Noname manuscript No.
(will be inserted by the editor)

Fast 3D Collision Avoidance Algorithm for Fixed Wing UAS

Zijie Lin · Lina Castano · Edward Mortimer · Huan Xu

Received: date / Accepted: date

Abstract This paper presents an efficient 3D collision avoidance algorithm for fixed wing Unmanned Aerial Systems (UAS). The algorithm increases the ability of aircraft operations to complete mission goals by enabling fast collision avoidance of multiple obstacles. The new algorithm, which we have named Fast Geometric Avoidance algorithm (FGA), combines geometric avoidance of obstacles and selection of a critical avoidance start time based on kinematic considerations, collision likelihood, and navigation constraints. In comparison to a current way-point generation method, FGA showed a 90% of reduction in computational time for the same obstacle avoidance scenario. Using this algorithm, the UAS is able to avoid static and dynamic obstacles while still being able to recover its original trajectory after successful collision avoidance. Simulations for different mission scenarios show that this method is much more efficient at avoiding multiple obstacles than previous methods. Algorithm effectiveness validation is provided with Monte Carlo simulations and flight missions in an aircraft simulator. FGA was also tested on a fixed-wing aircraft with successful results. Because this algorithm does not have specific requirements on the sensor data types it can be applied to cooperative and non-cooperative intruders.

Zijie Lin
E-mail: zlin1@terpmail.umd.edu

Lina Castano
E-mail: linacs@umd.edu

Edward Mortimer
E-mail: eddym@umd.edu

Huan Xu
E-mail: mumu@umd.edu

Keywords fast obstacle avoidance · air vehicle obstacle avoidance · avoidance efficiency · optimal avoidance starting time

1 Introduction

Unmanned Aerial Systems (UAS) have been utilized in a number of civil and military applications, including important public missions such as disaster relief, firefighting, search and rescue, as well as other applications in agriculture, infrastructure and scientific research [1]. UAS are a fast growing industry with extensive economic implications and will eventually be integrated into the national airspace system (NAS) [2], which will require UAS to safely interact with other air vehicles. Standards for safe integration must be in place before a UAS can share the same airspace as manned aircraft, or when they operate in areas where they may inadvertently deviate into airspace designated to manned air vehicles. A UAS consists of an unmanned air vehicle (UAV) plus a ground control station, dedicated software, sensors, and telemetry, etc., all of which need research, development, testing and evaluation for safe integration.

Small UAS (under 55 pounds) may enter the mainstream of U.S. civil aviation provided they comply with Federal Aviation Administration (FAA) safety regulations [3]. One of the imperative technological requirements for safe integration of UAS into NAS is sense and avoid (SAA) capabilities. The purpose of SAA features is to prevent collisions with other traffic, natural fliers, population on the ground, or collateral damage to property. The SAA function needs to be able to provide a self 'separation' service, which would engage before a collision avoidance maneuver is needed and could therefore consist of gentler maneuvers [4].

Development of ground and airborne SAA sensors, data communication and avoidance algorithms need to provide solutions for safe separation from other aircraft and avert possible collisions. SAA algorithms should consider airspace restrictions when flying in relative nearness to restricted airspace for airports, military operation areas, temporary flight restriction areas and prohibited areas. These airspace limitations may create potential scenarios where the air vehicle has limited space to operate in. UAVs that stray from safe flight zones and into unpermitted airspace are also hazardous. According to a report [5] released by the Federal Aviation Administration, between Aug, 2015 and Jan. 31, 2016, about 600 UAVs were in close proximity to manned aircraft or close to airports.

Low altitude airspace UAS operations, such as agricultural and wildlife monitoring applications [6], may benefit from a collision avoidance mechanism which allows the UAV to stay on its intended trajectory as much as possible, while keeping a safe distance from any potential obstacles. Moreover, UAVs tasked with aerial surveillance and mapping at low altitudes may encounter unexpected features in the landscape with which they may collide [7]. FGA allows the aircraft to go back to it's original route after collision avoidance, while also allowing for a selective separation distance from the obstacle. Another class of applications where FGA may be of high relevance is for UAV landing operations, where there may be more vehicles in close proximity or on the ground.

Additionally, obstacles, e.g. buildings, birds and aircraft in the environment, have the potential to compromise the safety of the UAV's flight mission as well as of any obstacles encountered along its path. To guarantee operational safety of UAS, new detection and avoidance algorithms need to be developed and tested [8], [4]. These new algorithms need to be compatible with current and future manned aircraft collision avoidance systems such as TCASII/ACAS X [9], as well as surveillance systems such as automatic dependent surveillance-broadcast (ADS-B) and air traffic control (ATC) separation management procedures and tools [10].

In this work, we develop an algorithm for navigation scenarios where self-separation was either not possible or precluded by occurrence of emergent events and therefore obstacle avoidance was necessary. These scenarios have an increased likelihood of occurrence in low altitude operations where the vehicle might be more exposed to interacting with obstacles. In addition, FGA was designed such that selective start times for the avoidance maneuver would allow for shorter paths and selective separation from obstacles. This allows a UAV to get through busy airspace in a less costly and more time efficient way. This might be especially useful when the UAV is flying near restricted airspace areas and natural boundaries.

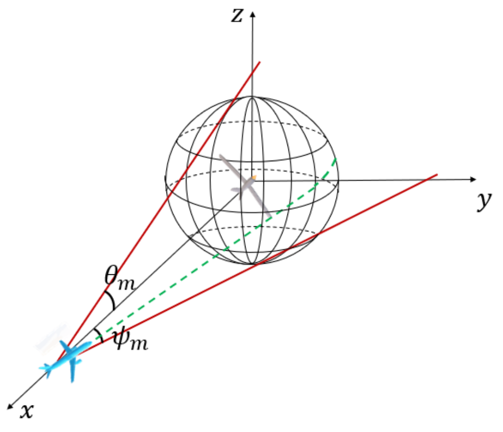


Fig. 1: UAV collision avoidance with FGA

FGA entails avoidance of collisions with static obstacles such as buildings, as well as avoidance of multiple dynamic intruders. A protected sphere of radius d_m is created around the obstacle and an obstacle free path is generated. Figure 1 shows the case of a critical avoidance maneuver. In this schematic, the UAV has calculated a collision free path (green) which turns at the closest separation from the obstacle, with a specific elevation and heading direction.

1.1 Overview of Avoidance Algorithms

Obstacle avoidance algorithms have different properties depending on the amount of information available from the environment. When sufficient information is available about the obstacle set and a single vehicle is present, then global planning methods may be used. These methods are based on the underlying hypothesis of having accurate global knowledge of the environment and obstacles in it [11]. Local planning methods are used when only local information is available, such as that provided by sensors. Sensor based planning methods can be global or local. Local sensor-based path planning approaches use local sensory information in a reactive way [12]. Reactive planning algorithms can use varying degrees of global information. Autonomous UAVs need a combination of both global and local methods, for successful navigation.

Global planning techniques compute a collision free trajectory with a configuration that complies with the vehicle constraints. These methods generally consist of searching for a path in a graph, where the graph is computed from the constraints. Some of these methods include: Dijkstra's algorithm, A*, B*, D*, D*Lite algorithms [13] [14]. The A* search algorithm is an extension of Dijkstra's that reduces the total number of states in the search by incorporating heuristics. B* is the best-first graph search algorithm that finds the least cost path from a given initial node to any goal node. D* or Dynamic A*, selectively expands nodes backwards from the goal node. All these search algorithms however have an increased computational cost, typically require environment modeling [14], and are more suitable for static obstacles.

Another broad classification for robot path planning entails classical [15], and heuristic methods [16], which include fuzzy logic, simulated annealing, and particle swarm optimization algorithms [17]. Some of the most important classical approaches for finding a collision free path include: Cell Decomposition, Roadmap, Mathematical Programming, Sampling Methods and Potential Fields. Cell Decomposition [15], partitions the configuration space into a set of cells in order to find a collision free set of connected adjacent cells. A visibility graph is an example of a Roadmap, where obstacles are represented as the locus of forbidden positions [18]. The force field approach, [19] is based on the concept of potential fields; it calculates the potential or the gradient of particular variables between the UAV and obstacles [20]. These reactive algorithms assume that obstacles cause an attraction and repulsion force, and find a trajectory to avoid collisions. However, local minima in these methods is a challenging issue, as well as oscillations when passing by closely spaced obstacles and narrow passages [21]. Computational efficiency is another drawback of these methods as they are better suited for single obstacles. When the obstacle number increases, the computational efficiency decreases.

The combination of the potential field method and a certainty grid produces another method called the Virtual Force Field (VFF) [22]. This method, however, has limitations when the vehicle is traveling at increased speed among densely cluttered obstacle environments. The Vector Field Histogram (VFH) eliminates the shortcomings of the VFF method [23]. The VFH method consists of constructing polar histograms around the vehicle and choosing an avoidance direction accordingly [24]. The histogram feature of this method makes it well suited to accommodate inaccurate sensor data, such as that produced by ultrasonic sensors. Improvements of this method include VFH+ [25], which includes binary histograms, and VFH*, which uses the A* algorithm to minimize the cost and heuristic functions. Navigation vector field [26] is another potential field type method which avoids local minima by creating a gradient field. These methods share similar limitations as the ones encountered in potential field methods.

Sampling-based path planning algorithms include rapidly-exploring random trees [27], and probabilistic road maps [28]. RRT based algorithms combine the environment construction phase, with the pathfinding phase [14]. Their strategy consists on expanding a tree by randomly sampling space and growing from a starting point until the tree is sufficiently close to a known target [29]. In each iteration the tree is extended by adding a new node. The main disadvantage of the RRT algorithms consists in their failure to stop execution and report when no possible solution exists. Optimal path planning algorithms, optimal trajectory [30] [31] and waypoint generation methods [32] [33], have also been studied. These methods generate obstacle-free paths but are particularly costly in computation time [34]. They test all available points around a cube to choose a sub-waypoint [32][33], then develop a path between the current waypoint and the sub-waypoint, and repeat this step until an optimal obstacle free path is found [34],[35]. Though these algorithms are successful at generating an obstacle-free path for a UAV in the situation of single or multiple obstacles, these methods increase runtime and are not practical for real-time avoidance [36][37].

Methods which use grids have two main drawbacks: obstacle modeling is likely to be less accurate, and increased memory overhead when increasing the resolution of the grid. In general, global methods are not well suited for fast collision avoidance due to slowness caused by their complexity. Potential fields and associated methods have issues with local minima. In addition, the associated optimization in potential field methods is done in an iterative way which becomes computationally expensive for multiple obstacles [38]. Sampling based methods are more computationally expensive as well, as they generate new waypoints out of exploring the environment. Probabilistic approaches do not guarantee finding a path within a finite time horizon. Reactive methods, however, are best suited for real-time avoidance.

Reactive methods use sensory information to issue motion directives. They compute a set of commands based on navigation strategies, such as calculating steering angles or velocities. Nearness Diagram is a reactive method which uses proximity to obstacles and areas of free space for navigation in cluttered environments [39]. Other reactive methods include boundary following algorithms [40] such as Bug algorithms [41] and wall-following algorithms [42]. These types of methods combine local planning with global information to guarantee convergence. Most of the bug algorithm methods consist of two reactive modes: move toward the target and follow obstacle boundaries [43]. These methods are more suitable for large static obstacles or blocked areas with large boundaries. Another method, the dynamic window approach, [44] consists of reducing the set of possible velocities, by considering velocities that can be reached within the next time interval and by maximizing an objective function within a dynamic window. This function includes a measure of progress toward a goal location, distance to the goal and the forward velocity of the robot. Velocity obstacles [45], reciprocal avoidance [46], and collision cones [47] are methods which select avoidance maneuvers based on current positions and velocities of the ownship (UAV) and obstacles.

There have been a number of other algorithms developed in the area of sense, detect, avoid (SDA) [48] [49]. Some work in the literature concentrates on the sensing and detection aspect, including improvement of sensor sensitivity [50],[51]. Sensor technologies which have been used for SAA include active interrogation transponders [4], electro-optic [52] , laser/LIDAR[10], onboard radar, ground based radar, ADS-B [53] and acoustic technologies [4].

Another widely used approach is found in geometric-based algorithms [4], [54],[46], which can be thought of as a type of see and avoid algorithm. In this work, we define geometric-based algorithms as those which use the geometric relationship between a UAV and its obstacle to calculate an updated path for the UAV. These algorithms require less processing power and are therefore more suitable for calculations onboard UAVs due to their size, weight and power constraints. Geometry based avoidance algorithms can be roughly classified into three kinds: ones which produce angle changes using geometry information such as the vehicle motion/location [4]; those which use velocity variation [46]; and those which use combinations of these as well as elements of other types of methods, e.g. global [55], and probabilistic [49]. Velocity based approaches also use geometric calculations but change the speed of the UAV to avoid the intruders instead of changing its direction.

Different aspects of the collision avoidance problem have been explored in the literature. Guidance laws, controllers, tracking points, maneuver types, different vehicle types, single/multiple obstacle types, optimizations, among other aspects, have been extensively explored. In the literature, geometric based algorithms have used linear and nonlinear guidance laws [38] to align heading and pitch angles to the desired spatial orientation. Minimum-effort guidance laws with collision cone approach were applied to guide the vehicle to a given collision-free waypoint [56]. Guidance algorithms that produce constant curvature maneuvers have also been formulated [57]. Strategies for geometric based collision avoidance include keeping desired relative bearing and elevation during the collision avoidance encounter [58]. Where the local planner switches between a pre-arranged controller and a collision avoidance controller. Analytical solutions to the encounter of two moving aircraft have been provided by [59] and [60], where geometric optimization results in a single discrete trajectory modification to avoid the obstacle. Analytical solutions are useful for performance evaluations and implementation of nonlinear controls. These however have not been implemented and demonstrated on hardware platforms. Hardware in the loop (HIL) tests have been conducted for collision avoidance using ADS-B transmitters for intruder aircraft [61]. Multiple obstacle formulations have also been explored using Velocity obstacle methods for cooperative conflict resolution [46]. Other geometric conflict resolution approaches include the vector resolution approach [54], which restricts avoidance maneuvers to the miss distance vector, which is formed between the UAV and the closest point of approach (PCA). To maximize the miss distance, a cost function is used to find the optimal solution.

In this work, we have chosen an angle-based geometric approach to reduce on runtime, and for ease of implementation on UAV testbeds. Our algorithm, FGA, uses geometric concepts for avoiding static and dynamic obstacles by changing the vehicle's direction, and assumes that the UAV is flying at constant speed. We consider that the obstacles are cooperative and we use a local planning approach that executes an avoidance maneuver after conflict detection. It then has access to the next waypoint, which is part of the higher level mission plan. In addition, we use curvature based guidance at the obstacle avoidance execution level and single point linear guidance to resume the UAV's waypoint course. Furthermore, we have formulated this algorithm for direct implementability on UAV platforms. This is missing from the current literature.

The contribution of this work entails the formulation of a real-time hardware-implementable reactive 3D obstacle avoidance algorithm (FGA), which uses selective start times to minimize deviations from its original mission plan. In addition, we have provided Monte Carlo simulations with high success rates, validating the utility of FGA. Furthermore, we successfully demonstrated implementation of FGA in an aircraft simulator (SITL) as well as flights on a physical UAV using virtual obstacles. Simulator and flight test results demonstrate the utility and feasibility of FGA.

This paper is organized as follows: Section II describes the problem formulation and model assumptions of FGA. Section III describes the working mechanism of FGA control algorithm. Section IV shows Matlab simulation results and analysis. Section V presents flight simulator results and hardware implementation on a fixed wing aircraft and test flights using FGA. Section VI draws conclusions and discusses future work.

2 FGA Problem Formulation

2.1 Algorithm Overview

This paper describes a geometric obstacle avoidance algorithm which classifies obstacles, performs avoidance operations and allows for the vehicle to return to its original navigation course. The Fast Geometric Avoidance algorithm (FGA) builds on concepts presented by [62]. It is constructed using differential geometry, which uses spatial location, relative velocities, angular displacements, as well as collision cone concepts [63] to determine probable conflicts [55].

The algorithm assumes that the fixed-wing vehicle is flying at a constant speed. Data about position and velocity of the obstacles is assumed to be given and accessible to the UAV. FGA first scans the nearby environment using an onboard sensor (which for the purposes of this paper is assumed to be a perfect sensor) and continuously determines if there are any obstacles that could potentially threaten the UAV and are closer than a safety surveillance distance R. After a potential obstacle is detected, two 2D collision cones, one in the x-y plane and another one in the x-z plane, in the body frame of the aircraft, are projected onto the encountered potential obstacle's protected sphere of radius d_m. It then determines which particular avoidance procedure to apply depending on whether the obstacle has been determined to be static or dynamic, and whether there are single or multiple obstacles. Depending on the identified obstacle type and associated information, FGA then automatically calculates the time to collision function to determine the optimal time for the UAV to start the avoidance operation. All the variable values needed for the avoidance maneuver are calculated and locally stored for each potential obstacle.

The avoidance starting time in FGA will also be selected based on airspace limitations, and how crowded the local airspace may be. This adds functionality to typical avoidance procedures, where obstacle avoidance operations normally begin upon detection of a potential conflict. The FGA algorithm allows for selective separation with respect to the closest point of approach (CPA). FGA also allows for selective start times based on desired separation from the obstacle. In the event that separation from the obstacle needs to be minimal, as may be the case in certain specific or emergency scenarios, the UAV triggers the avoidance maneuver at t_c, the critical avoidance time. A detailed explanation of this calculation is presented in the next section. FGA then triggers the avoidance operation and allows it to operate during a time window of flexible duration until it is determined that a safe distance from the obstacle has been reached. After that, the algorithm searches for the next reachable way-point located on the original path and lets the UAV return to the original intended path shortly thereafter. This minimizes the length of the path that is to be recalculated to avoid the obstacles, reducing computation time in comparison to way-point generation or optimal path planning methods, which typically recalculate full path lengths. A way-point generation method was compared in simulation to FGA, and results showed that FGA is faster and yields shorter avoidance paths.

The algorithm's distinct features can be summarized as a combination of: (1) obstacle classification after detection (2) shorter computation time as compared to other algorithms (see simulation results in Section 4, (3) selective avoidance maneuver starting time, (4) least changing path (67% shorter than optimal path planning method), (5) constant UAV speed, and (6) the ability to deal with complex environments such as multiple moving obstacles.

2.2 Obstacle Avoidance Geometry

If the surveillance distance R is breached for one or multiple obstacles, it is determined that those are now potential obstacles. Additional cylindrical and spherical surveillance layers can be added for obstacle monitoring. This is highly sensor dependent and will also be determined by the airspace segmentation.

After the obstacle is detected, FGA will consider that the obstacle is potentially a threat and will begin to calculate collision cones to ascertain whether the obstacle is actually a threat. The separation distance d_m is set around obstacle A, with coordinates $A(A_x, A_y, A_z)$, and is calculated based on available information about the obstacle. Based on relative velocity calculations, the obstacle will be deemed a threat if the relative velocity lies within the collision cone. If so, a corrective maneuver is issued and the UAV is steered out of that cone in pitch or yaw, or a combination of both.

Figures 2 and 3 show 2D views of a UAV flying towards a dynamic obstacle. Figure 2 shows a collision scenario where the relative velocity is inside the collision cone and therefore will not be at the desired minimum separation distance from the obstacle. In this case, FGA calculates a critical start time based on velocity and time to closest approach. After a corrective maneuver, the relative velocity can be altered so that it veers away from the collision cone, as show in Figure 3. Here the relative velocity is outside the collision cone and in this case the UAV is on an obstacle free path.

In these figures, $\overrightarrow{V_a}$ is the velocity of the obstacle ($||\overrightarrow{V_a}|| = 0$ for a static obstacle) and $\overrightarrow{V_u}$ is the velocity of the controlled UAV. The resulting relative velocity vector is $\overrightarrow{V_r} = \overrightarrow{V_{ua}} = \overrightarrow{V_a} - \overrightarrow{V_u}$ with initial angle θ_{vr0} while the relative distance vector is $\overrightarrow{r}$, with initial value $r_0 = \overline{UA}$ and initial angle θ_{r0}.

$\overrightarrow{V_r}$ and $\overrightarrow{r_0}$ determine whether the UAV will run into the obstacle's protected area of radius d_m or safe separation distance.

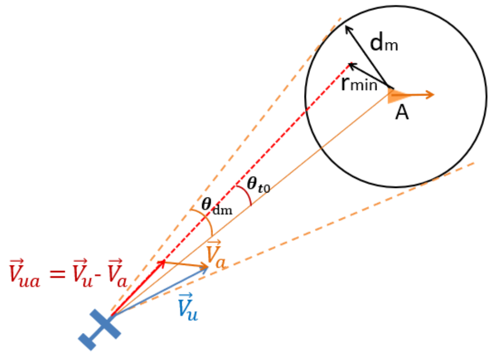


Fig. 2: FGA dynamic obstacle avoidance for relative velocity inside the 2D collision cone

In this work, the closest point of approach (CPA) is defined as the intersection of the tangent and the radius of the protected sphere. The 3D collision cone is defined by all the tangents that intersect the protected sphere with base radius $r_{min} \leq d_m$, and 3D opening angle $\nu = atan(r_{min}/r_0)$.

In Figures 2 and 3, the angle difference between the relative distance vector $\overrightarrow{r_0}$ and relative velocity vector $\overrightarrow{V_r}$ is defined as:

$$\theta_{t0} = \theta_{vr0} - \theta_{r0}, \tag{1}$$

and threshold angle of safe separation is calculated as:

$$sin\theta_{dm} = d_m/r_0. \tag{2}$$

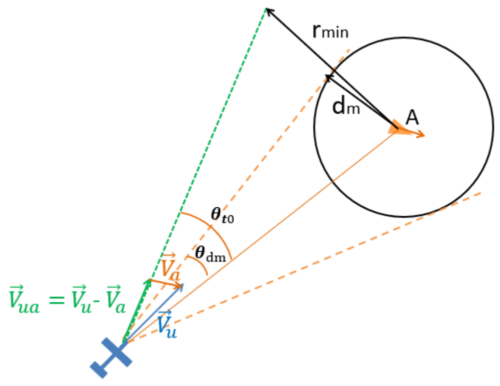


Fig. 3: FGA dynamic obstacle avoidance for relative velocity outside the 2D collision cone

The collision cone is then defined by:

$$r_{min} < d_m, \tag{3}$$

$$|\theta_{t0}| < sin\theta_{dm}, \tag{4}$$

and the safe cone is defined by:

$$r_{min} \geq d_m, \tag{5}$$

$$|\theta_{t0}| \geq sin\theta_{dm}. \tag{6}$$

Table 1 shows the definitions of the variables that are involved in FGA.

Table 1: Main variables in FGA

$U = U_o$	Location of UAV when obstacles are detected
$A_i = (A_x, A_y, A_z)$	Obstacles' location when they are detected
$V_r(V_u)$	Velocity of the UAV
V_a	Velocity of the obstacle
$V_r(V_{ua})$	Relative Velocity
C	PCA, point of closest approach
ρ	Turn radius
r_0	Initial distance between ith obstacle and UAV when detected
R	Surveillance radius
r_c	Critical relative distance at which the avoidance operation must start immediately
d_m (d_m, h, d_m, v)	Minimum safe radius (in horizontal or vertical). Note that $R > r_c > d_m$
θ_{dm}	Threshold angle of safe region
θ_{t0}	Absolute value of relative velocity angle minus relative location angle

2.3 Model Assumptions

The obstacle avoidance simulations in this paper, performed in Matlab, are based on the following assumptions:

(1) The fixed-wing UAV model we are using has a velocity which ranges from 13 m/s to 19 m/s and is set to be constant during the simulation [4]. The maximum bank angle is set to 30 degrees.

For a coordinated turn in x-y, the turn radius of the fixed wing UAV, in terms of airspeed and bank angle ϕ is:

$$\rho = \frac{V_u^2}{gtan\phi}. \tag{7}$$

The minimum turn radius $\rho = \rho_{min}$ of an aircraft is a property of the vehicle kinematics, generally a function of velocity and bank angle of a fixed wing air vehicle. From Equation 7 we have that:

$$\rho_{min}(V_u) = \frac{V_u^2}{gtan\phi_{max}}, \tag{8}$$

where ϕ_{max} is usually set to be 30 degrees and V_u is set to be constant during the avoidance operation.

A a pure heading rate is then given by:

$$\dot{\psi} = \frac{V_u}{\rho} = \frac{gtan\phi}{V_u}, \tag{9}$$

and the aircraft kinematic model is given by:

$$\dot{x} = V_u cos(\theta)cos(\psi), \tag{10}$$

$$\dot{y} = V_u cos(\theta)sin(\psi), \tag{11}$$

$$\dot{z} = V_u sin(\theta). \tag{12}$$

(2) Controller in FGA: Let the current horizontal angle (heading angle) of UAV be ψ and the vertical angle (pitching angle) of UAV be θ. The commanded heading and pitching angle are ψ_c and θ_c. Let the proportional and integral parameters of the controller be K_p and K_i, such that:

$$\dot{\psi} = K_p(\psi_c - \psi) + K_i \int (\psi_c - \psi), \tag{13}$$

$$\dot{\theta} = K_p(\theta_c - \theta) + K_i \int (\theta_c - \theta). \tag{14}$$

(3) The FGA algorithm does not have any requirements on a specific type of onboard sensors. Any sensors that can be installed on a UAV and provide accurate location and speed data are applicable to the FGA algorithm, similar to [53]. In this work we do not consider uncertainties that arise from sensor accuracy and environment changes.

(4)The search radius of the sensor onboard the UAV is set to be 50 times of the minimum separation radius d_m. And the potential danger radius R is set to be 10 times the minimum separation radius d_m. These parameters can be changed according to the different requirements.

(5) In the simulation, all of the obstacles are randomly generated by an obstacle generation module, to guarantee general properties. The module generates 10-20 static or dynamic objects for each case and provides their information to the detection module. Usually about $20-30\%$ of objects become actual obstacles for the UAV. The information generated for the obstacles includes velocity, initial location, direction:

– Test the UAV's capability to face different moving obstacles. The velocity of the obstacle is randomly generated in three ranges: 15-20 m/s, 35-40 m/s, 55-60 m/s. The successful rate of obstacle avoidance changes as the obstacles' velocity varies.
– The location of obstacles is generated 360 degrees around the UAV and the initial distance from obstacle to UAV goes from 100 m to 500 m.
– The heading angle of each obstacle can vary between 0 and 360 degrees relative to the UAV.

(6) The radius d_m of the protected sphere depends on the obstacle characteristics. Obstacles which are larger in size and have a higher velocity have a larger safe zone, which is described by d_{ma}. Similarly, larger fixed-wing UAVs with a higher speed require a larger safe area d_{mu}. The total radius will be a combination of d_{ma} and d_{mu}, and additional factors depending on airspace limitations and distance to other obstacles.

For the software in the loop (SITL) simulator runs with the Ardupilot flight software, the following assumptions were made:

(1) The UAV model used was a 3D kinematic aircraft standard model going at a constant speed, and constant heading before detection of the obstacle. The aircraft had aerodynamic characteristics in the form of linearized coefficients for lift, and drag, corresponding to a Rascal 110 RC aircraft.

(2) The obstacle was implemented as a GPS point object with a cylindrical protected volume of radius d_m.

(3) Altitude is held constant during the FGA avoidance maneuvers.

(4) Crosswinds were not used during the simulator runs.

(5) Controllers used in the simulator are those found in the Ardupilot flight software suite.

3 Fast Geometric Avoidance Algorithm (FGA)

In the Matlab simulation, different kinds of obstacles are generated to create a test environment for the FGA algorithm. After potential obstacles have been detected and determined to be a threat to the UAV, the FGA is triggered.

The FGA algorithm includes three major processes. First, an automatic selection of the avoidance module depending on the type of obstacles detected (static or dynamic, single or multiple) (see Algorithm 1). Second, one that calculates the optimal avoidance starting time to shorten the avoidance path (see Algorithm 2). FGA aims at decreasing computation time, while producing least path deviations from the initial path and keeping the cruise speed of the UAV constant. Third, when the avoidance operation stops, the algorithm returns the UAV to the next feasible waypoint (see Algorithm 3). FGA checks the remaining waypoints one by one from the nearest to the farthest relative to the UAV. If a waypoint satisfies the constraints of the UAV's minimum turning radius ρ_{min} and maximum heading or pitching angular velocity and other kinematic constraints, the UAV will fly to this new waypoint and then continue its initial path. During the flight to this waypoint, the UAV will be following a Dubin's path.

3.1 Automatic Avoidance Module Selection based on Obstacle Classification (or Collision Scenarios)

In order to select the obstacle avoidance module, FGA will go through the following steps:

- First, it will determine if the obstacles encountered are static or dynamic. Static obstacles include buildings, electric towers and trees. Dynamic obstacles include other aircraft or flying animals such as birds.
- Next, it will start the avoidance procedure for the UAV. The UAV can choose to pitch up, down, head left or right, or a combination of these four major directions. The combination, which will yield a 3D turn, is based on the cost of the chosen heading and pitch angle (ψ_c, θ_c).

3.2 FGA Avoidance Mechanism

This section describes the characteristics of the avoidance mechanism in FGA, which applies to both static and dynamic obstacles.

Some of the important parameters in the avoidance maneuver calculations involve the vehicle's minimum turn radius ρ_{min} as calculated in Equation 8, and the estimation of the protected distance d_m. These quantities are used in the calculation of the avoidance time and the critical avoidance time.

3.2.1 Obstacle protected distance

In FGA, the safe separation radius d_m is centered at the obstacle location with coordinates $A(A_x, A_y, A_z)$, and delimits a volume which the UAV should not enter. The estimation of this parameter should include vehicle dimensions d_{ma}, vehicle velocity d_{mu}, and airspace limitations $d_{lim_{air}}$ as well as space limitations related to nearby obstacles $d_{lim_{obs}}$. In general, d_m can be formulated as:

$$d_m = c_1 d_{ma} + c_2 d_{mu} + c_3 d_{lim_{air}} + c_4 d_{lim_{obs}}, \tag{15}$$

Where the constants $c_i, i = [1..4]$ are determined based on mission constraints.

FGA combines distance, relative velocity and calculation of avoidance time.

3.3 Avoidance Time

The avoidance time t_a is calculated for each obstacle which has been determined to be a threat. It is calculated using the distance to the obstacle r_0, relative angular displacement θ_{t0}, protected distance of the obstacle d_m, and relative velocity V_r. The avoidance time is calculated as:

$$t_a = \frac{r_0 cos\theta_{t0} - \sqrt{(\rho + d_m)^2 - (\rho + r_0 sin\theta_{t0})^2}}{||V_r||} \tag{16}$$

Figure 4 shows the parameters involved in the calculation of the avoidance time.

The detection time t_0 is the time at which the UAV detects an actual obstacle, after being selected as such, among the potential obstacles. The critical avoidance time t_c is the time beyond which, the aircraft will trespass the protected area of radius d_m centered at the obstacle location. Figure 5 shows an avoidance maneuver which is happening at t_c for two cases. One where the UAV's heading is not collinear with the obstacle (Figure 5(a)) , and two, when the UAV's heading is collinear with the obstacle (Figure 5(b)). The avoidance maneuver can be triggered at any time between t_0 and t_c, depending on mission goals as well as airspace limitations such as those represented by $d_{lim_{air}}$, and distance to obstacles $d_{lim_{obs}}$.

Mission goals include whether the shortest path to a goal location is preferred, or if the least deviation from the original path is desired. It may also be the case that it is possible to have an increased separation from the

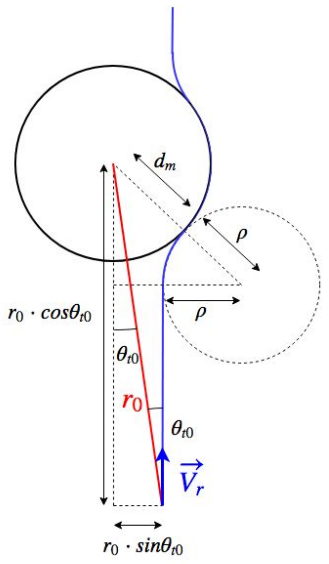


Fig. 4: Calculation of avoidance time

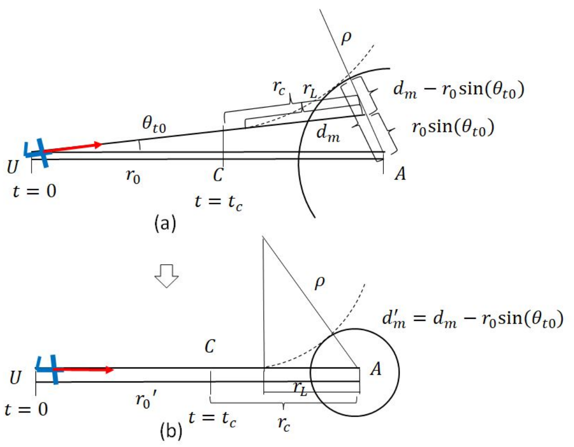


Fig. 5: Time control process of the Fast Geometric Avoidance algorithm

obstacle. For the 2D avoidance maneuvers, the path of least deviation will be executed when using the minimum turn radius, triggered at time t_c. The shortest path, as calculated in [64], can be achieved by using the minimum turn radius and a tangent to the protected disk of radius d_m. These two types of paths are part of FGA and can be triggered depending on mission constraints. If an increased radial distance from the obstacle is required in a particular mission scenario, then the turn radius will be calculated such that it takes into account all the relevant constraints (Equation 15).

The avoidance time encodes the features of the avoidance geometry and can be used to trigger the avoidance maneuver. Table 2 shows a classification of threat level based on avoidance time calculations. The avoidance time calculation has three main functions:

(1) Determine the risk of intruding into the obstacle's protected distance.

(2) Determine the optimal time and optimal location for the UAV to start the avoidance.

The avoidance time can incorporate safety considerations based on the type of obstacle encountered if $t_0 > t_a > t_c$. An obstacle which is known to be carrying passengers will cause the FGA to be triggered much faster and will be the highest priority causing the avoidance maneuver to begin at t_0. Whereas when it is determined that the obstacle is, for instance a natural barrier, the avoidance can begin at later times $t_a > t_0$. Avoidance times will be closer to t_c either by emergency procedures or in situations where the avoidance is highly constrained. A safety score constant C_s may be useful for determination of the avoidance time $t_a = C_s t_c$, where C_s has values between 0 and 1. An obstacle with passengers would have $C_s = 0$ and a natural barrier would have $C_s = 1$.

Table 2: Collision likelihood and avoidance operation determination using avoidance time

Avoidance Time	**Collision Likelihood**	**Avoidance**
$t_a < 0$	Imminent	Start immediately
$t_a = t_c = 0$	Critical	Start at t_c
$t_0 < t_a < t_c$	Likely	Start depending on constraints

3.3.1 Critical avoidance time

The critical avoidance starting time t_c is a time after which the UAV is unable to avoid entering the avoidance area of radius d_m of the obstacles. This is applicable to cases where path deviations are undesirable or during emergency procedures that require more agile vehicle maneuvers.

If $\rho = \rho_{min}$ in Equation 16, the last geometrically feasible time t_L can be calculated as:

$$t_L = \frac{r_0 cos\theta_{t0} - \sqrt{(\rho_{min} + d_m)^2 - (\rho_{min} + r_0 sin\theta_{t0})^2}}{||V_r||}. \tag{17}$$

If we use a geometric approximation which amounts to adding a buffer time t_b, as shown in Figure 5, we have that the critical avoidance time is:

$$t_c = t_L - t_b \tag{18}$$

or,

$$t_c = \frac{r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0})}{||V_r||} \tag{19}$$

The r_c corresponds to the distance between the UAV and the obstacle at time t_c:

$$r_c = r_0 - ||V_r|| t_c = r_0 - r_0 * cos\theta_{t0} + \rho + d_m - r_0 sin\theta_{t0}. \tag{20}$$

In addition, due to physical constraints, the start time for the avoidance operation needs to anticipate system response, such that $t_k = t_c - t_{res}$. Therefore:

$$t_k = t_c - t_{res} = t_L - t_b - t_{res}. \tag{21}$$

The avoidance operation will be successful if the UAV starts before the avoidance time t_k. In the specific case where $\theta_{t0} = 0$, as shown in Figure 5(b), the minimum geometric feasible distance r_L to avoid a collision with the obstacle is:

$$r_L = \sqrt{(\rho + d_m)^2 - \rho^2}, \tag{22}$$

and the critical distance becomes:

$$r_c = \rho + d_m, \tag{23}$$

where,

$$r_c \geq r_L. \tag{24}$$

The critical time is then equal to:

$$t_c = \frac{r_0 - \rho_{min} - d_m}{||V_r||} = \frac{r_0 - r_c}{||V_r||}. \tag{25}$$

3.4 FGA Collision Avoidance Bounds

In this section, we calculate the bounds or margins for when FGA is valid, that is, for the UAV to avoid collisions with detected obstacles. Three bounds are defined for this purpose in 2D and 3D. For both, Bound 1 establishes a detection distance margin, Bound 2 entails the bound on the UAV velocity and the obstacle velocity, and Bound 3 expresses the margins on the UAV heading angle.

3.4.1 2D Avoidance Bounds

Bound 1: Boundary on the initial distance to guarantee $t_c > 0$.

Based on the critical avoidance time calculation in Eq. 19, a critical avoidance time will exist if:

$$t_c = \frac{r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0})}{||V_r||} > 0, \tag{26}$$

rearranging this equation, we can see that:

$$r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0}) > 0, \tag{27}$$

$$r_0 cos\theta_{t0} + r_0 sin\theta_{t0} > \rho_{min} + d_m. \tag{28}$$

Such that the initial detection radius bound is:

$$r_0 > \frac{\rho_{min} + d_m}{cos\theta_{t0} + sin\theta_{t0}}, \tag{29}$$

where θ_{t0} is the angle between the relative velocity vector and relative location vector of the UAV and the obstacle.

For n obstacles, the UAV should satisfy the distance constraint for each:

$$r_{0i} > \frac{\rho_{min} + d_{mi}}{cos\theta_{t0i} + sin\theta_{t0i}}, i = 1, 2, ...n. \tag{30}$$

Bound 2: Boundary on the UAV velocity and obstacles velocity.

In Figure 6, the initial location of the UAV is set at U(0). The initial dynamic obstacle location is $A(A_x, A_y)$. The velocity of the UAV is $\overrightarrow{V_u}$, which points in the positive x axis for simplicity. The velocity of the obstacle is $\overrightarrow{V_a}$. The surveillance distance R becomes r_0 upon the detection of the obstacle.

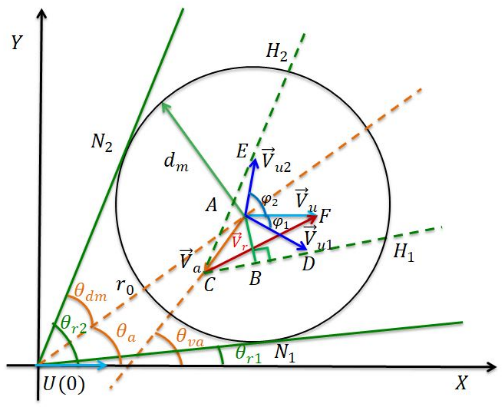


Fig. 6: 2D collision avoidance geometry for Bound 2: UAV can turn clockwise or counterclockwise to avoid crossing protected area d_m in 2D space.

The safety cone, as illustrated in Figure 2, is defined by tangent line UN_1 and UN_2 in Figure 6. If the relative velocity remains outside the cone defined by UN_1 and UN_2, then collision with the obstacle is avoided.

Let's now move the UAV velocity vector to the obstacle center A such that $\overrightarrow{AF} = \overrightarrow{V_u}$, and $|AF| = |\overrightarrow{V_u}|$. The velocity of the obstacle, $\overrightarrow{V_a}$, has absolute value $|\overrightarrow{AC}| = \overrightarrow{V_a}$, and is generated in random directions in our obstacle avoidance simulations. Let's now draw line segment $CH_1 // UN_1$. In order for a collision to be avoided, the relative velocity can be rotated clockwise or counterclockwise. Looking at the clockwise solution first, the relative velocity needs to be in the direction of CH_1 or outside line CH_1. This means the UAV needs to rotate at least φ_1 radians to $\overrightarrow{V_{u1}}$ ($|\overrightarrow{V_u}| = |\overrightarrow{V_{u1}}|$), and have at least one intersection point with line CH_1, or point D. Therefore, if $\overrightarrow{AD} = \overrightarrow{V_{u1}}$, there will be a collision free solution. This constraint is equivalent to letting triangle $\triangle ACD$ exist, which requires $AD \geq AB$. Where $|AB|$ is perpendicular to line CD. Therefore,

$$V_u = |\overrightarrow{V_u}| = |\overrightarrow{AD}| \geq |AB|, \tag{31}$$

and

$$V_u \geq |AB|. \tag{32}$$

In the triangle $\triangle ACD$,

$$|AB| = |AC||sin\angle ACD| \tag{33}$$

Where,

$$\angle ACD = \theta_{va} - \theta_{r1} \tag{34}$$

and,

$$|AC| = |\overrightarrow{V_a}| = V_a \tag{35}$$

Therefore:

$$|AB| = |AC||sin\angle ACD| = Va|sin(\theta_{va} - \theta_{r1})| \tag{36}$$

and since $V_u \geq |AB|$, we now have that:

$$V_u \geq V_a|sin(\theta_{va} - \theta_{r1})| \tag{37}$$

or,

$$V_a \leq V_u/|sin(\theta_{va} - \theta_{r1})|. \tag{38}$$

Where θ_{va}, the direction of the obstacle velocity, is $\theta_{va} = tan^{-1}(\frac{V_{ay}}{V_{ax}})$. θ_a, the direction of the obstacle location, is $\theta_a = tan^{-1}(\frac{A_y}{A_x})$. In Figure 6, θ_{r1} is the angle between UN_1 and the x-axis, hence:

$$\theta_{r1} = \theta_a - \theta_{dm} = tan^{-1}(\frac{A_y}{A_x}) - sin^{-1}(\frac{d_m}{r_0}). \tag{39}$$

Thus, Equation 38 becomes:

$$V_a \leq V_u/|sin[tan^{-1}(\frac{V_{ay}}{V_{ax}}) - tan^{-1}(\frac{A_y}{A_x}) + sin^{-1}(\frac{d_m}{r_0})]|. \tag{40}$$

Therefore, for a UAV flying at constant speed, the speed of the obstacle and the UAV must satisfy Equation 40, for successful obstacle avoidance with FGA.

Similarly, if we look at the counterclockwise solution, we have that:

$$V_u \geq V_a|sin(\theta_{r2} - \theta_{va})| \tag{41}$$

or,

$$V_a \leq V_u/|sin(\theta_{r2} - \theta_{va})| \tag{42}$$

Then, the analogous solution becomes:

$$V_a \leq V_u/|sin[tan^{-1}(\frac{A_y}{A_x}) + sin^{-1}(\frac{d_m}{r_0}) - tan^{-1}(\frac{V_{ay}}{V_{ax}})]|. \tag{43}$$

Thus, to guarantee at least one solution, the UAV velocity needs to be greater than either the clockwise or the counterclockwise conditions:

$$V_u \geq min(V_a|sin(\theta_{va} - \theta_{r1})|, V_a|sin(\theta_{r2} - \theta_{va})|) \tag{44}$$

If there are n obstacles, for every obstacle, the speed of the UAV should satisfy:

$$V_u \geq min(V_{ai}|sin(\theta_{vai} - \theta_{r1i})|, V_{ai}|sin(\theta_{r2i} - \theta_{vai})|) \quad i = 1, 2, ...n \tag{45}$$

Bound 3: Boundary on the UAV angles:

From our previous discussion, the UAV velocity needs to rotate at least clockwise by φ_1 or counterclockwise by φ_2 to avoid a collision. That is, the suitable velocity direction variation of the UAV is: $\varphi \leq -\varphi_1$ or $\varphi \geq \varphi_2$. Writen as a uniform set:

$$\varphi \subseteq \Phi_i. \tag{46}$$

For n obstacles:

$$\varphi \subseteq (\Phi_1 \cap \Phi_2 \cap ...\Phi_n,) \tag{47}$$

Thus, to let the solution exist:

$$(\Phi_1 \cap \Phi_2 \cap ...\Phi_n) \neq \emptyset. \tag{48}$$

3.4.2 3D Avoidance Bounds

The 2D bounds can be extended to 3D constraints. In this case we have that: **Bound 1**: Extending Equation 19 to 3D space, we can apply analogous margins. In this case, $\overrightarrow{r_0}$ is a vector in 3D space, θ_{t_0} is a 3D angle, and $\overrightarrow{V_r}$ is a 3D vector.

Bound 2: An extension of the 2D collision cone to a 3D one, is shown in Figure 7. Inside this cone, we can define a cone K which starts a point C. The axis of this smaller cone is parallel to line UA and the apex angle of the two cones is the same. If the avoidance solution exists, the sphere centered at the obstacle A, and radius V_u must have a tangent point with the cone K. To find this, we cut the cone K by a plane passing through AC and the cone's axis. In this cut-plane, there will be two angles; one between the cone edge and the line AC and one between the cone edge and the axis. Figure 7 shows the 3D cones, and the smaller angle produced by the plane-cut, which is symbolized by σ. Thus, as in the 2D case, if a solution exists:

$$V_u \geq V_a|sin\sigma|. \tag{49}$$

For n obstacles, to let the solution exist:

$$V_u \geq V_{ai}|sin\sigma_i|, i = 1, 2, ...n. \tag{50}$$

Bound 3: The solid angle rotation of the UAV velocity for the i-th obstacle is:

$$\Omega \subseteq \Omega_i. \tag{51}$$

Where, for n obstacles:

$$\Omega \subseteq (\Omega_1 \cap \Omega_2 \cap ...\Omega_n), \tag{52}$$

Thus, to let the solution exist:

$$(\Omega_1 \cap \Omega_2 \cap ...\Omega_n) \neq \emptyset. \tag{53}$$

For 3D, the velocity of the UAV can rotate by a full 4π solid angle, which gives the UAV much more free space than in the 2D plane.

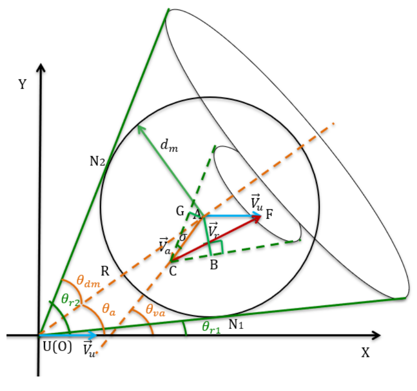


Fig. 7: 3D collision avoidance geometry for Bound 2.

3.4.3 Comparison of shortest paths to clear the obstacle

Sections 3.4.3 and 3.4.4 provide a comparison of the length of the updated path, starting at different times to perform collision avoidance. The proofs show that FGA, which starts at time t_c, has the shorter updated path as compared to the path starting from t_0.

For the following proofs we consider that the obstacle has been cleared when the UAV has reached the perpendicular intersection defined by the line between points: S and S', as shown in Figure 8. For these proofs we draw the comparison for the paths which start at t_c.

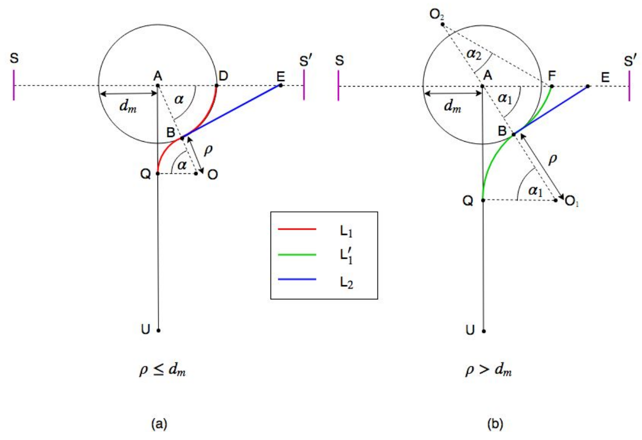


Fig. 8: Comparison of shortest paths from t_c to clear the obstacle: (a) Shortest path from t_c when $\rho \leq d_m$ and (b) Shortest path from t_c when $\rho > d_m$

Assuming the UAV starts from t_c, which is denoted by point Q in Figure 8, to prove that the shortest path clears the obstacle. We start with our first conjecture: all possible paths that pass the object share a common subsection which is characterized by L_1 in Figure 8, reaches point B - a tangent point to the obstacle. This path is a subset of all possible paths that clear the obstacle, since it is the only course of action that the UAV can initially take to avoid entering the avoidance radius while considering its non-holonomic restrictions. From point B, the comparison of two paths is made: continuing straight until one reaches point E or following the curvature of the obstacle until point D. This is also consistent with the Dubin's path planning theory [27].

In Figure 8, and through the following proofs we will assume that ρ is the minimum turning radius of the UAV. In Figure 8(a), $\alpha = \angle BAD$, where $\alpha \in (0, \pi/2)$, and B is the tangent point between ρ and the avoidance radius of obstacle A. Figure 8(a) shows $\rho \leq d_m$ and (b) shows $\rho > d_m$.

Proof (1) For $\rho \leq d_m$

where:
$L_1 : Q \rightarrow B \rightarrow D$
$L_2 : Q \rightarrow B \rightarrow E$
$L_1 = \alpha\rho + \alpha d_m$
$L_2 = \alpha\rho + d_m tan\alpha$

We can calculate:
$L_2 - L_1 = d_m(tan\alpha - \alpha), \alpha \in (0, \pi/2)$

Since:

$\frac{d}{d\alpha}(tan\alpha) = \frac{1}{cos^2\alpha}$ and $\frac{d}{d\alpha}(\alpha) = 1$,

for $\alpha \in (0, \pi/2)$, then
$\frac{1}{cos^2\alpha} \in (1, +\infty) \geq 1$

therefore $tan\alpha \geq \alpha$ always exists at $\alpha \in (0, \pi/2)$ and $d_m > 0$,

hence

$$L_2 > L_1.$$

(2) For $\rho > d_m$
After reaching tangent point B, since $\rho > d_m$, the UAV needs to fly around the avoidance radius using turning radius ρ and finally reaches the safe point F, as shown in Figure 8(b); this defines path L_1'. Or it follows the tangent line of circle A and arrives at point E, this defines L_2. Where $\alpha_1 = \angle BAE \in (0, \pi/3)$, $\alpha_2 = \angle BO_2D, \alpha_2 \in (0, \alpha_1)$. O_2 is the center of the UAV turn radius around the obstacle A.

Where: $L_1' : Q \rightarrow B \rightarrow F$
$L_2 : Q \rightarrow B \rightarrow E$
$L_1' = \alpha_1\rho + \alpha_2\rho$
$L_2 = \alpha_1\rho + d_m tan\alpha_1$

We calculate: $L_2 - L_1 = d_m tan\alpha_1 - \rho\alpha_2$

Since $\rho = O_1B = O_2B$

Also, $d_m tan\alpha_1 = CE \geq \rho tan\alpha_2$

So $L_2 - L_1' \geq \rho tan\alpha_2 - \rho\alpha_2 = \rho(tan\alpha_2 - \alpha_2) > 0$,

Thus: $L_2 > L_1'$. □

The following proof classifies deviation as the total time spent away from the initial heading before the avoidance maneuver, with a constant velocity constraint. This is equivalent to path length away from the initial heading. It is proved that the deviation of the UAV when starting the avoidance process from t_0 (the time at which the obstacle is detected), will be much longer than UAV starting from t_c (the critical avoidance time). That is: $L_3 > L_1$, these paths are shown in Figure 9.

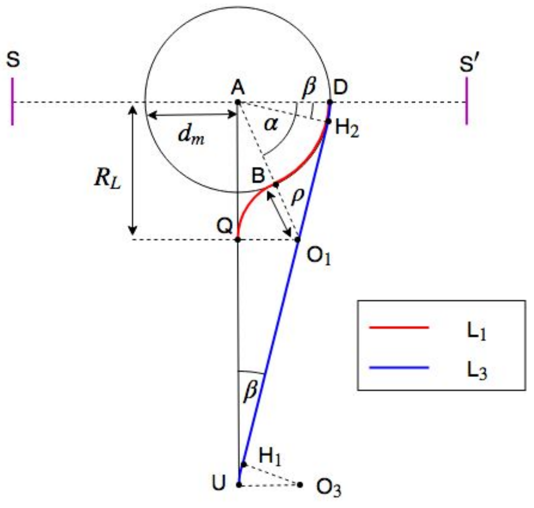


Fig. 9: L_1 and L_3 path comparison

Where ρ is the minimum turning radius for UAV, $\alpha = \angle BAE$, where point B is the tangent point of the path L_1 and the d_m is the avoidance radius of obstacle A. $\beta = \angle H_2AD$, where point H_2 is the tangent point of the path L_3 and circle A.

Proof $L_1 : Q \rightarrow B \rightarrow D$,
$L_3 : U \rightarrow H_1 \rightarrow H_2 \rightarrow D$.
$L_1 = \int dl = LB + BD = \alpha\rho + \alpha d_m$
$L_3 = \int dl = UH_1 + H_1H_2 + H_2D$

Let AU=R, where R is the initial distance between the UAV and obstacle when the obstacle is first detected.

Since the minimum distance between two points is the straight line, $L_3 = UH_1 + H_1H_2 + H_2D > UD$, where UD is the straight line from point U to point D. Let $L_{30} = UD$. Thus, $L_3 > L_{30}$ and the following will compare the length between L_1 and L_{30}.

Let $\rho = k_1 d_m, R = k_2 d_m$, we have that:

$$L_{30} = \sqrt{R^2 + d_m^2} = d_m(\sqrt{k_2^2 + 1}) \tag{54}$$

$$L_1 = (d_m + \rho)\alpha = d_m(k_1 + 1)\alpha \tag{55}$$

where $0 < \alpha < \frac{\pi}{2}$,

While α is a function of k_1, which decreases, let $\alpha = \frac{\pi}{2}$ and $L_1^* = d_m(k_1+1)\frac{\pi}{2}$ for brevity of this proof. Since it will be subsequently proved that conditions where L_3

is bigger than L_1^* where $\alpha = \frac{\pi}{2}$ must also extend to L_1 where $\alpha \leq \frac{\pi}{2}$. So the lower bound for k_2 will exceed that of which is necessary.

$L_1^* = d_m(k_1 + 1)\frac{\pi}{2}$
Let $L_{30} > L_1^*$, then:
$d_m(\sqrt{k_2^2 + 1}) > d_m(1 + k_1)\frac{\pi}{2}$, thus we have:

$$k_2 > \sqrt{\frac{\pi}{4}(1 + k_1)^2 - 1} \tag{56}$$

□

The Equation 56 is the lower bound of k_2. For any k_2 satisfies this equation, $L_{30} > L_1^*$. From the precondition of the FGA where $k_2 >> k_1$, eq 56 should always be satisfied in real world scenarios, where the detection radius is almost always much larger than the minimum turn radius. Since $L_3 > L_{30}$ and it can be said that $L_{30} > L_1^*$ and $L_1^* > L_1$, it is trivial to see that $L_3 > L_1$. An expanded version of this proof is presented in Appendix A.

3.4.4 Multiple Obstacles

For multiple obstacle cases, each moving obstacle has a t_{ci} associated with it, such that t_{ci} corresponds to the i-th obstacle. Therefore, for n multiple obstacles, the overall critical t_c, as calculated in Equation 18, is the minimum of all the calculated t_{ci}:

$$t_c = min(t_{ci}), i = 1, 2, 3...n; \tag{57}$$

and taking vehicle response into account:

$$t_k = min(t_{ci}) - t_{res}; i = 1, 2, 3...n; \tag{58}$$

For a multiple obstacles scenario, we can classify it roughly into two cases:

Case 1. More than one obstacle has overlap with the initial path of the UAV, as shown in Figure 10(a).

Case 2. Only one obstacle has overlap with the initial path of the UAV, as shown in Figure 11(a).

For Case 1, the area occupied by the multiple obstacles can be equal to a single obstacle with larger safe distance d_e, as shown in Figure 10(b). Thus, the problem to prove $L_1 < L_3$ transfers to the single obstacle situation which has been proved before.

For Case 2, there is more than one obstacle in the future path of the UAV, but only one obstacle is in direct conflict with the UAV's initial path. Thus, other obstacles can be ignored. The problem to prove $L_1 < L_3$ transfers to the single obstacle situation, as shown in Figure 11(b), which has been proved before.

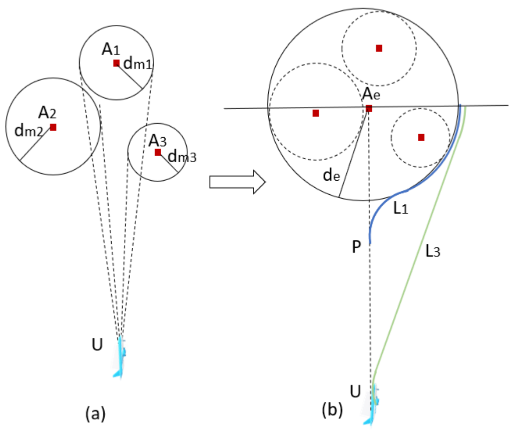


Fig. 10: Multiple obstacles: Multiple obstacles have overlap with initial path of the UAV

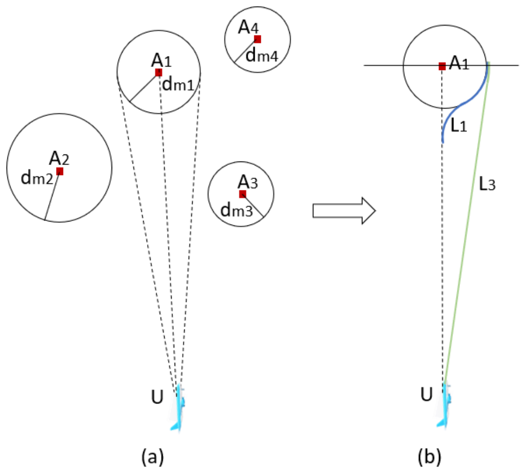


Fig. 11: Multiple obstacles: No obstacles have overlap with initial path of the UAV

3.5 Determination of Heading and Pitching Angle Simultaneously for UAV in 3D Collision Avoidance Maneuvers

FGA allows for obstacle avoidance maneuvers to be executed in the horizontal plane (heading maneuver) or in the vertical plane (pitching maneuver) [65] depending on which maneuver is feasible for the UAV at the avoidance time. Another type of avoidance maneuver in FGA is one which allows the UAV to make a 3D turn. This is achieved by combining heading and pitching maneuvers and is determined by a cost function.

Suppose the original heading and pitching direction of the UAV before encountering the obstacle is $(\psi_{ori}, \theta_{ori})$. The range of angles that the heading angle ψ_{range} and pitching angle θ_{range} can have are defined as follows:

$$-\psi_{range} < \psi_{ori} < \psi_{range};$$

$$-\theta_{range} < \theta_{ori} < \theta_{range}.$$

If we divide both $[-\psi_{range}, \psi_{range}]$ and $[-\theta_{range}, \theta_{range}]$ into n units, there will be n by n pairs of (ψ, θ) values. For pair (ψ_i, θ_j), the cost function is defined as:

$$Cost(k, j) = \eta_\psi(\varphi_i - \varphi_{ori}) + \eta_\theta(\theta_j - \theta_{ori})$$

Where η_ψ and η_θ is the coefficient of the horizontal and vertical direction deviation and they are positive numbers. Each one of the n by n pairs has its own associated cost. This is shown in Table 3.

The cost function will:

(1) Search from these pairs and mark which pairs can drive the UAV to a safe distance away from the avoidance area.

(2) Since changing a fixed-wing vehicle's pitch is less desirable due to possible collisions with the ground or to avoid loosing altitude, its maximum angular deflection in pitch will be less than the one for heading. Due to this limitation the cost coefficient for pitch η_θ is set to be larger than η_ψ.

(3) From marked pairs and their calculated costs, a minimum cost $Cost_{min}$ is found. Its response angle pair $[\psi(cost_{min}), \theta(cost_{min})]$ is then selected. The commanded heading and pitching angle of the UAV will be:

$$\psi_{command} = \psi(cost_{min});$$

$$\theta_{command} = \theta(cost_{min}).$$

Table 3: Cost function to determine command direction in 3D avoidance maneuver

$Cost_{11}(\psi_1, \theta_1)$	$Cost_{12}(\psi_1, \theta_2)$	...	$Cost_{1n}(\psi_1, \theta_n)$
$Cost_{21}(\psi_2, \theta_1)$	$Cost_{22}(\psi_2, \theta_2)$	...	$Cost_{2n}(\psi_2, \theta_n)$
...	...	...	...
$Cost_{n1}(\psi_n, \theta_1)$	$Cost_{n2}(\psi_n, \theta_2)$	...	$Cost_{nn}(\psi_n, \theta_n)$

3.5.1 FGA Algorithm

The Fast Geometric Avoidance algorithm can be described in three main sub-algorithms, one for detection (Algorithm 1), one for obstacle avoidance (Algorithm 2) and one for the next waypoint search (Algorithm 3).

Algorithm 1 FGA algorithm: Detection

```
Precondition: necessary data is available for UAV
 1: for $i \leftarrow 1$ to $n$ do
 2:     $\overrightarrow{r_0} \leftarrow \overrightarrow{A_0} - \overrightarrow{U_0}$                ▷ relative location vector
 3:     $\overrightarrow{V_{r0}} \leftarrow \overrightarrow{V_{u0}} - \overrightarrow{V_a}$               ▷ relative velocity vector
 4:     $\theta_{t0} \leftarrow f(\overrightarrow{r_0}, \overrightarrow{V_{r0}})$                        ▷ angle bias
 5:     $r_{PCA} \leftarrow r_0 \cdot sin\theta_{t0}$            ▷ closest distance in future
 6:     if $r_{PCA} < d_m$ then
 7:         $status \leftarrow dangerous$
 8:         $M_1 \leftarrow$ obstacle information
 9:     end if
10: end for
11: return $M_1, status$
```

Algorithm 2 FGA Algorithm: Obstacle Avoidance

```
 1: for $i \leftarrow 1$ to $m$ do
 2:     $t_{ci} \leftarrow \frac{r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0})}{||V_r||}$
 3: end for
 4: $t_c \leftarrow min(t_{ci})$
 5: if $t_c < 0$ then
        avoidance starts immediately
 6: else
        avoidance starts at $t_c$
 7: end if
        get $\varphi_{range}$ and $\theta_{range}$
 8: for $k \leftarrow 1$ to $k_1$ do
 9:     for $j \leftarrow 1$ to $j_1$ do
10:         $\varphi \leftarrow -\varphi_{range} + \varphi_{range}/k_1 * k$
11:         $\theta \leftarrow -\theta_{range} + \theta_{range}/j_1 * j$
12:         $\overrightarrow{V_{ut}} \leftarrow (V_u cos\theta cos\varphi, V_u cos\theta sin\varphi, V_u sin\theta)$
13:         for $l \leftarrow 1$ to $l_1$ do
14:             $\overrightarrow{r_{ltc}} \leftarrow \overrightarrow{A_{ltc}}(A_{lx}, A_{ly}, A_{lz}) - \overrightarrow{U_{tc}}(U_x, U_y, U_z)$
15:             $d = min(d_l) \leftarrow |\overrightarrow{V_{ut}} \times \overrightarrow{r_{ltc}}|/|\overrightarrow{r_{ltc}}|$
16:         end for
17:         $Cost(k, j) \leftarrow \eta_\psi(\varphi - \varphi_{ori}) + \eta_\theta(\theta - \theta_{ori})$
18:     end for
19: end for
        $Cost_{min}, \varphi_{com}, \theta_{com} = min(Cost(k, j)), \varphi_k, \theta_j;$
20: return $\varphi_{com}, \theta_{com}, newpath$
```

Algorithm 3 FGA Algorithm: Next Waypoint Search

```
1: for $i \leftarrow 1$ to $w$ do
2:     $\overrightarrow{r_{wi}} \leftarrow \overrightarrow{W_i}(W_{xi}, W_{yi}, W_{iz}) - \overrightarrow{U_t}(U_x, U_y, U_z)$
3:     $\theta_{wi} \leftarrow f(\overrightarrow{r_{wi}}, \overrightarrow{V_u})$             ▷ waypoints angle bias
4:     if $\theta_{wi} < \pi$ then            ▷ search the next waypoint
5:     end if
6: end for
7: return $W_i$                               ▷ get the next waypoint
```

4 Simulation Results and Analysis

This section contains the Matlab simulation results of the FGA algorithm. Table 4 shows the value of vehicle variables and obstacle parameters used in the simulations. The Monte Carlo simulations consisted in a set of 5000 to 20000 runs for randomly generated obstacle avoidance scenarios. The obstacles in these cases are randomly generated by an obstacle generator module we designed, as described in Section 2.3.

Table 5 summarizes the Monte Carlo simulation time for avoidance of static and dynamic obstacles using FGA. In this table, it is shown that the average time for the UAV to update its path to avoid static obstacles is 0.028s and 0.03s for dynamic obstacles. Which makes it suitable for hardware implementation. This is further indicated by Table 6, where FGA is compared with other collision avoidance methods. The average time consumed by FGA is much less than others, such as potential field method and optimal methods.

Table 4: Vehicle and obstacle variables for Monte Carlo simulations

V_a, **Velocity of the obstacle**	Randomly generated between 0 and 60 m/s
V_u, **velocity of the UAV**	15 m/s
ρ, **min turning radius**	30 m
R, **potential danger radius**	300 m
$r_c(t_c)$, **critical avoidance starting distance (time)**	Dependent on different situations
$d_m(d_m, v, d_m, h)$, **safe separate radius**	30m, $R > r_c > d_m$
η_ψ, **coefficient of heading angle deviation**	0.2
η_θ, **coefficient of pitching angle deviation**	0.5

Table 5: FGA obstacle avoidance time efficiency

Obstacle type	Num. of cases	Total computation time	Average time
Static obstacles	1413	39.564	0.028
Dynamic obstacles	3750	112.5	0.030
the number of obstacles in each case varies from 1-3			

Table 6: Time cost of major collision avoidance methods

Methods	aver. time each case	time saved by FGA	evaluation
Potential field [66]	0.082	63.41%	medium
Optimal 1[36]	0.18	83.33%	slow
Optimal 2 [32]	0.3	90.00%	slow

Figure 12 summarizes the Monte Carlo simulation results of collision scenarios where the UAV faces different static obstacles. The success rate decreases as (1) the initial distance (when the obstacle is first detected) decreases; (2) the number of those static obstacles increases.

Similarly, Figure 13 summarizes the Monte Carlo simulation results for the case of two dynamic obstacles. The success rate decreases as (1) the initial distance (when obstacles first detected) decreases; (2) the number of obstacles increases, which increases obstacle density; (3) the velocity of the dynamic obstacles increases, which reduces response time and increases difficulty for the UAV to avoid the obstacles. This figure shows the variation of failure rate with obstacle velocity when the initial distance is fixed (300-400m). If the obstacles' velocity is not larger than the velocity of the UAV (15m/s), 95% successful rate of FGA can be guaranteed.

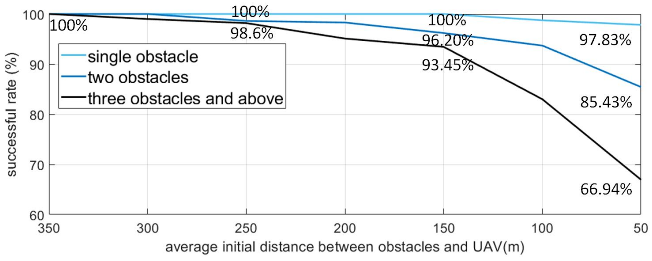


Fig. 12: FGA static obstacle avoidance success rate.

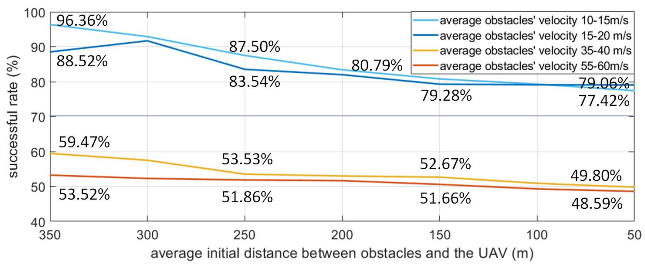


Fig. 13: FGA dynamic obstacle avoidance success rate with initial distance and obstacles' velocity.

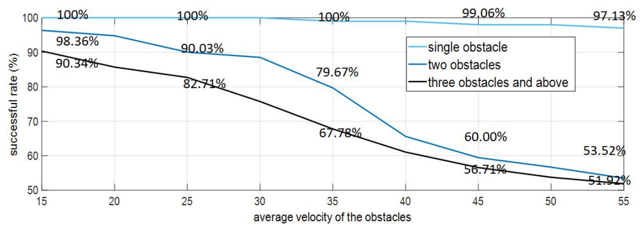


Fig. 14: FGA dynamic obstacle avoidance success rate with obstacle average velocity and number of obstacles.

Figure 14 shows Monte Carlo simulation results for average obstacle velocity and number of obstacles, for a fixed initial distance. In this simulation set the initial distance is fixed to be 300 meters and the velocity of

UAV is 15m/s. It is clear that when the obstacles' average velocity increases, the successful avoidance rate goes down quickly. When the average velocity of obstacles are smaller than 30m/s (twice the UAV speed), the successful avoidance rate for two obstacles remains near a 90%. If the obstacle velocity goes up to 45-60m/s (3-4 times of the UAV's speed), the successful avoidance rate decreases to about 60%. It is compatible with the results from bound 2. In addition, for a fixed number of obstacles, there is a strong correlation between the failure rate and velocity of the obstacles, as shown in Figure 15.

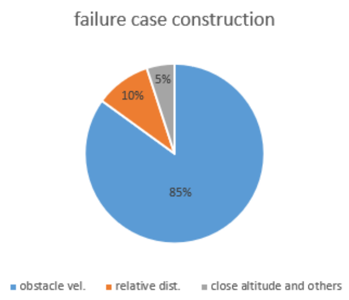


Fig. 15: Factors cause failure rate for dynamic obstacles

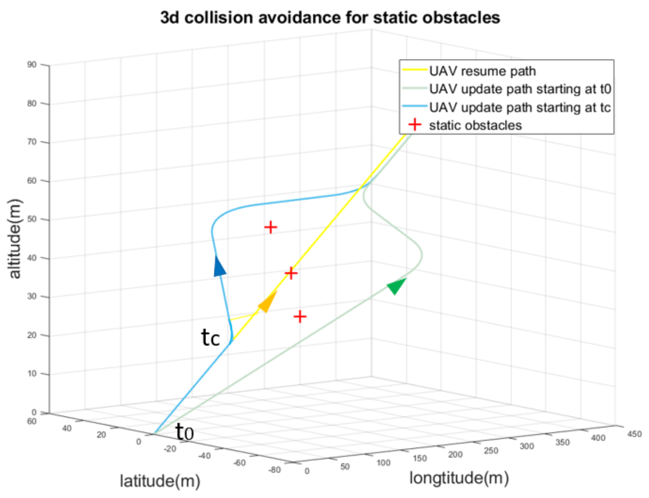


Fig. 16: 3D avoidance for static obstacles

The following two figures show obstacle avoidance turns in 3D using cost functions, as described in Section 3.5. Figure 16 shows that the UAV avoids a single obstacle successfully while causing minimal deviations to its original heading. Figure 17 shows that the UAV avoids multiple obstacles successfully, and returns to its original path. These two figures also show avoidance maneuvers that begin at t_0 and t_c. Table 7 shows the cost function used to generate the two 3D turns shown in Figure 17. In this case, a pair (ψ, θ) was selected for each turn. Each entry in this table is a non-dimensional value that represents each angle pair.

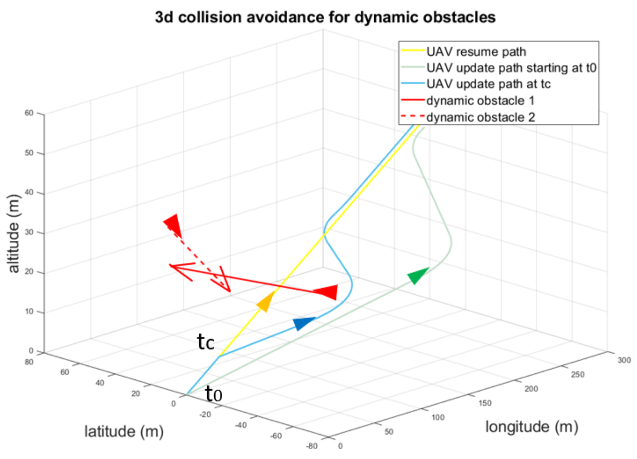


Fig. 17: 3D avoidance for dynamic obstacles

Table 7: Use cost function to determine command direction in 3D

	θ_1	θ_2	...	θ_j	...	θ_n
ψ_1	0.4137	0.3987	...	0.1863	...	0.2313
ψ_2	0.3937	0.3787	...	0.1663	...	0.2113
...	...	...	...	...	...	...
ψ_i	0.3737	0.3587	...	0.1463	...	0.1913
...	...	...	...	...	...	...
ψ_n	0.4337	0.4187	...	0.2063	...	0.2513

In summary, FGA provides successful collision avoidance for the UAV while shortening computation time and path length, with consideration of the minimum turning radius and the nominal speed, which is set to be constant.

5 Implementation in SITL Simulator and Fixed-Wing Aircraft

FGA was implemented in a Software-In-The-Loop fixed-wing vehicle simulator, as well as in hardware on board a fixed-wing UAV aircraft. The chosen test maneuver corresponds to one triggered at the critical avoidance time, where the aircraft goes through a rapid emergency maneuver. A description of each setup and results are described in this section.

5.1 Software In The Loop (SITL) Simulator

5.1.1 Ardupilot

The software in the loop simulation was performed using Ardupilot [67], an open source autopilot software for unmanned vehicles. It is an advanced autopilot software and is widely used by the UAS research community. Ardupilot includes functionality for rovers, boats and submersible vehicles, multi-rotors and fixed wing UAVs. ArduPilot is written in C++ to maximize efficiency when running on the on-board hardware of the vehicle, which is normally limited in regard to computation power. It has a comprehensive range of autonomous functions and contains the necessary dynamic controllers, navigation algorithms and hardware interfaces, among other features, for functional operation of unmanned vehicles, both in hardware and in simulation.

Ardupilot allows for software in the loop (SITL) simulations with custom flight dynamic models (FDM) based on vehicle kinematics. Simulated flights with the full ArduPilot software can be conducted without testing on a real world vehicle. In SITL, the sensor data is generated by the FDM and environmental variables, which can be specified into the simulator. For our simulations, SITL has been implemented in Linux, but it can also run in Windows. Figure 18 shows the architecture of the SITL setup. The main variables in the architecture are shown in Table 8.

The use of an autopilot software is typically paired with a ground control station (GCS) so users can observe vehicle states and control the vehicle from the ground. Small unmanned aerial vehicles normally communicate with a GCS using the Micro Aerial Vehicle link (MAVlink) protocol [68]. This facilitates a standard of communication between a wide array of different vehicles and ground stations, which can be used to extract information on the state of vehicle or its sensors, while simultaneously allowing commands to be uploaded. In our simulations, MAVProxy was used as a ground control station due to it's compatibility with Linux and flexibility of use.

Expensive computations can be carried off-board in the GCS, however this may induce latency and communication loss. Therefore having on-board computations for collision avoidance are preferred. FGA is computationally light and can be easily executed by the avionics interface on-board the aircraft.

5.1.2 DroneKit

Implementation of the FGA algorithm in SITL was done through a Python API known as DroneKit. It allows users to implement standalone Python scripts that can communicate with the vehicle by sending out MAVLink commands. Python provides ease of use when manipulating the MAVlink messages as opposed to directly editing the source code of ArduPilot itself, which is more involved. In the hardware implementation, the DroneKit script runs on-board the actual vehicle.

An additional Python script was used for translating keyboard inputs to PWM values which go to the control surfaces of the aircraft. This was done in order to simulate test flights and to ensure safe and reliable transitions between manual control to autonomous control. This script was run in parallel to the Python script that controls the FGA avoidance maneuver. A third script which was also run in parallel, simulated an ADS-B vehicle and was used for demonstrating dynamic obstacle avoidance features of FGA in the SITL simulator.

Figure 19 shows a screen of the SITL simulator ground control station (GCS). In this graph, the UAV is performing an avoidance maneuver to avoid a static obstacle. A SITL simulator video is provided in the supplemental materials. The simulator allows for a vehicle trace to follow the vehicle as it flies in the avoidance

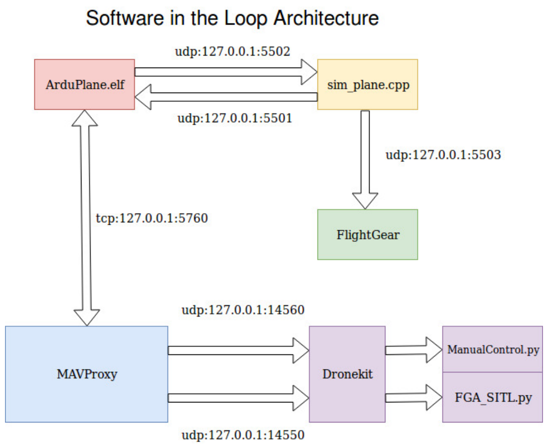


Fig. 18: SITL Architecture

Table 8: SITL architecture components

SITL Component	Description
ArduPlane.elf	The ArduPilot autopilot program
Sim Plane	FDM, provided alongside ArduPilot for SITL
FlightGear	Chosen 3D visualization program
MAVProxy	Chosen ground control station
DroneKit Scripts	Python API to provide the autonomous logic

maneuver path. Figure 20 shows a screen of the visualization software used in the SITL runs.

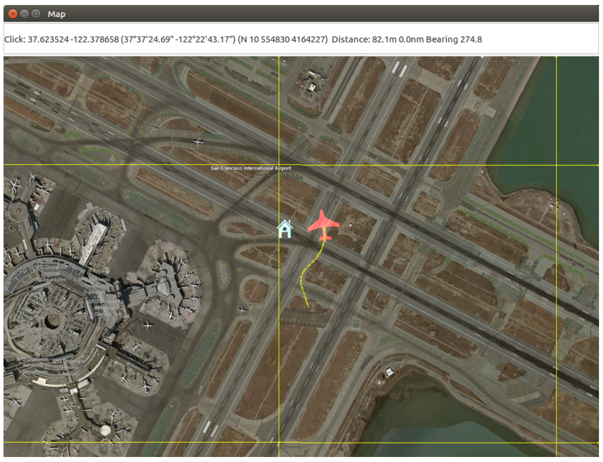

Map
Click: 37.623524 -122.378658 (37°37'24.69" -122°22'43.17") (N 10 554830 4164227) Distance: 82.1m 0.0nm Bearing 274.8
San Francisco International Airport

Fig. 19: SITL simulator: Obstacle avoidance of static obstacle

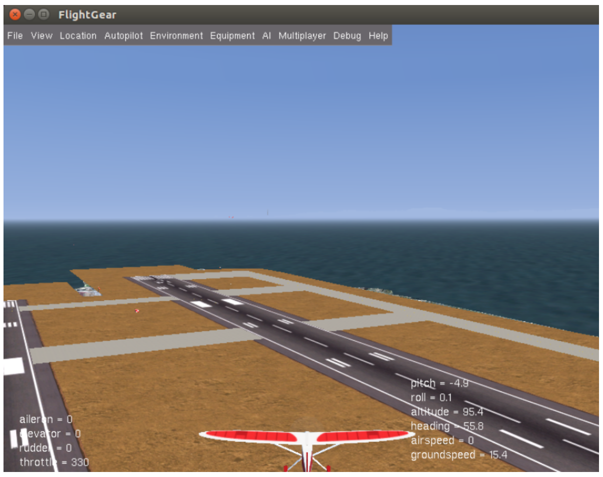


Fig. 20: SITL with FlightGear visualization

5.2 Hardware Implementation

An off the shelf aircraft, RC Apprentice 15e, was instrumented with a NAVIO2 board, telemetry and GPS. The FGA was implemented via DroneKit/MavLink in the NAVIO2 board which runs the Ardupilot flight software. The hardware architecture is described in Figure 21.

The flight software can be used and configured in several flight modes to various degrees of autonomy, e.g. manual, fly-by-wire, etc. One of these modes was customized with the FGA algorithm. The aircraft was flown to a certain safe altitude on manual control. Then, upon detection of a static virtual obstacle, it triggered the FGA avoidance maneuver, after which manual control was regained. The customized Apprentice RC aircraft used in flight testing is shown in Figure 22.

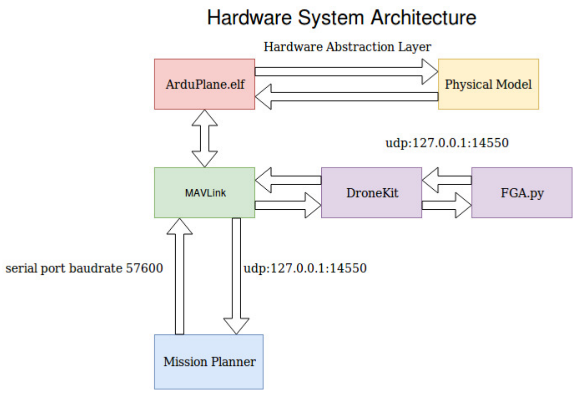


Fig. 21: Hardware Implementation Architecture

As part of the test setup, the virtual obstacle had pre-determined GPS coordinates and a specified safe distance d_m, centered at the location of the obstacle. These values were determined on-site. Based on a physical maximum bank angle and associated turn radius, the ailerons were commanded to follow a desired turning maneuver.

The FGA algorithm which was implemented in the NAVIO2 board is shown in Algorithm 4.

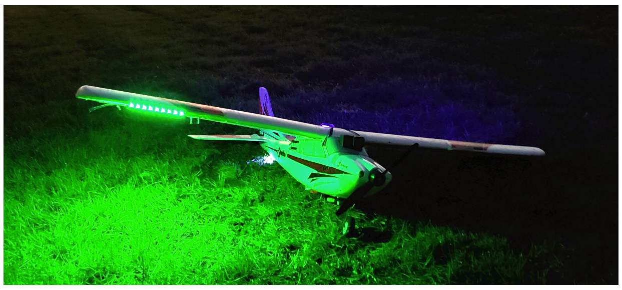


Fig. 22: Hardware Implementation: Apprentice Aircraft

5.3 Test algorithm

The test algorithm depicts an emergency maneuver where the UAV turns at its maximum deviation to avoid the protected area. The algorithm implementation makes use of the dynamic control that ArduPilot provides with its flight modes. The algorithm is triggered when the

Algorithm 4 Implementation of FGA algorithm

```
 1: function FGA(uav,obs)    ▷ The ownship and obstacle
 2:     while uav.mode $\neq$ CRUISE do
 3:         <wait>
 4:     end while

 5:     if objectDetected then
 6:         $P_{xy} \leftarrow$ PositionalAngle(uav.pos, obs.pos)
 7:         $\theta_{t0} \leftarrow P_{xy} -$ uav.heading
 8:         ▷ uav.heading: current vehicle heading
 9:         $r \leftarrow$ norm(uav.pos, obs.pos)
10:         ▷ 2D distance to the obstacle
11:
12:         if $|r \cdot sin\theta_{t0}| <$ obs.dm then
13:             ▷ Collision projected
14:             $\theta_i \leftarrow$ uav.heading
15:             ▷ Initial heading before deviation
16:             $\rho_{calc} \leftarrow$ TurnRadius(uav.vel, uav.bankAngle)
17:             $d_c \leftarrow \sqrt{(\texttt{obs.dm} + \rho)^2 + (r \cdot sin\theta_{t0} + \rho)^2)}$
18:             ▷ Critical distance
19:             $t_c \leftarrow \frac{d_c}{\texttt{uav.vel}}$
20:             ▷ Critical Time
21:             $\alpha \leftarrow cos^{-1}(\frac{\rho + r \cdot sin|\theta_{t0}|}{\rho + \texttt{obs.dm}})$
22:             ▷ Max Deviation Angle
23:             $t \leftarrow$ startTimer()

24:             while norm(uav.pos,obs.pos) $\leq d_c$
25:                                  and $t \geq t_c$ do
26:                 <wait>
27:                 ▷ Waiting on either $t_c$ or $d_c$ to be met
28:             end while

29:             uav.mode $\leftarrow$ FBWB
30:             ▷ Setting vehicle to FBWB mode

31:             dir $\leftarrow$ sign(uav.heading − PosAngle(uav.pos, obs.pos))
32:             Turn(uav, obs, $\theta_i, \alpha, 1500 + dir \cdot 500$)

33:             ▷ Turning away from obstacle until tangent

34:             pwm $\leftarrow$ PWMReq(uav.vel, obs.dm, dir)
35:             Turn(uav, obs, $\theta_i, \alpha$, pwm)

36:             ▷ Turning along obstacle perimeter

37:             Turn(uav,obs, $\theta_i, 0, 1500 +$ dir $\cdot 500$)

38:             ▷ Passed obstacle, turn to initial heading

39:             uav.returnRoll()
40:             ▷ Sets the aircraft roll back to 0
41:             vehicle.mode $\leftarrow$ CRUISE
42:             while vehicle.mode $\neq$ CRUISE do
43:                 <wait>
44:                 ▷ Continues in CRUISE mode
45:                 ▷ Breaks if otherwise specified
46:             end while
47:         end if
48:     end if
49: end function
```

Aircraft is switched to CRUISE mode; a flight mode that holds altitude and heading. This is to simulate the UAV maintaining its heading while en-route to its next waypoint. It calculates whether a collision is projected and subsequently computes its expected turn radius based off of Equation 7.

Critical distance, critical time and deviation angle are calculated. Then the script simultaneously checks for whether the critical distance has been met through GPS or if a previously set timer, has reached the critical time. The decision control on triggering FGA is shown in Figure 23. If any one of these conditions are met, then the avoidance maneuver begins. While ideally the vehicle maintains constant velocity, there is always the possibility for the physical model to undergo small changes or estimation errors. The on-board timer also mitigates low sampling time errors in the GPS, minimizing the chance of the maneuver beginning too late.

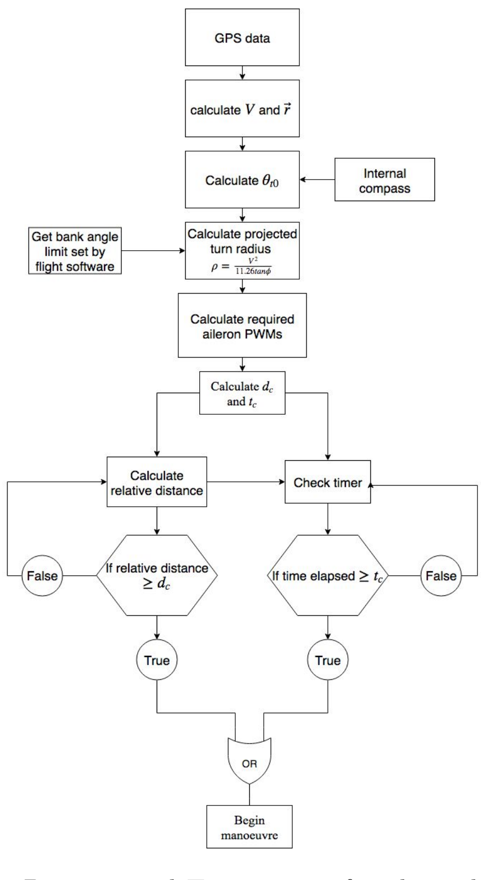

GPS data
calculate V and $\vec{r}$
Calculate θ_{t0}
Internal compass
Get bank angle limit set by flight software
Calculate projected turn radius $\rho = \frac{V^2}{11.26tan\phi}$
Calculate required aileron PWMs
Calculate d_c and t_c
Calculate relative distance
Check timer
If relative distance $\geq d_c$
If time elapsed $\geq t_c$
False
False
True
True
OR
Begin manoeuvre

Fig. 23: Distance and Time trigger for physical implementation

The max deviation angle α, as shown in Figure 24, is used as reference so the aircraft knows when to turn away, when to turn along the obstacle and when to turn away again back to its original path. From Figure 24, the α calculated from the geometry will be the same angle as the deviation from its initial heading.

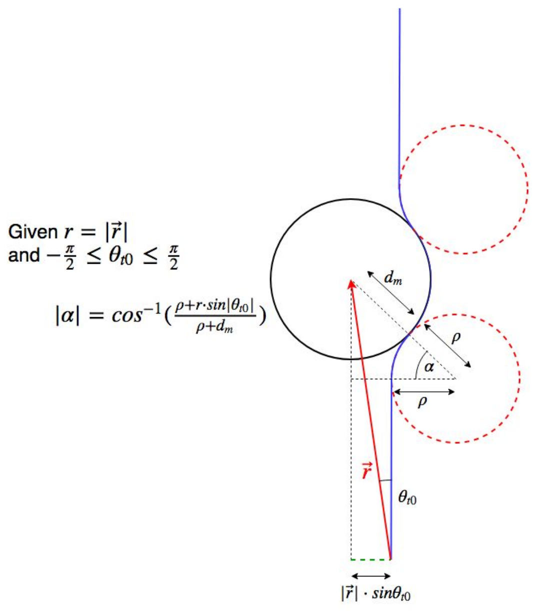

Given $r = |\vec{r}|$
and $-\frac{\pi}{2} \le \theta_{t0} \le \frac{\pi}{2}$
$|\alpha| = cos^{-1}(\frac{\rho + r\cdot sin|\theta_{t0}|}{\rho + d_m})$
d_m
ρ
α
ρ
$\vec{r}$
θ_{t0}
$|\vec{r}| \cdot sin\theta_{t0}$

Fig. 24: Critical avoidance maneuver implementation of test algorithm for SITL and Apprentice RC aircraft

The vehicle is subsequently put in to fly-by-wire B (FBWB) mode, this mode retains altitude control however allows the pilot to control the roll angle from the RC controller. ArduPilot interprets the PWM values that are being sent to the ailerons and converts it to a roll angle input which is passed through a controller. The PWM of the ailerons varies from $1000 - 2000$, and the maximum bank angle is a parameter that can be set in the ArduPilot software.

The relationship between input PWM and desired roll angle is linear, where a PWM value of 1500 corresponds to a roll of 0°. PWM values of 1000 and 2000 correspond to maximum left roll and right roll, respectively. The relationship can be calculated as:

$$PWM_{req} = 1500 + 500 \cdot \frac{dir \cdot Roll_{req}}{\phi_{uav.max}}, \tag{59}$$

where dir is the turn direction and is calculated as the sign of θ_{t0} and is either 1 or -1 representing anti-clockwise turn and clockwise turn respectively.

The vehicle then turns at its minimum turn radius and when the difference between its heading and its approach heading, θ_i, reaches the max deviation angle, α it starts turning the opposite direction. This involves turning around the obstacle at the same turn radius as the avoidance radius and therefore a PWM input must be calculated to achieve this. Taking Equation 7 and rearranging it to:

$$\phi = tan^{-1}(\frac{V^2}{11.26\rho}). \tag{60}$$

We can then substitute this bank angle in to equation 59 to obtain the required PWM value.

5.4 Results

The simulation was set up over San Francisco International Airport and a virtual static obstacle with a $d_m = 150m$ set in the centre of the runway. The UAV was made to take off and go to waypoints whose paths crossed through the obstacle's avoidance radius from multiple angles. FGA successfully allowed for the UAV to avert collisions with the protected area, as shown in Figure 25 and Figure 26 .

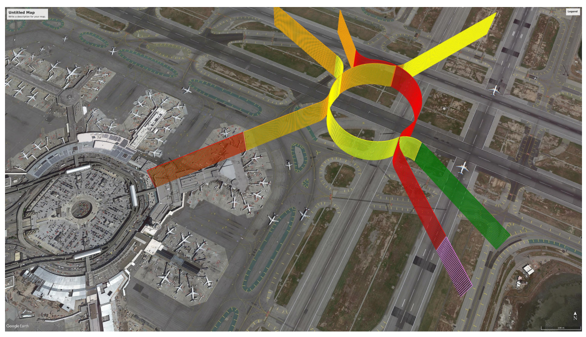


Fig. 25: Critical avoidance SITL implementation: Flight paths and FGA avoidance, Google Earth view.

Flight test results are shown in Figure 27 for three flights of the Apprentice aircraft in emergency avoidance maneuvers. The aircraft averted collisions with a virtual static obstacle with a protected radius.

6 Conclusion

The FGA algorithm was tested in Matlab simulations, SITL aircraft simulator, and in hardware on the Navio2 board which controlled the Apprentice RC aircraft. From these software and hardware implementations it can be concluded that FGA is suitable for fast obstacle avoidance. Matlab simulations showed that FGA is efficient at dealing with multiple obstacles. Its short computation time, selective start time and small deviation from the UAV initial path make FGA applicable to real world scenarios.

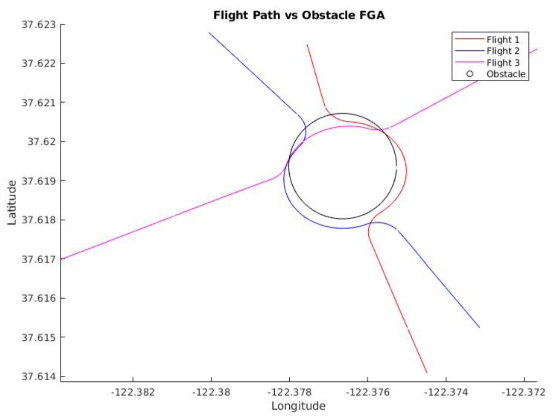
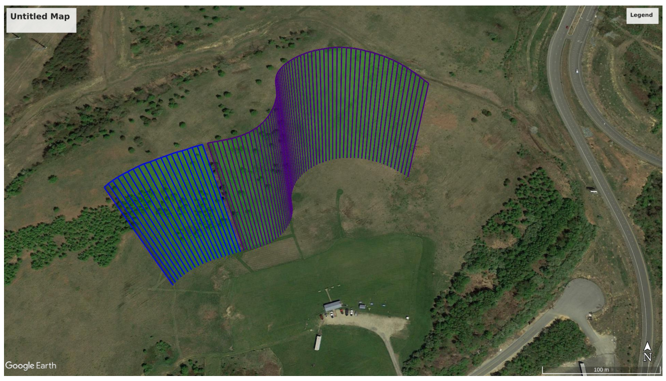


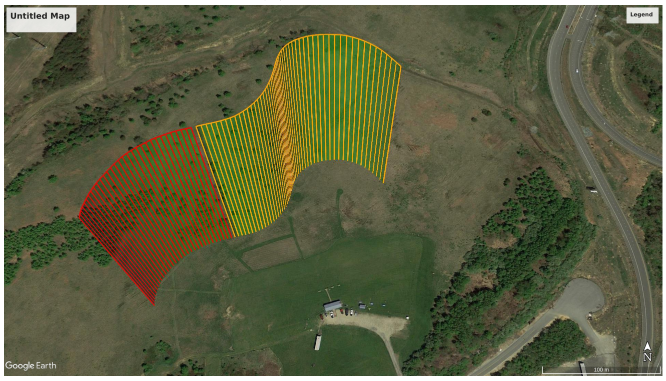
Fig. 26: Critical avoidance SITL implementation: Flight paths and avoidance maneuvers around protected disk centered at obstacle

In a crowded environment with different obstacles, depending on the critical avoidance time t_c calculated, FGA could divide these obstacles into different threat levels and let UAV to take proper action, such as avoid them in sequence instead of simultaneously, This increases the collision avoidance success rate. The avoidance time mechanism in FGA also reduces the deviation trajectory for a UAV from its initial path, limiting the possibility of it trespassing into the paths of other moving air vehicles, which benefits the safety of the airspace.

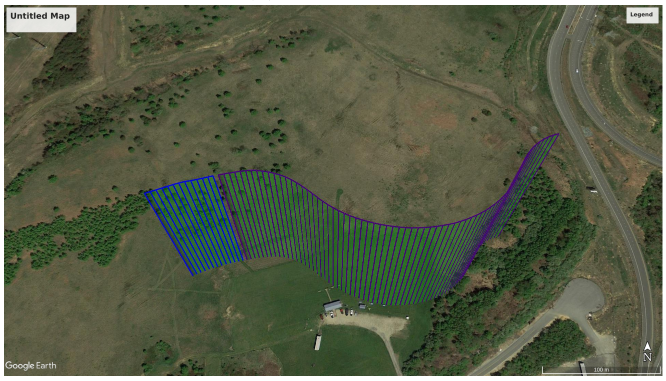
Future work includes flights of the Apprentice along a multiple obstacle course, as well as among moving virtual obstacles. A further step towards demonstrating NAS integration feasibility would consist of using ADS-B enabled aircraft as obstacles, for demonstration of FGA's efficient avoidance.



(a) First flight



(b) Second flight



(c) Third flight

Fig. 27: FGA collision avoidance test flights with Apprentice RC aircraft

Acknowledgment

The authors would like to thank the Maryland Industrial Partnerships (MIPS) Program and Millennium Engineering and Integration, Co. for their support of this project. We would also like to thank Patrick Fox, Henry Tuit Farquhar for implementation of autonomy hardware and for piloting the Apprentice aircraft. Thanks as well go to Camilo Melnyk who also implemented hardware on the Apprentice and for piloting the Phantom II quadcopter for videography as well as for management of the ground control station. Thanks to Sharon Shallom who also helped to manage the ground control station.

References

1. H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)," *Biosystems engineering*, vol. 108, no. 2, pp. 174–190, 2011.
2. K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl, "On unmanned aircraft systems issues, challenges and operational restrictions preventing integration into the national airspace system," *Progress in Aerospace Sciences*, vol. 44, no. 7-8, pp. 503–519, 2008.
3. R. Clarke and L. B. Moses, "The regulation of civilian drones' impacts on public safety," *Computer Law & Security Review*, vol. 30, no. 3, pp. 263–285, 2014.

4. P. Angelov, *Sense and avoid in UAS: research and applications*. John Wiley & Sons, 2012.
5. I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Physical review letters*, vol. 113, no. 23, p. 238701, 2014.
6. J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (uas s) the future of wildlife monitoring? a review of accomplishments and challenges," *Mammal Review*, vol. 45, no. 4, pp. 239–252, 2015.
7. S. Griffiths, J. Saunders, A. Curtis, B. Barber, T. McLain, and R. Beard, "Obstacle and terrain avoidance for miniature aerial vehicles," in *Advances in Unmanned Aerial Vehicles*. Springer, 2007, pp. 213–244.
8. R. J. Kephart and M. S. Braasch, "See-and-avoid comparison of performance in manned and remotely piloted aircraft," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 5, pp. 36–42, 2010.
9. C. von Essen and D. Giannakopoulou, "Analyzing the next generation airborne collision avoidance system," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014, pp. 620–635.
10. M. Huerta, "Integration of civil unmanned aircraft systems (uas) in the national airspace system (nas) roadmap," *Federal Aviation Administration, Retrieved Dec*, vol. 19, p. 2013, 2013.
11. S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
12. M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.
13. K. Al-Mutib, M. AlSulaiman, M. Emaduddin, H. Ramdane, and E. Mattar, "D* lite based real-time multi-agent path planning in dynamic environments," in *2011 Third International Conference on Computational Intelligence, Modelling & Simulation*. IEEE, 2011, pp. 170–174.
14. O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*. IEEE, 2013, pp. 1–8.
15. F. Lingelbach, "Path planning using probabilistic cell decomposition," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 467–472.
16. T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.
17. P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.
18. T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.
19. K. Sigurd and J. How, "Uav trajectory design using total field collision avoidance," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, p. 5728.
20. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.
21. Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 1398–1404.
22. J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.
23. B. I. Kazem, A. H. Hamad, and M. M. Mozael, "Modified vector field histogram with a neural network learning model for mobile robot path planning and obstacle avoidance." *Int. J. Adv. Comp. Techn.*, vol. 2, no. 5, pp. 166–173, 2010.
24. J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.
25. Y. Zhang and G. Wang, "An improved rgb-d vfh+ obstacle avoidance algorithm with sensor blindness assumptions," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2017, pp. 408–414.
26. G. Bianco and P. Fiorini, "Visual avoidance of moving obstacles based on vector field disturbances," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2704–2709.
27. Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 296–304.
28. M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.
29. P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-d trajectory planning of aerial vehicles using rrt," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1116–1123, 2017.
30. P. Liu, H. Yu, and S. Cang, "Optimized adaptive tracking control for an underactuated vibro-driven capsule system," *Nonlinear Dynamics*, vol. 94, no. 3, pp. 1803–1817, 2018.
31. ——, "Trajectory synthesis and optimization of an underactuated microrobotic system with dynamic constraints and couplings," *International Journal of Control, Automation and Systems*, vol. 16, no. 5, pp. 2373–2383, 2018.
32. T. Yu, J. Tang, L. Bai, and S. Lao, "Collision avoidance for cooperative uavs with rolling optimization algorithm based on predictive state space," *Applied Sciences*, vol. 7, no. 4, p. 329, 2017.
33. P. Yang, K. Tang, J. A. Lozano, and X. Cao, "Path planning for single unmanned aerial vehicle by separately evolving waypoints," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1130–1146, 2015.
34. D.-S. Jang, H.-J. Chae, and H.-L. Choi, "Optimal control-based uav path planning with dynamically-constrained tsp with neighborhoods," *arXiv preprint arXiv:1612.06008*, 2016.
35. D. Bozhinoski, A. Bucchiarone, I. Malavolta, A. Marconi, and P. Pelliccione, "Leveraging collective run-time adaptation for uav-based systems," in *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*. IEEE, 2016, pp. 214–221.
36. H. Jing-Lin, S. Xiu-Xia, L. Ri, D. Xiong-Feng, and L. Mao-Long, "Uav real-time route planning based on multi-optimized rrt algorithm," in *Control And Decision Conference (CCDC), 2017 29th Chinese*. IEEE, 2017, pp. 837–842.

37. D. Pascarella, S. Venticinque, and R. Aversa, "Autonomic agents for real time uav mission planning," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE, 2013, pp. 410–415.
38. A. Mujumdar and R. Padhi, "Reactive collision avoidance of using nonlinear geometric and differential geometric guidance," *Journal of guidance, control, and dynamics*, vol. 34, no. 1, pp. 303–311, 2011.
39. J. Minguez and L. Montano, "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.
40. S. S. Ge, X. Lai, and A. Mamun, "Boundary following and globally convergent path planning using instant goals," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 2, pp. 240–254, 2005.
41. I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, 1997.
42. K. Al-Mutib and F. Abdessemed, "Indoor mobile robot navigation in unknown environment using fuzzy logic based behaviors," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 2, pp. 327–337, 2017.
43. V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.
44. D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
45. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.
46. S. J. Guy, J. Van Den Berg, M. C. Lin, and D. Manocha, "Geometric methods for multi-agent collision avoidance," in *Proceedings of the twenty-sixth annual symposium on Computational geometry*. ACM, 2010, pp. 115–116.
47. A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.
48. H. Pham, S. A. Smolka, S. D. Stoller, D. Phan, and J. Yang, "A survey on unmanned aerial vehicle collision avoidance systems," *arXiv preprint arXiv:1508.07723*, 2015.
49. J. Seo, Y. Kim, and A. Tsourdos, "Differential geometry based collision avoidance guidance for multiple uavs," *IFAC Proceedings Volumes*, vol. 46, no. 19, pp. 113–118, 2013.
50. J. Haugen and L. Imsland, "Monitoring moving objects using aerial mobile sensors," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 475–486, 2016.
51. C. Luo, S. I. McClean, G. Parr, L. Teacy, and R. De Nardi, "Uav position estimation and collision avoidance using the extended kalman filter," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 6, pp. 2749–2762, 2013.
52. S. Saha, A. Natraj, and S. Waharte, "A real-time monocular vision-based frontal obstacle detection and avoidance for low cost uavs in gps denied environment," in *Aerospace Electronics and Remote Sensing Technology (ICARES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 189–195.
53. Y. Kwang and Y. Kwang, "Performance simulation of radar sensor based obstacle detection and collision avoidance for smart uav," in *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*, vol. 2. IEEE, 2005, pp. 10–pp.
54. J.-W. Park, H.-D. Oh, and M.-J. Tahk, "Uav conflict detection and resolution based on geometric approach," *International Journal of Aeronautical and Space Sciences*, vol. 10, no. 1, pp. 37–45, 2009.
55. Z. Daniels, L. Wright, J. Holt, and S. Biaz, "Collision avoidance of multiple uas using a collision cone-based cost function," *Computer Science and Software Engineering Department, Auburn University, Tech. Rep. CSSE12-07*, 2012.
56. Y. Watanabe, A. Calise, E. Johnson, and J. Evers, "Minimum-effort guidance for vision-based collision avoidance," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2006, p. 6641.
57. R. A. Sasongko, S. Rawikara, and H. J. Tampubolon, "Uav obstacle avoidance algorithm based on ellipsoid geometry," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 567–581, 2017.
58. X. Yang, L. M. Alvarez, and T. Bruggemann, "A 3d collision avoidance strategy for uavs in a non-cooperative environment," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 315–327, 2013.
59. C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo, "A novel 3d geometric algorithm for aircraft autonomous collision avoidance," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1580–1585.
60. K. Bilimoria, "A geometric optimization approach to aircraft conflict resolution," in *18th Applied Aerodynamics Conference*, 2000, p. 4265.
61. C. Yoo, A. Cho, B. Park, Y. Kang, S. Shim, and I. Lee, "Ads-b hils test for collision avoidance of smart uav," in *2011 Tyrrhenian International Workshop on Digital Communications-Enhanced Surveillance of Aircraft and Vehicles*. IEEE, 2011, pp. 253–257.
62. H.-S. Shin, A. Tsourdos, and B. White, "Uas conflict detection and resolution using differential geometry concepts," *Sense and avoid in UAS: Research and applications*, vol. 62, 2012.
63. J. Goss, R. Rajvanshi, and K. Subbarao, "Aircraft conflict detection and resolution using mixed geometric and collision cone approaches," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4879.
64. D. Yang, D. Li, and H. Sun, "2d dubins path in environments with obstacle," *Mathematical Problems in Engineering*, vol. 2013, 2013.
65. Z. Lin, L. Castano, and H. Xu, "A fast obstacle collision avoidance algorithm for fixed wing uas," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 559–568.
66. Q. Wang and J. Zhang, "Mpc and tgfc for uav real-time route planning," in *Control Conference (CCC), 2017 36th Chinese*. IEEE, 2017, pp. 6847–6850.
67. A. D. Team, "Ardupilot autopilot suite," *URL http://ardupilot. com/. Accessed*, pp. 05–20, 2016.
68. L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga *et al.*, "Mavlink: Micro air vehicle communication protocol," *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]*, 2013.