surrounding terrain to warn pilots of imminent collisions with terrain, water, or tall structures [13].

UAV autonomous navigation systems often make use of path planning techniques to determine a collision-free path between a start and goal location. Commonly used path planning techniques in such systems include sampling-based methods, node-based optimal methods, and mathematical model methods [4, 14]. Sampling-based methods such as the Rapidly-exploring Random Tree (RRT) are popular for UAV path planning in urban environments due to their computational efficiency [5], especially in higher dimensions when it becomes extremely difficult to define obstacle regions mathematically. Sampling-based methods typically explore the environment using randomly sampled points, meaning that their resulting paths approach optimality as time tends to infinity and more of the environment is explored. However, in practice an iteration limit must be enforced due to computational constraints which leads to suboptimal paths [5] as exploration of the environment is prematurely terminated. Another technique that has been used for UAV path planning in urban environments is the artificial potential field (APF) method due to its simplicity and smooth paths [15]. Unfortunately, a well-known drawback of the APF method is that UAVs are occasionally unable to reach goal locations that are behind or too close to an obstacle, and so it is often necessary to introduce additional measures to resolve these "inaccessible goal issues" [15].

Collision avoidance between UAVs is another common feature of many UAV autonomous navigation systems in literature. UAV collision avoidance techniques include rules-based methods, path planning methods, model predictive control (MPC), and rewards-based methods [4, 16]. In urban environments, the artificial potential field (APF) method has also been used for inter-UAV collision avoidance [15] which is less likely to cause inaccessible goal scenarios seeing as other UAVs represent dynamic obstacles. Collision avoidance strategies in UAV systems are broadly classified as being either cooperative or non-cooperative. In cooperative strategies, UAVs avoid collisions while considering the intended trajectories of other UAVs, whereas UAVs in non-cooperative strategies only consider their own intended trajectories and simply treat other UAVs as dynamic obstacles to avoid. Cooperative strategies can be further classified as being either *centralised*, where UAVs plan simultaneously and consider every possible combination of actions between them, or *decoupled*, where UAVs plan sequentially according to a priority order [4]. In manned commercial aircraft, the *Traffic Alert and Collision Avoidance System* (TCAS) is a rules-based, decoupled collision avoidance system that alerts pilots of intruder aircraft in their surrounding airspace and provides resolution advisories (RAs) to help pilots maintain sufficient separation from the intruder aircraft [17].

1.3 Research contributions

The literature review indicates that although UAV path planning and collision avoidance are well-studied problems, they are seldom verified in simulations that feature accurate urban terrain models. By developing an autonomous navigation system for multiple UAVs that uses a 3D model of real urban terrain, this research aims to contribute the following:

- An optimised technique for detecting UAV collisions with a 3D polygon mesh terrain model.
- A method for planning long-routes through urban terrain that considers only those obstacles that a UAV is likely to encounter during its mission.
- A cooperative collision avoidance function that replans UAV trajectories while taking their motion constraints into account.

2 Methodology

This section explains the methodology used to develop the UAV autonomous navigation system. The section begins with an overview of the proposed navigation system. Thereafter, a description of both the terrain model and the UAV model that was chosen for the system is provided. This is followed by an explanation of the system's route planner and collision avoidance function.

2.1 Overview

The proposed autonomous navigation system comprises a long-term route planner and a short-term cooperative collision avoidance function. As illustrated in Figure 1, the route planner first generates a long-term route for each UAV's specified mission which they then attempt to follow while cooperatively predicting and resolving impending collisions with terrain and other UAVs.

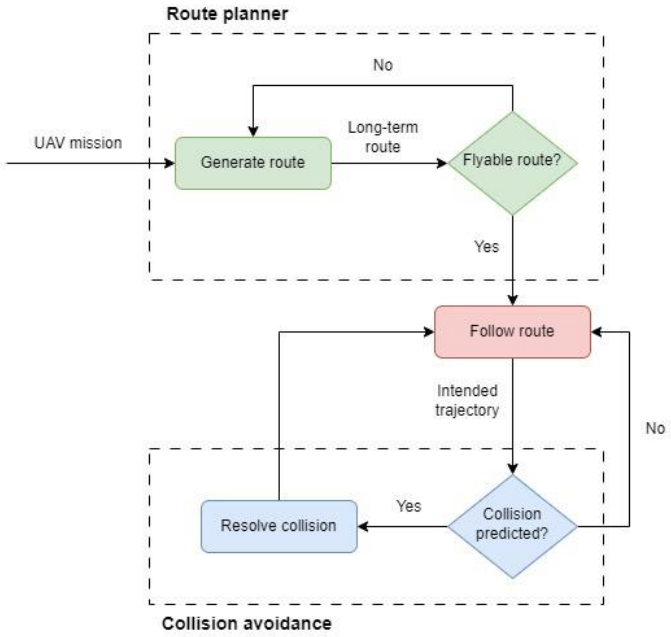

Fig. 1. Overview of the UAV autonomous navigation system.

2.2 Terrain model

To be representative of a real urban environment, a 3D model of Manhattan, New York City (NYC), was chosen as the system's terrain model. The model was created by the NYC Department of City Planning and made publicly available as a 3DM CAD (computer-aided design) file [18] - a file format intended for viewing/editing the 3D model in dedicated CAD software [19]. To use the model as terrain in the navigation system, it was converted into the PLY file format. A PLY file is a polygon mesh approximation of a 3D object that explicitly defines the geometry of the object using a list of vertices and faces [20]. This enables UAVs to perform efficient collision detection against the terrain model and allows for easier route planning through the terrain given that the geometry of the model is now accessible.

The simplest implementation of a PLY file is one in which triangles are the only type of polygon used in the polygon mesh approximation of the 3D object, resulting in a triangle mesh. For this reason, a triangle mesh implementation of a PLY file was used for the terrain model in the navigation system and is illustrated in Figure 2.

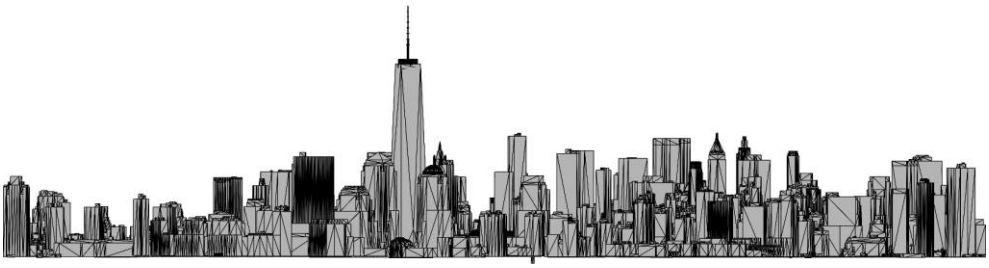

Fig. 2. Triangle mesh PLY file of the terrain model.

2.2.1 Terrain model preprocessing

Converting objects from a CAD file format that contains intricate curved surfaces into a polygon mesh approximation (a process known as *tessellation*) often leads to various geometric anomalies in the resulting mesh [21]. These anomalies cause the mesh to become poorly defined and include duplicate vertices and faces, degenerate faces, and unintended gaps between faces (Figure 3) which all reduce the "geometrical robustness" of the mesh [22].

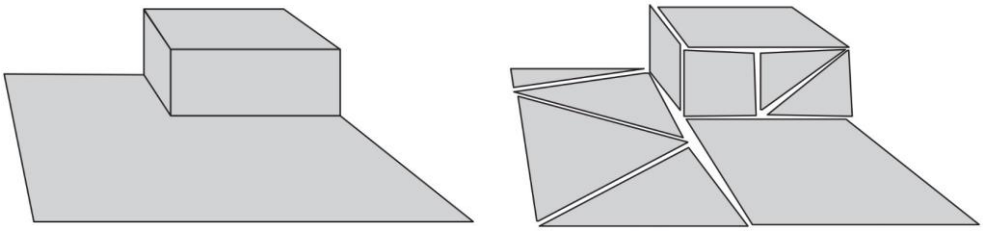

Fig. 3. A 3D object (left) that was tessellated into a poorly defined polygon mesh (right) with unintended gaps between faces (adapted from [22]).

Geometric anomalies can be problematic for the navigation system if they are present in the terrain model's triangle mesh. Duplicate vertices and faces occupy unnecessary memory and, in the case of duplicate faces, slow down collision detection against the terrain model since redundant collision checks are performed against these duplicate faces. More importantly, duplicate vertices prevent adjacency information from being maintained [22] between triangles in the mesh. This is because a shared vertex between neighbouring triangles that is erroneously duplicated will give the impression that the triangles are not joined seeing as each is connected to different version of the shared vertex. Unreliable adjacency information hinders the process of identifying groups of connected triangles, which is required for efficient collision detection.

Degenerate faces in a polygon mesh include faces that are nonconvex, self-intersecting, or those that have zero area [22]. Recalling that the polygon mesh of the terrain model contains only triangles, and it is not possible for a triangle to be nonconvex or self-intersecting, zero-area faces are the only type of degenerate face that is of concern in the terrain model. Zero-area faces can cause errors when attempting to perform polygon operations on them (such as computing their surface normal vector) considering that a zero-area face is essentially a line for which many polygon operations are not defined.

Lastly, unintended gaps between faces in a polygon mesh can cause unexpected behaviour during collision detection seeing as a line could pass through these gaps allowing it to traverse a seemingly solid object. This is not of particular concern for the terrain model considering that these gaps are typically extremely small (in fact, they are often invisible [22]) and UAVs in the navigation system have a finite size meaning that they would not be able to pass through these gaps. However, unintended gaps prevent adjacency information from being maintained in the mesh in the same way that duplicate vertices do. This is because many of these gaps are caused by vertices at slightly different locations that were meant to be joined. Hence, a process known as *vertex welding* [22] must be performed on the triangle mesh to merge vertices within a certain tolerance (Figure 4) and re-establish adjacency information as well as close many of the unintended gaps.

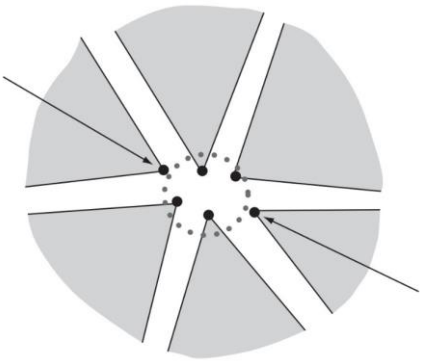

Fig. 4. An illustration of vertex welding showing the welding tolerance within which vertices are merged (adapted from [22]).

To ensure that the triangle mesh of the terrain model is not poorly defined and susceptible to geometrical robustness issues, the above anomalies must be removed in a preprocessing stage using polygon mesh "cleaning" and "repairing" operations [22].

2.2.2 Collision detection

To enable the UAVs to perform collision detection against this vast polygon mesh, it was decided to use well-established computational geometry techniques that were largely developed by the video game industry for collision detection in early video games. As illustrated in Figure 5, a spherical *exclusion zone* is defined around the centre of mass of each UAV to account for its finite size as well as potential uncertainty in its position. A collision with terrain can now be defined as any triangle from the terrain model breaching a UAV's exclusion zone, whereas an inter-UAV collision is defined as an overlap of UAV exclusion zones. The navigation system simulates UAV trajectories in discrete time steps, and so collision detection against the terrain model for a single time step requires determining whether a UAV's exclusion zone intersected any triangle in the terrain model as it moved from its previous position to its new position over the duration of the time step. In computational geometry, this refers to an intersection test between a moving sphere and a triangle [23], as depicted in Figure 6.

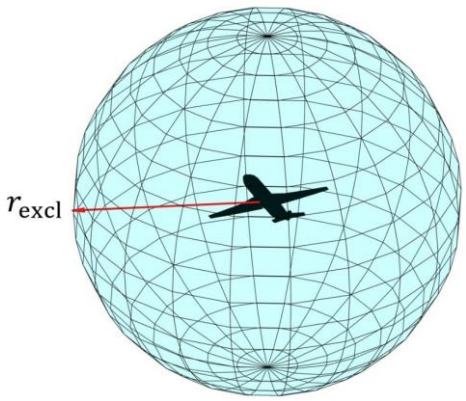

Fig. 5. A UAV's spherical exclusion zone defined by its radius of exclusion r_{excl}.

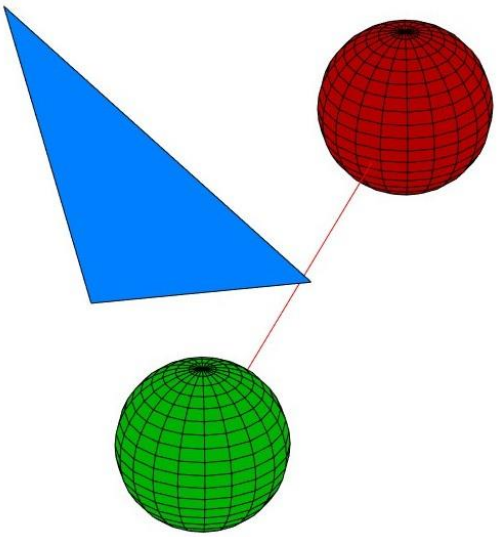

Fig. 6. Intersection test between a moving sphere and a triangle. The green sphere shows the starting position of the moving sphere, and the red sphere shows its end position after a time step.

The triangle mesh of the terrain model comprises approximately 170 000 triangles after preprocessing. A naïve approach to collision detection would be to repeatedly perform the moving sphere-triangle intersection test described above against every triangle in the mesh after every time step. Although this naïve approach would work, it is highly inefficient and would subsequently be extremely slow, considering the size of the triangle mesh. The time complexity (approximate number of intersection tests performed per time step) of collision detection using the naïve approach is $O(n_{\mathrm{tr}})$, where $n_{\mathrm{tr}} \approx 170\ 000$ is the number of triangles in the terrain model.

The time complexity of collision detection can be improved significantly [24] by enclosing groups of connected triangles within *bounding volumes*, as shown in Figure 7. Collision detection now begins by first performing intersection tests against these bounding volumes (which are inexpensive tests [25]). Thereafter, further intersection testing is only performed against those triangles within the few bounding volumes that were intersected. This eliminates the need to test every triangle in the mesh since a triangle cannot be intersected if its bounding volume was not intersected first.

The use of bounding volumes improves the time complexity of collision detection to $O(n_{\mathrm{BV}})$, where n_{BV} is the number of bounding volumes required for the groups of connected triangles in the mesh. Considering that $n_{\mathrm{BV}} \approx 800$ for the terrain model, this is a significant improvement in time complexity compared to that of the naïve approach. This improvement is especially appreciable considering that an intersection test against a bounding volume is much cheaper than against a triangle [25].

Although the use of bounding volumes alone allows for a significant improvement in time complexity, this improvement is only by a constant factor $n_{\mathrm{tr}}/n_{\mathrm{BV}}$ [24]. By arranging the bounding volumes into a *bounding volume hierarchy* (BVH), the time complexity of collision detection can be reduced to logarithmic in n_{BV} [24].

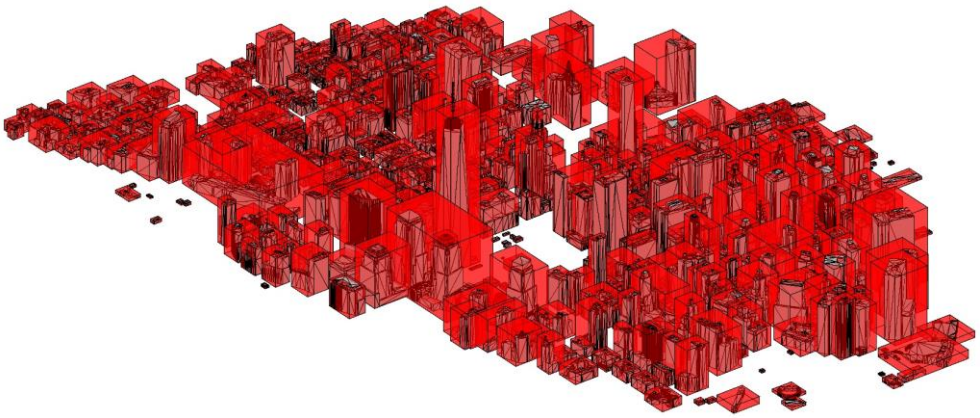

Fig. 7. Terrain model with groups of connected triangles enclosed in bounding volumes.

As illustrated in Figure 8, a BVH is a tree hierarchy that is constructed by recursively merging existing bounding volumes until a single bounding volume encloses all the complex geometry. This single bounding volume becomes the root node of the BVH tree, and collision detection is performed by traversing the tree according to which bounding volumes are intersected at each level. The complex geometry in each leaf node of the tree is only tested for intersection if that leaf node is reached during traversal and its bounding volume is subsequently intersected. Figure 8 also highlights that child nodes in the tree are only tested if their parent node is intersected. The nature of this BVH traversal further improves the time complexity of collision detection to $O(2\log_2(n_{\mathrm{BV}}))$, where $2\log_2(n_{\mathrm{BV}}) = 2\log_2(800) \approx 19$.

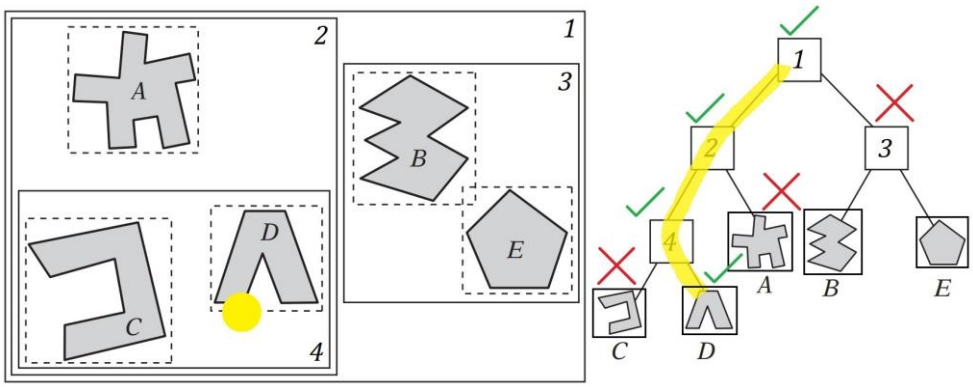

Fig. 8. Example of a bounding volume hierarchy (BVH) traversal during collision detection between the yellow circle and complex geometry A-E (adapted from [24]).

2.3 UAV model

A kinematic model of a point mass was chosen to simulate the motion of the UAVs in the navigation system. The state of each UAV at time t is described by the state vector $\mathbf{x}(t)$ in Equation (1).

$$\mathbf{x}(t) = [x_{\mathrm{I}}(t), y_{\mathrm{I}}(t), z_{\mathrm{I}}(t), \psi(t), \gamma(t), \phi(t), v(t)]^T \quad (1)$$

In Equation (1), x_I, y_I, and z_I are the Cartesian coordinates of the UAV's centre of mass in the inertial axis system, ψ, γ, and ϕ are the heading, flight path angle, and bank angle of the UAV, respectively, and v is the magnitude of the UAV's velocity vector **v** (Figure 9).

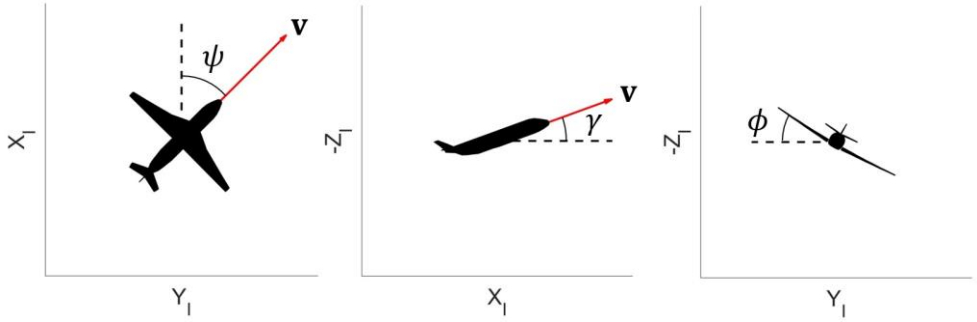

Fig. 9. Visualisation of a UAV's state vector **x**.

The kinematic model uses first-order differential equations to model the response of a UAV's state vector $\mathbf{x}(t)$ to reference commands of flight path angle (γ_{ref}), bank angle (ϕ_{ref}), and velocity (v_{ref}), as shown in Equations (2) − (8).

$$\dot{x}_I(t) = v(t)\cos\gamma(t)\cos\psi(t) \tag{2}$$
$$\dot{y}_I(t) = v(t)\cos\gamma(t)\sin\psi(t) \tag{3}$$
$$\dot{z}_I(t) = -v(t)\sin\gamma(t) \tag{4}$$
$$\dot{\psi}(t) = \frac{g}{v(t)}\tan\phi(t) \tag{5}$$
$$\dot{\gamma}(t) = \frac{1}{\tau_\gamma}\left(\gamma_{\text{ref}}(t) - \gamma(t)\right) \tag{6}$$
$$\dot{\phi}(t) = \frac{1}{\tau_\phi}\left(\phi_{\text{ref}}(t) - \phi(t)\right) \tag{7}$$
$$\dot{v}(t) = \frac{1}{\tau_v}\left(v_{\text{ref}}(t) - v(t)\right) \tag{8}$$

In Equations (6) − (8), τ_γ, τ_ϕ, and τ_v are the time constants associated with the model's response to changes in γ_{ref}, ϕ_{ref}, and v_{ref}, respectively. By assigning values to these time constants from step responses of real-life UAVs, the model can consider the kinematic constraints of each UAV without having to explicitly model their aircraft flight dynamics. Figure 10 shows an example of a trajectory that can be simulated using the UAV model.

2.3.1 Cross-track controller

As mentioned in the overview of the navigation system (Section 2.1), the system's route planner first generates a long-term route for each UAV which they then attempt to follow to reach their goal location. These long-term routes comprise a set of ordered waypoints at the UAV's specified mission altitude which are connected by straight-line segments known as *waypoint legs* (Figure 11). UAVs follow their route by attempting to minimise the perpendicular distance to their current waypoint leg. This distance is called the UAV's *cross-track error* and is denoted by n in Figure 11.

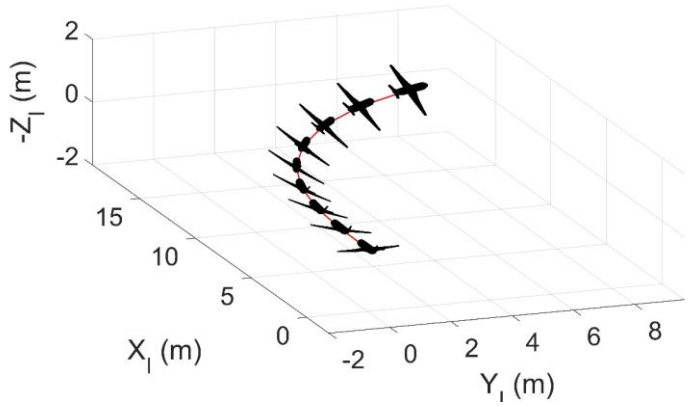

Fig. 10. Example of a trajectory simulated using the UAV model.

To minimise this cross-track error, a *cross-track controller* is used. The cross-track controller calculates a bank angle reference command ϕ_{ref} as a function of the UAV's cross-track error n and the rate of change of cross-track error $\dot{n}$, according to Equation (9).

$$\phi_{\text{ref}} = -K_n n - K_{\dot{n}} \dot{n} \tag{9}$$

In Equation (9), K_n and $K_{\dot{n}}$ are the gains for the cross-track error n and the rate of change of cross-track error $\dot{n}$, respectively. The values for K_n and $K_{\dot{n}}$ are selected to achieve a balance between speed of response and overshoot.

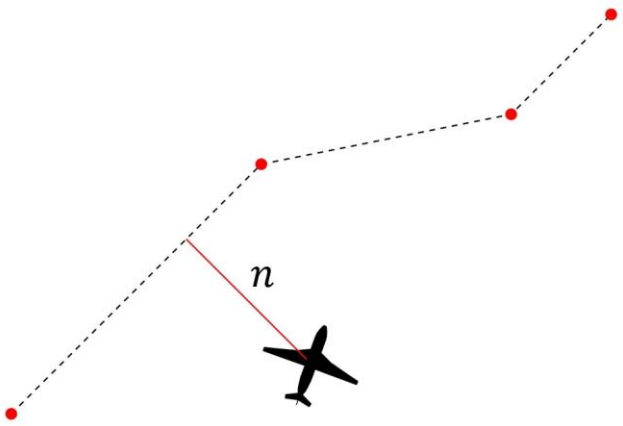

Fig. 11. Example of a long-term route generated by the route planner showing the UAV's cross-track error n.

2.3.2 Altitude controller

An altitude controller is used to keep the UAV flying at its reference altitude, which is typically the UAV's specified mission altitude. The altitude controller is similar to the cross-track controller and calculates a flight path angle reference command γ_{ref} as a function of the

UAV's altitude error h_e and the rate of change of altitude error $\dot{h}_e$, according to Equation (10).

$$\gamma_{ref} = K_h h_e - K_{\dot{h}} \dot{h}_e \tag{10}$$

In Equation (10), K_h and $K_{\dot{h}}$ are the gains for the altitude error h_e and the rate of change of altitude error $\dot{h}_e$, respectively. Again, the values of K_h and $K_{\dot{h}}$ are selected to achieve a balanced response.

2.4 Route planner

Each UAV has an assigned mission with a start and goal location at a specified altitude as well as a specified cruising airspeed. The route planner generates a long-term route for each UAV that comprises a series of waypoints to satisfy the UAV's mission. As mentioned in Section 2.3.1, the sequence of waypoint legs between these waypoints serves as the UAV's reference path that it must attempt to follow to reach its goal location.

The obstacles that each UAV is likely to encounter during its mission depend on its specified mission altitude, with obstacle density generally tending to decrease with increased mission altitude. It is therefore necessary for the route planner to identify which obstacles are present at each UAV's specified mission altitude so that a unique obstacle map can be created for that altitude and used to determine the UAV's long-term route. To do this, the route planner first intersects the entire the terrain model with a horizontal slab representing the UAV's mission altitude within some altitude tolerance, as shown in Figure 12. This intersection is computed using a process known as *polygon clipping*. The resulting 'clipped' geometry that was within the slab (Figure 12) is the terrain that will be considered when planning that UAV's long-term route.

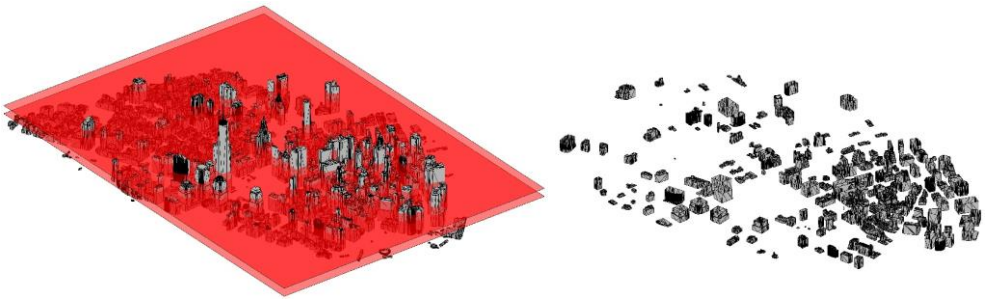

Fig. 12. Intersecting the terrain model with a horizontal slab (left) resulting in the clipped geometry within the slab (right).

Once the clipped geometry has been computed, it is projected onto a horizontal plane to create a unique 2D obstacle map for the specified altitude. The route planner can now use this obstacle map to generate a long-term route for the UAV's mission. To do this, the route planner first creates a *Voronoi diagram* between obstacles in the map. A Voronoi diagram is a spatial partitioning for a set of input geometry wherein each individual input geometry is enclosed in a cell, such that any point in a certain cell is closest to the input geometry in that cell. Figure 13 shows the Voronoi diagrams that were created using the obstacle maps at mission altitudes of 50 m and 200 m, respectively.

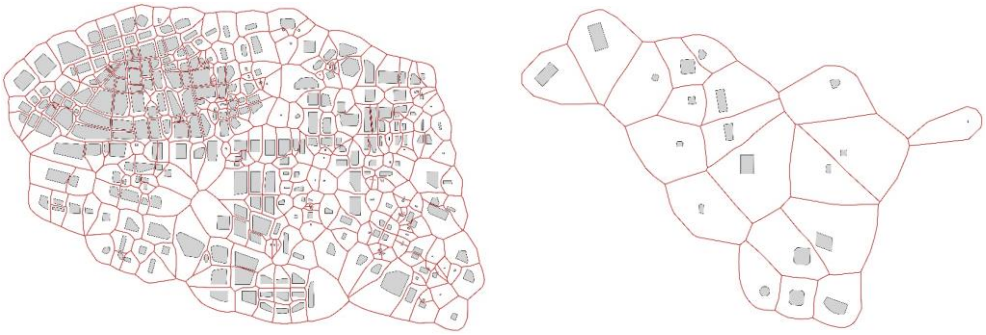

Fig. 13. Voronoi diagrams at mission altitudes of 50 m (left) and 200 m (right), respectively.

Consulting Figure 13, the boundary between neighbouring cells is, by the definition of a Voronoi diagram, the maximum-clearance path between obstacles. Thus, the route planner can generate a UAV's long-term route by connecting the UAV's start and goal locations to this maximum-clearance 'roadmap' [26] and then use a graph search algorithm (such as Dijkstra's algorithm) to compute the shortest path to the goal location along the roadmap.

2.5 Collision avoidance

While UAVs are following their long-term routes, collisions with terrain and other UAVs are predicted by forward propagating each UAV along its intended upcoming trajectory to check whether this intended trajectory results in a collision. The time up to which the UAVs are forward propagated is determined by the duration of the prediction horizon t_{hor}, which is chosen so that UAVs have enough time to perform two successive avoidance manoeuvres should a collision be detected at t_{hor} (one manoeuvre is typically sufficient with two manoeuvres only required in exceptionally congested collision scenarios). If no impending collisions are detected up to t_{hor}, UAVs keep following their long-term routes while continuously checking their intended trajectories for potential collisions.

Conversely, if an impending collision is detected within t_{hor}, collision resolution must be performed which involves cooperatively replanning the intended trajectories of all UAVs involved in the impending collision so that it can be avoided. The time up to which the intended trajectories are replanned is also determined by t_{hor}. This ensures that the intended trajectories of all UAVs are collision-free up to t_{hor} after collision resolution has been performed, thereby maintaining that collisions will always be detected a duration of t_{hor} in advance which gives UAVs sufficient time to attempt two successive avoidance manoeuvres.

The navigation system supports various collision avoidance manoeuvres including vertical and horizontal manoeuvres, as well as airspeed changes. The available set of avoidance manoeuvres can also be selected based on the expected congestion of a UAV's mission. Figure 14 shows an example of a vertical avoidance manoeuvre which is performed by temporarily adjusting the UAV's reference altitude with a predetermined increment causing the UAV to climb to that altitude and avoid the collision.

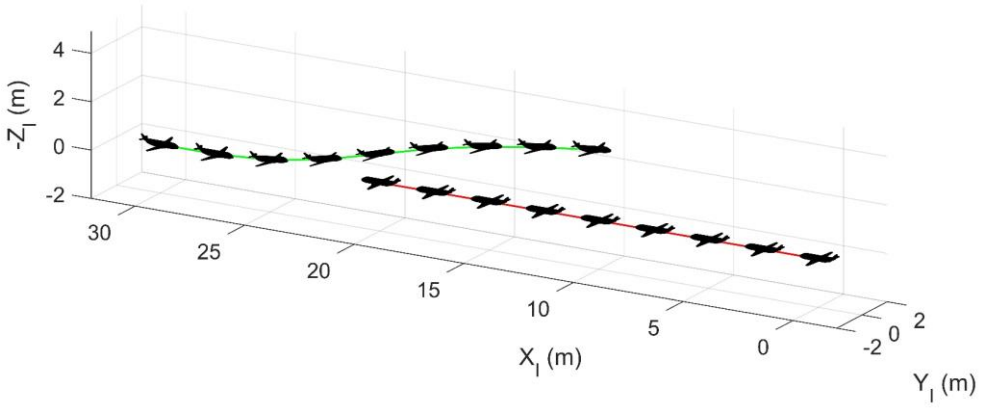

Fig. 14. Example of a vertical avoidance manoeuvre where the UAV from the left temporarily incremented its reference altitude to avoid colliding with the UAV from the right.

Upon entering collision resolution, each UAV involved in the impending collision grows a search tree of possible action/manoeuvre sequences up to the prediction horizon t_{hor}. The optimal sequence of actions for each UAV is then determined using a graph search algorithm such as the A^* algorithm, whose cost function is chosen to minimise each UAV's cumulative deviation from its long-term route during collision resolution.

As mentioned above, UAVs perform collision resolution cooperatively meaning that the UAVs involved in an impending collision communicate how they intend to replan their trajectories so that other UAVs can replan accordingly. This cooperative replanning is performed in a decoupled manner according to a predetermined priority order. This allows higher priority UAVs to replan their intended trajectories without considering lower priority UAVs who must subsequently replan around the intended trajectories of higher priority UAVs.

The decoupled approach to cooperative collision resolution seldom achieves the optimal solution [4] because trajectory replanning is performed in a priority order which means that not every possible action combination between UAVs is considered. Worse still, the specific priority order of UAVs in a collision scenario can occasionally cause collision resolution to fail despite a solution to the scenario existing.

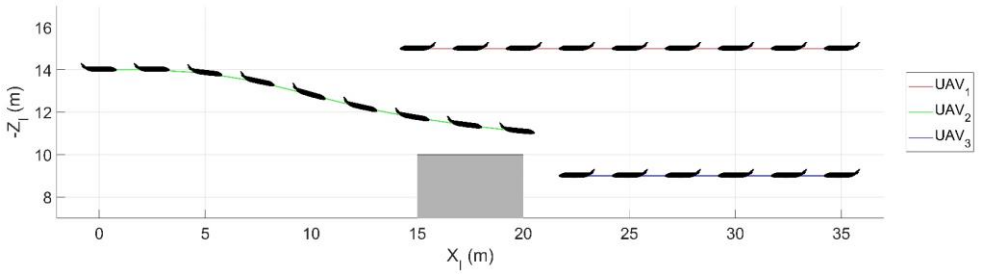

Fig. 15. Failure of the decoupled approach to resolve a collision scenario due to the priority order of the UAVs involved.

This is explained using Figure 15 where it is seen that UAV_3, the lowest priority UAV, is unable to avoid an impending collision because it had to replan its trajectory after both UAV_1 and UAV_2, which left UAV_3 without a viable resolution action. However, examination of the collision scenario in Figure 15 reveals that a solution to the scenario certainly does exist. That is, if UAV_3 was able to replan first to climb above the obstacle, UAV_1 and UAV_2 could both climb too, and all collisions would be avoided.

As mentioned in the literature review (Section 1.2), an alternative approach to cooperative collision resolution is the centralised approach. The centralised approach is not susceptible to the above priority order limitation of the decoupled approach, because it considers every possible action combination between UAVs and can therefore always find the optimal solution to a collision scenario (if one exists). However, research [4] has indicated that centralised approaches are far too slow to be considered for a system that aims to be representative of the real-life constraints of UAV autonomous navigation, especially for a large number of UAVs.

3 Results

This section provides a discussion of the results from the UAV autonomous navigation system. The section begins by presenting some illustrative results for both the route planner and the collision avoidance function. This is followed by an analysis of the system's numerical results from Monte Carlo simulations.

3.1 Illustrative results

The illustrative results are intended to provide visual confirmation that the route planner and collision avoidance function behave as expected. To this end, specific examples for each function are selected from simulations to be illustrated and discussed.

3.1.1 Route planner

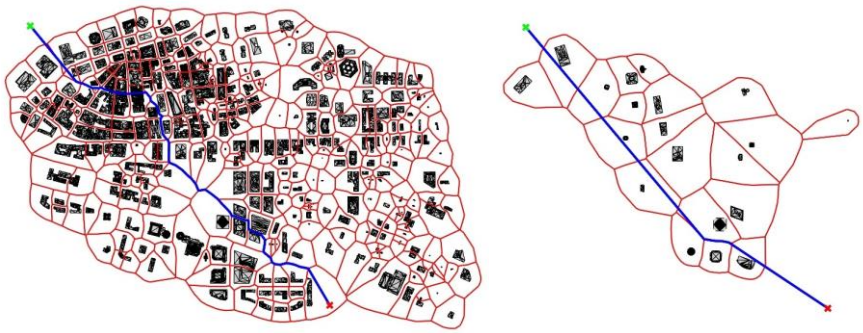

Fig. 16. Long-term routes generated by the route planner (blue) along the maximum-clearance roadmaps (red) for UAVs with the same start and goal locations at a mission altitude of 50 m (left) and 200 m (right).

Figure 16 shows the long-term routes that were generated by the route planner for UAVs with the same start and goal locations at mission altitudes of 50 m and 200 m, respectively. As seen in the figure, these routes are the shortest path between the UAV's start and goal locations along the maximum-clearance roadmap at that altitude. Interesting to note are the

longer straight-line connections that are possible from the UAV's start and goal locations onto the maximum-clearance roadmap at a mission altitude of 200 m because of the sparse arrangement of obstacles at higher altitudes.

3.1.2 Collision avoidance

Figure 17 shows a collision scenario from a simulation of the navigation system in which two UAVs successfully performed collision resolution. In this scenario, UAV_1 had a higher priority than UAV_2, so when the impending collision was predicted at t_{hor} and the UAVs began to cooperatively replan their intended trajectories, UAV_1 replanned its intended trajectory without considering the intended trajectory of UAV_2. Thus, UAV_1 opted to keep its original intended trajectory seeing as this is the most optimal trajectory for minimising deviation from its long-term route. UAV_2 had a lower priority than UAV_1 and therefore had to replan its intended trajectory while considering the intended trajectory of UAV_1. This meant that UAV_2 was forced to perform a vertical avoidance manoeuvre by temporarily increasing its altitude (while still following its long-term route) so that its intended trajectory up to t_{hor} was once again collision-free.

Interesting to note is that UAV_2 could have equally chosen to temporarily decrease its altitude by the same amount given that this would result in the same cumulative deviation from its long-term route as increasing its altitude. In fact, the only reason that UAV_2 chose to increase its altitude is because, by convention, collision resolution in the system first attempts an increase in altitude because a decrease in altitude is more likely to result in a subsequent collision with terrain.

3.2 Numerical results

Monte Carlo simulations were performed to generate numerical results that quantify the performance of the proposed UAV autonomous navigation system. Considering that obstacle density is generally dependent on altitude, the Monte Carlo simulations were performed in different altitude ranges to investigate the effect of altitude on the system's performance. In each altitude range, 1000 simulations were performed with each simulation comprising 100 UAVs. Each UAV was assigned a randomised mission that was generated by randomly sampling its start and goal locations, mission altitude (within the relevant range), cruising airspeed, and physical size. The UAVs were also restricted to only vertical avoidance manoeuvres. Figure 18 is a snapshot taken 50 s into one of the Monte Carlo simulations in the 50-60 m altitude range which illustrates the degree of UAV congestion in these simulations.

After each simulation, the number of route planner attempts, replans, and failures was recorded. A route that would have caused a UAV to collide with terrain and subsequently had to be replaced with an alternative route to the same goal location was considered a route planner replan. If every possible alternative route to the goal location caused the UAV to collide with terrain, this was considered a route planner failure. Similarly, the number of collision resolution activations and failures was recorded. A collision resolution activation simply refers to a single instance of collision resolution being triggered by a predicted collision. If a UAV had no collision-free actions to take during collision resolution, this was considered a collision resolution failure.

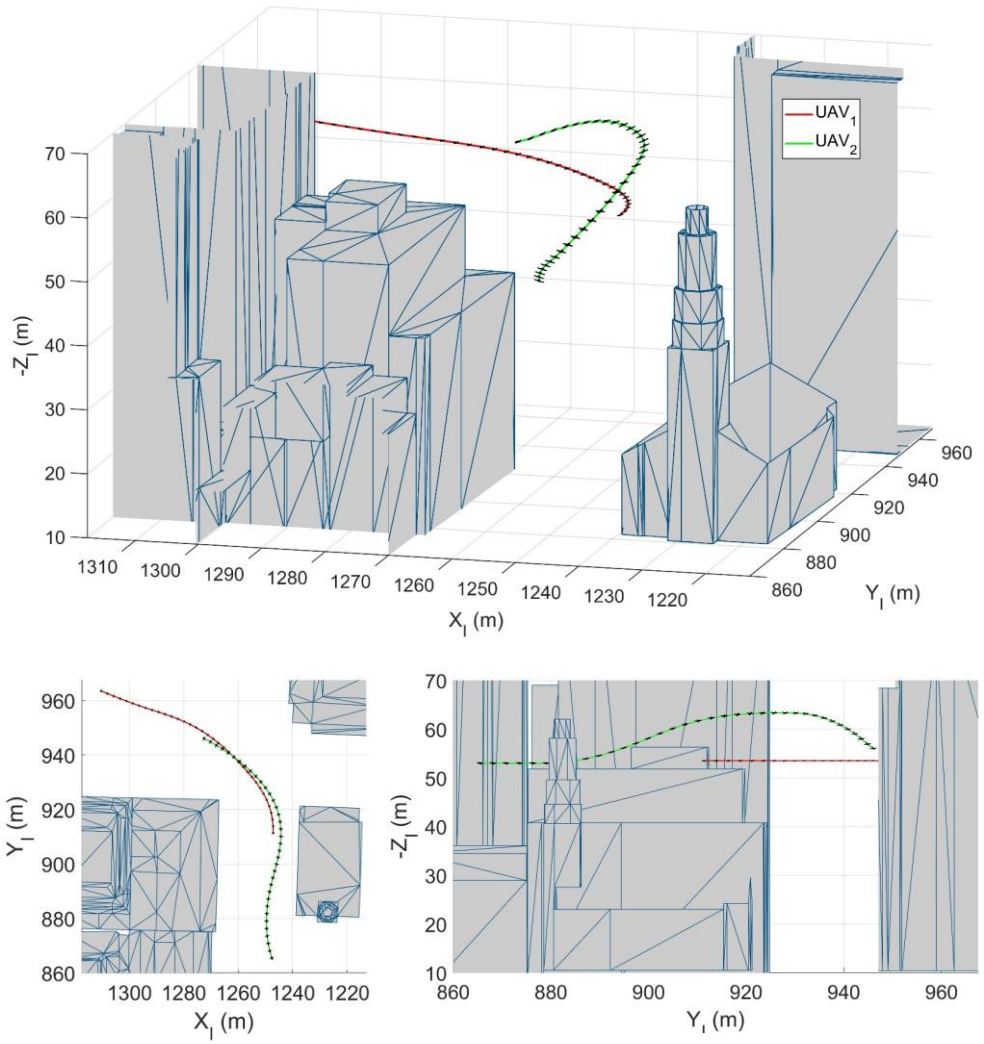

Fig. 17. A collision scenario demonstrating successful collision resolution between UAV_1 and UAV_2.

Fig. 18. Snapshot taken 50 s into one of the Monte Carlo simulations in the 50-60 m altitude range. The current position of each UAV is circled in red with its trajectory up to that position also shown.

The above parameters allow for the definition of performance metrics to evaluate the system's performance under different conditions. These performance metrics are defined in Equations $(11) - (13)$ and are, namely, the route planner replan rate (RR_{RP}), route planner failure rate (FR_{RP}), and collision resolution failure rate (FR_{CR}).

$$RR_{\mathrm{RP}} = \left(\frac{\text{replans}}{\text{attempts}}\right) \times 100 \tag{11}$$

$$FR_{\mathrm{RP}} = \left(\frac{\text{failures}}{\text{attempts}}\right) \times 100 \tag{12}$$

$$FR_{\mathrm{CR}} = \left(\frac{\text{failures}}{\text{activations}}\right) \times 100 \tag{13}$$

Table 1 and Table 2 below summarise the numerical results of the Monte Carlo simulations on the navigation system. Table 1 contains the numerical results for the route planner while Table 2 contains the numerical results for collision resolution. Figure 19 indicates the effect that mission altitude has on system performance by showing the system's performance metrics in each altitude range. In Table 1, note that the number of route planner attempts also includes routes that were discarded because they were shorter than the prediction horizon t_{hor}, or would have caused an inter-UAV collision within t_{hor} of the UAV starting its mission resulting in an unsolvable collision scenario. Despite these routes having to be discarded for the UAV's mission, their number of replans and failures was still recorded given that they are valid route planner attempts.

Table 1. Route planner numerical results.

Altitude range (m)	Attempts	Replans	Failures	Replan rate (%)	Failure rate (%)
50-60	101 832	11 866	1	11.65	9.82×10^{-4}
100-110	101 657	747	0	0.73	0
150-160	101 602	9	0	8.86×10^{-3}	0

Table 2. Collision resolution numerical results.

Altitude range (m)	Activations	Failures	Failure rate (%)
50-60	50 550	62	0.12
100-110	22 305	4	1.79×10^{-2}
150-160	15 730	3	1.91×10^{-2}

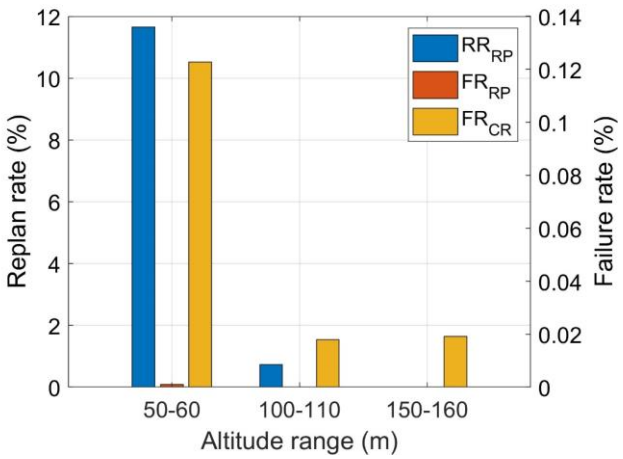

Fig. 19. Performance metrics for the system in each altitude range.

Consulting Figure 19, the route planner replan rate (RR_{RP}) and route planner failure rate (FR_{RP}) both tended to decrease with an increase in altitude range. This trend is expected considering the reduced obstacle density at higher altitudes (see Figure 16) which results in routes that are less likely to cause UAVs to collide with terrain.

Shown in Table 2, the number of collision resolution activations tended to decrease with an increase in altitude range. Again, this trend is expected considering the increased obstacle

density at lower altitudes meaning that UAVs are more likely to encounter each other during their missions and are subsequently required to activate collision resolution more frequently. This also increases the likelihood of collision resolution failure especially considering that collision resolution actions become limited due to the increased obstacle density at lower altitudes. This is confirmed by the significant reduction in collision resolution failure rate (FR_{CR}) in the 100-110 m and 150-160 m altitude ranges compared to the 50-60 m altitude range, as shown in Figure 19. Figure 19 also suggests that obstacle density may have less of an effect on collision resolution failure rate (FR_{CR}) above a certain altitude considering that FR_{CR} remained approximately the same in the 100-110 m and 150-160 m altitude ranges.

4 Conclusion

This paper presented an autonomous navigation system for multiple unmanned aerial vehicles (UAVs) in urban environments. The navigation system comprises a long-term route planner and a short-term cooperative collision avoidance function. Each UAV has an assigned mission with a start and goal location at a specified mission altitude as well a mission cruising airspeed. The route planner first generates a long-term route for each UAV to complete its mission, which they then attempt to follow while cooperatively predicting and resolving impending collisions with other UAVs and terrain. The navigation system uses a terrain model of Manhattan, NYC, to be representative of real-life urban terrain and considers the motion constraints of each UAV by incorporating time constants from real-life UAVs into its kinematic UAV model.

The navigation system was tested by examining illustrative results from both the route planner and the collision avoidance function, as well as performing Monte Carlo simulations to obtain numerical results that quantify the system's performance in different altitude ranges. The illustrative results indicate that both the route planner and collision avoidance function perform as expected. The numerical results show promising failure rates for both the route planner and the collision avoidance function. The route planner achieved a worst-case replan rate of 11.65 % and failure rate of 9.82×10^{-4} % over 101 832 attempts, while the collision avoidance function achieved a worst-case collision resolution failure rate of 0.12 % over 50 550 activations. As expected, the numerical results also indicate a decrease in system performance at lower altitudes due to the increased obstacle density.

This paper also highlighted an aspect of the navigation system that requires improvement. That is, the limitation of the decoupled approach to collision resolution wherein the specific priority order of UAVs in a collision scenario can cause collision resolution to fail despite a solution existing. Thus, future work on the system will focus on implementing a measure to address this limitation, with a potential solution being considered that involves temporarily adjusting the priority order of UAVs in a collision scenario until a solution is found, after which the original priority order will be restored. Future work will also introduce external disturbances and restrict the kinematic constraints of the UAVs further to verify whether the (appropriately) small failures rates will be sustained in more realistic simulations.

The research contributions of this paper listed in Section 1.3 were achieved by 1) implementing a bounding volume hierarchy (BVH) for optimised UAV collision detection with a 3D polygon mesh terrain model, 2) suggesting a method for computing maximum-clearance long-term routes through urban terrain by constructing a Voronoi diagram at a UAV's mission altitude, and 3) describing a cooperative collision avoidance function that grows a search tree of possible manoeuvre sequences to find the optimal avoidance trajectory that consider a UAV's motion constraints.

References

1. S.A.H. Mohsan, M.A. Khan, F. Noor, I. Ullah, M.H. Alsharif, Drones, Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review, **6**, [2, 15-17] (2022)
2. H. Shakhatreh, A. Khreishah, B. Ji, IEEE ICC, Providing wireless coverage to high-rise buildings using UAVs, 1 (2017)
3. X. Lin, C. Wang, K. Wang, M. Li, X. Yu, Transportation Research Part C: Emerging Technologies, Trajectory planning for unmanned aerial vehicles in complicated urban environments: A control network approach, **128**, 14 (2021)
4. L. Palmer, PhD dissertation (Faculty of Engineering Stellenbosch University), Cooperative Collision Avoidance Strategies for Unmanned Aerial Vehicles, [12-18, 85, 138] (2021)
5. M.V. Ramana, S. A. Varma, M. Kothari, IFAC-PapersOnLine, Motion Planning for a Fixed-Wing UAV in Urban Environments, **49**, [419, 423] (2016)
6. S. Bhagat, P.B. Sujit, ICUAS, UAV Target Tracking in Urban Environments Using Deep Reinforcement Learning, [698, 701] (2020)
7. J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, H. Yang, Neurocomputing, Path planning for solar-powered UAV in urban environment, **275**, 2064 (2018)
8. M. Kothari, I. Postlethwaite, D.W. Gu, JINT, UAV Path Following in Windy Urban Environments, **74**, 1027 (2014)
9. L. Singh, J. Fuller, Proceedings of the 2001 American Control Conference, Trajectory generation for a UAV in urban terrain using nonlinear MPC, **3**, 2301 (2001)
10. C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, D. Wu, IEEE Internet of Things Journal, Offline and Online Search: UAV Multiobjective Path Planning Under Dynamic Urban Environment, **5**, 553-557 (2018)
11. T. Castelli, A. Sharghi, D. Harper, A. Trémeau, M. Shah, ArXiv, Autonomous navigation for low-altitude UAVs in urban areas, 1 (2016)
12. F.M. Adolf, H. Hirschmüller, JINT, Meshing and Simplification of High Resolution Urban Surface Data for UAV Path Planning, **61**, 177 (2011)
13. IATA, Honeywell, Performance assessment of pilot response to Enhanced Ground Proximity Warning System (EGPWS), 5 (2019)
14. S. Aggarwal, N. Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, Computer Communications, **149**, 281 (2020)
15. D. Choi, D. Kim, K. Lee, NAECON, Collision Avoidance of Unmanned Aerial Vehicles In an Urban Environment, [25, 29-30] (2021)
16. S. Huang, R.S.H. Teo, K. K. Tan, Annual Reviews in Control, Collision avoidance of multi unmanned aerial vehicles: A review, **48**, 150 (2019)
17. FAA, Introduction to TCAS II, Version 7.1, 5 (2011)
18. NYC Department of City Planning, NYC 3D Model by Community District (2018)
19. FileInfo.com, .3DM File Extension (2016)
20. G. Turk, Leland Stanford Junior University, The PLY Polygon File Format, 1 (1994)
21. M. Attene, M. Campen, L. Kobbelt, ACM Computing Surveys, Polygon mesh repairing: An application perspective, **45**, 2 (2013)
22. C. Ericson, Real-Time Collision Detection, Geometrical Robustness, 465-468 (2005)
23. C. Ericson, Real-Time Collision Detection, Basic Primitive Tests, 226-227 (2005)

24. C. Ericson, Real-Time Collision Detection, Bounding Volume Hierarchies, 235-236 (2005)
25. C. Ericson, Real-Time Collision Detection, Bounding Volumes, 75 (2005)
26. S. M. LaValle, Planning Algorithms, Combinatorial Motion Planning, 260 (2006)