Simulation of Autonomous UAV Navigation with Collision Avoidance and Space Awareness

1st Jian Li
School of Aerospace,
Transport and Manufacturing
Cranfield University
Bedford, UK
jian.li@cranfield.ac.uk

2nd Hongmei He
School of Aerospace,
Transport and Manufacturing
Cranfield University
Bedford, UK
h.he@cranfield.ac.uk

3rd Ashutosh Tiwari
Department of Automatic Control
and Systems Engineering
University of Sheffield
Sheffield, UK
a.tiwari@sheffield.ac.uk

***Abstract*—This research developed a safe navigation system of an autonomous UAV within a comprehensive simulation framework. The navigation system can find a collision-free trajectory to a randomly assigned 3D target position without any prior map information. It contains four main components: mapping, localisation, cognition and control, where the cognition system makes execution command based on the perceived position information about obstacles and the UAV itself from mapping and localisation system respectively. The control system is responsible for executing the input command made from the cognition system. Three case studies for real-life scenarios, such as restricted area avoidance, static obstacle avoidance and dynamic obstacles, are conducted. The experiments demonstrate that the UAV is capable of determining a collision-free trajectory under all three cases of environments. All simulated components are designed to match their real-world counterparts' dynamics and properties. Ideally, the simulated navigation framework can be transferred to a real UAV without any changes. As the navigation system of a drone is implemented in a modular way, it is easier to test and validate to ensure the system's performance. Moreover, the system has good readability, maintainability and extendability. Hence, the simulation framework provides a good platform for future robotic research.**

***Index Terms*—Cognition, control system, dynamic obstacle avoidance, localisation, mapping, offline navigation, safe navigation, space awareness, static obstacle avoidance.**

I. Introduction

The ongoing trend towards the integration of robotics and UAV(Unmanned Aerial Vehicle) systems has attracted significant interest in a wide range of applications and became a major research topic in recent years. Compared to manned aircraft, UAVs were originally used for missions that are "dull, dirty or dangerous" for humans [1]. In December 2016, Amazon had completed the first trial of its futuristic UAV-based delivery plan, to utilise UAV technology to fly individual packages autonomously to customers. More recently, in the opening ceremony of the 2018 PyeongChang Winter Olympics, Intel used more than 1200 purpose-built UAVs, which feature built-in LED lights, to fly simultaneously and formed a larger-than-life choreographed snowboarder. It transformed UAV technology into an entirely new form of entertainment and created a memorial experience at one of the most-watched events in the world.

However, most of the deployed automation systems utilise the online navigation solution, which only works with on-board communication unit [2], [3]. For instance, Intel uses the traditional centralised solution to control the UAVs by sending command according to the feedback flight status from UAVs. Although Intel can control over 1200 UAVs with the assistance of sophisticated programme by a single pilot, It will be more challenging to keep track each UAV within the increasingly crowded airspace and jeopardise the flight mission when there are only limited signals [4], [5]. On the other hand, most of the available offline navigation solutions have a minimal perception of the working area, which only operates within known or partially known environment [6]. It can be tedious and impractical to get access to a full map of all possible working area. Hence, the system needs to be capable of performing navigation without any prior map data.

Furthermore, in order to test algorithms on UAVs, it costs the researchers both financially and timely to get access to expensive hardware and to train safety-pilots. Most of the available navigation solutions are tested as stand-alone scenarios and sub-systems rather than a complete navigation system. Additionally, Most of the behaviours occurring on real UAVs are hard to replicate, and often cause harm to UAV's safety. This motivates the revolution to simulate the behaviours of UAVs. A well-implemented simulator should provide accurate graphical and physical simulation to emulate the real-world with appropriate models and dynamics. Furthermore, the simulator should provide the necessary tools to behave in a similar manner of its real-world counterparts; ideally, the simulated navigation system can be implanted to a real UAV with minimum efforts, thus to provide a platform for easy development of algorithms to implement trustworthy UAV systems.

A well-implemented simulation system can facilitate the development of the navigation system. Therefore, we develop a simulation framework of offline autonomous navigation system for UAVs, regarding three essential cases: space awareness, static obstacle avoidance and dynamic obstacle avoidance.

- For space awareness, the UAV should be able to avoid restricted areas, defined by regulators. It should fly along with the restricted areas by searching the shortest path regarding the UAVs current position, target position, and the geometric shape of the restricted area.

- For static obstacle avoidance, the UAV should be able to avoid static obstacles, which arbitrarily appear within an unknown 3D environment, while approaching to a randomly assigned target position.
- For dynamic obstacle avoidance, the UAV should be able to avoid a dynamic obstacle and plan its trajectory regarding the movement of the obstacle.

II. Simulation Framework

The simulation framework is developed based on Rotors simulator package developed by Autonomous Systems Lab at ETH Zurich [7], which is based on the integration of ROS (Robot Operating System) and Gazebo simulator. ROS provides libraries and tools to facilitate software development for the navigation system. Gazebo provides physics engine, graphics, programmatic and graphical interfaces. Here, Gazebo provides simulated models for UAV dynamics, sensors, as well as the working environment. The simulated components are designed to match their real-world counterparts on the right block (Fig. 1).

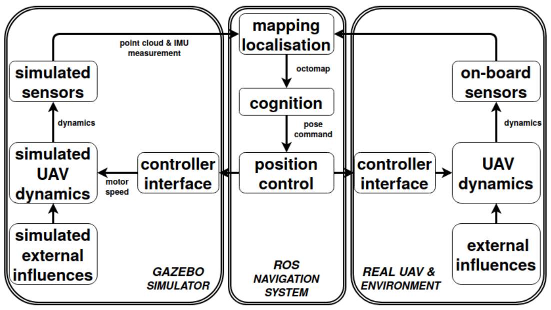


Fig. 1. Simulation framework overview

A. Navigation System Framework

Autonomous navigation is described as the process of generating a map representation of the UAV's immediate working environment, detecting potential hazard to the UAV's movement, and navigating within the immediate working environment to a target position [8]. Safe navigation of UAVs may be affected by various factors, such as communication and weather; even cyber-attacks could jeopardise the safety of UAV. However, all of these are out of the scope of the research. Fig. 2 shows the four interrelated components for the navigation framework. The cognition system takes the inputs from both the mapping and localisation systems; then produces execution command for the control system. This paper focuses on the development of high-level tasks for the cognition system.

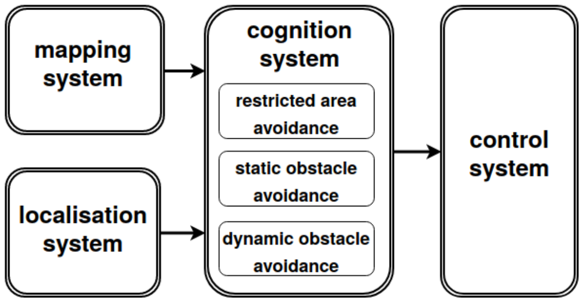

mapping system
localisation system
cognition system
restricted area avoidance
static obstacle avoidance
dynamic obstacle avoidance
control system

Fig. 2. Navigation system overview

1) Mapping: The degree of autonomous navigation systems is highly linked to their capability of mapping the environment. The mapping system should be able to distinguish between the free and occupied space of the immediate working environment; identify the moving obstacles and keep a record of their historical trajectories; update the map information to keep track of new entered obstacles; and if possible, predicting the moving obstacles' future position. For this paper, mapping is necessary for the following reasons: (1) to localise the UAV itself and the target position; (2) to plan a path between the UAV's current position and the target position; (3) to support UAV's collision avoidance.

2) Localisation: Localisation refers to the strategy to estimate the UAV's position and dynamics within the spatial map that occurs simultaneously during navigation, to assist the UAV in planning and executing movements, and building a correct map of the environment. It is the process of determining the UAV's position, velocity and orientation information related to the constructed map of the environment. Interestingly, the map building process requires knowledge of the UAV's position, and the localisation requires a map, which led to another important technique: Simultaneous Localisation And Mapping (SLAM), which intends to tackle two challenges. The first challenge is the map errors related to the real environment due to the limited range of observation or imperfection of the sensory system, and another is the dynamic environment modelling due to the change in the object's position or shape.

3) Cognition: In this paper, the cognition system is responsible for generating a trajectory towards a target position from a specific starting point, without colliding to any obstacles detected by the system, within minimum time. Moreover, in order to avoid the dynamic obstacle, the cognition system should be able to make the appropriate decision by predicting action into the future based on the characteristics of both the UAV and obstacle's current and potential behaviour. Lastly, the path planning system should also avoid some predefined regions that will not physically cause damage to the UAV, such as military restricted fly zone or any area that UAVs are prohibited.

4) Control: In autonomous UAV systems, the navigation system must implement the approach to controlling the vehicle in an uncertain environment throughout the mission [9]. It involves the process of combining the outputs of mapping, localisation, path planning, and translating them into actuator commands for UAVs' mobility and payload response [8]. Therefore, the UAV should have the capability of self-governance in the performance of the control function in a very unstructured environment. The specific behaviours will vary with the operational requirement. In this paper, the UAV should be able to perform the essential flight operations such as taxi, take-off, climb, cruise, glide, landing, etc.

III. Navigation System Design

A. Low-level Tasks

1) Mapping: The mapping system utilises a depth camera to convert the simulated environment into ROS-compatible point cloud data, and then converted into Octomap representation, an efficient probabilistic 3D mapping framework developed by Hornung et al. [10]. In Octomap, the environment is segmented into multiple spaces, and each space can be further segmented into eight sub-spaces with the same size. The segmentation process can be applied recursively until the map reaches the desired resolution to represent more complex parts of the environment.

2) Localisation: An ideal odometry sensor is used to mimic a generic on-board tracking system. The localisation system is responsible for detecting and publishing UAV's position, orientation, linear and angular velocity.

3) Cognition: Due to the inherent nature between space awareness, static obstacle avoidance and dynamic obstacle avoidance, the implementation of these high-level tasks is integrated into a cognition system.

4) Control: A geometric control approach proposed by Lee et al. [11] is chosen to directly calculate the thrust and moments required, with different types of commands, such as position, orientation, or angular rate.

B. Space Awareness

For restricted area avoidance, the UAV requires prior information about the size, position of the restricted area. Secondly, the UAV needs to monitor its position continually and determine an appropriate trajectory to the target without intruding into the restricted area. However, it is futile and a waste of computer resource to check for potential intrusion during the whole navigation process. A condition is given to the algorithm to ensure intrusion checking are only executed when the UAV is relatively close to the restricted area. Fig. 3 shows the flowchart of implementing the restricted area avoidance algorithm.

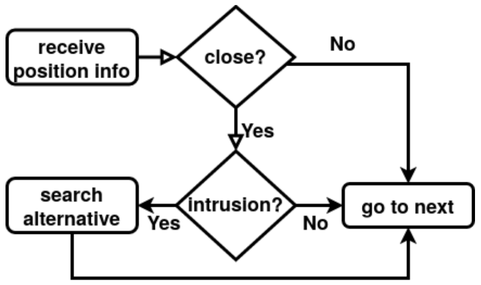
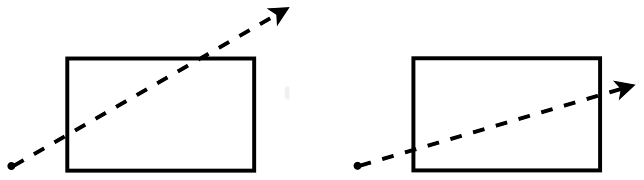


Fig. 3. Flowchart for restricted area avoidance

1) Intrusion Checking: In order to evaluate potential intrusion into the restricted area, a 2D rectangle bounding box was constructed, in terms of the coordinates of the specified restricted area. The length of the box is extended with two metres in each axis to compensate the system errors, and the distance travelled during intrusion checking stage. As the UAV will only start to check the restricted area when the UAV is approaching any one of the four restricted boundaries, it is safe to assume the trajectory will not intrude the restricted area when both the UAV and target position are located at the same side out of the box. Otherwise, the algorithm judges whether the UAV could potentially intrude the restricted area, in terms of the UAV's current position, next waypoint and the geometry of the bounding box.



(a) The trajectory intersects with two adjacent boundaries (b) The trajectory intersects with two parallel boundaries

Fig. 4. 2D overview of restricted area scenarios, where the dashed arrow represents the po

2) Exiting Strategy: There are two cases of how the UAV fly away from the restricted area, but towards the target, the trajectory could potentially intersect with either two adjacent or two parallel boundaries (Fig. 4). Assuming there is no obstacle around the restricted area, different exiting strategies will be applied to avoid the intrusion to a restricted area for these two cases. For two adjacent boundaries, the UAV will always choose to exit the rectangle by moving towards the intersection point between those two adjacent boundaries for a quick exit. Otherwise, the UAV will calculate the distance to each safe-corner and exit by the closer one. If the distances to the two corners are the same, the UAV will pick a corner randomly to fly away from the restricted area.

C. Static Obstacle Avoidance

As it is shown in the flowchart (Fig. 5), the implementation of static obstacle avoidance starts by searching for the next intermediate waypoint. A potential collision is evaluated with the Octomap data from the mapping system by constructing a

3D bounding box, with respect to the position of the UAV and the proposed intermediate waypoint. The task can be roughly broken down into three steps:

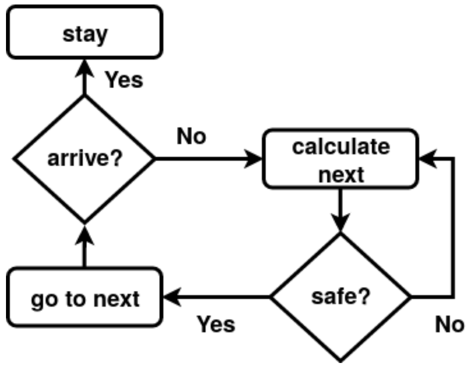


Fig. 5. Flowchart for static obstacle avoidance

1) Next Waypoint Calculation: The next waypoint is calculated based on the geometrical difference between the UAV's current position and the target position in each axis; the algorithm will increase or decrease with a configurable step size in the corresponding axis according to the difference.

2) Collision Checking: Instead of acquiring the exact position of the obstacle, the algorithm finds the necessary region for the UAV to pass through, which is the aforementioned bounding box. The box is constructed from the UAV current position to the proposed intermediate waypoint, with bidirectional extension in each axis to account for the system error, any occupied or unknown region within the bounding box triggers the algorithm search for an alternative collision-free waypoint.

3) Alternative Waypoint Searching: The default strategy of searching for an alternative waypoint is to look for free space above the UAV's current position. When the up space is occupied, the system will divide the UAV's near-space into five different bounding boxes ("front", "back", "left", "right", "down") and evaluate them in the order base on their distances to the target position.

D. Dynamic Obstacle Avoidance

For dynamic obstacle avoidance, the UAV is required to reach a randomly assigned target position by avoiding dynamic obstacle which the UAV has no prior knowledge of its movement. In a real-world scenario, the navigation system needs to classify the types of objects detected from the perception system, identify the dynamic objects by comparing the change of position between the UAV and obstacles. This process typically requires the mapping system to have segmentation and feature extractor algorithm [12], [13], which is beyond the scope of this paper. The dynamic obstacle avoidance is implemented separately as a stand-alone scenario without the segmentation and feature extractor process. One dynamic obstacle is implemented to test the algorithm's feasibility to work within a dynamic environment.

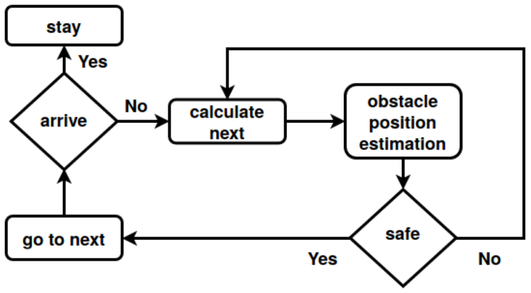


Fig. 6. Flowchart for dynamic obstacle avoidance

Fig. 6 shows the flowchart of dynamic obstacle avoidance implementation. It starts with calculating a waypoint towards the target position; then check for a potential collision based on the predicted range of obstacle's future position; Lastly, the algorithm will search for an alternative waypoint if the initial waypoint is dangerous. The implementation can be described in the following steps:

1) Collision Checking: With access to the obstacle's historical position, its velocity can be estimated by comparing its historical displacement at t second(s) ago. Hence, the potential collision can be determined by whether the UAV's trajectory intrudes the 3D bounding box constructed from obstacle's current position and predicted future position. Lastly, the algorithm will apply SAT (Separating Axis Theorem) to check whether the trajectory interferes with the interesting region. The theorem states that two polygons do not collide if it is possible to draw a line to separate them. An explanation can be found in [14].

2) Alternative Waypoint Searching: The algorithm searches for an alternative trajectory by assigning a waypoint towards one of the four corners of the constructed bounding box in 2D. The waypoint in the z-axis is assigned towards the opposite z-axis direction of the estimated obstacle's. The safe-corners can be determined by calculating the distance from UAV's current position to each of the four corners, where the shortest two are the safe options. For the exceptional cases when there are three safe-corners, the algorithm chooses randomly two of the shortest ones. Fig. 7 shows the eight possible scenarios of how the UAV locates outside of the bounding box in 2D, with the corresponding direction vectors to each safe-corner in both axes. For example, [++,++,++] indicates the UAV travels in a positive direction in both x and y-axis towards any of the three safe-corners. Lastly, the algorithm determines the appropriate corner based on the number of safe-corners and the relationship between the UAV's potential travel direction and the estimated moving obstacle's trajectory direction in each axis.

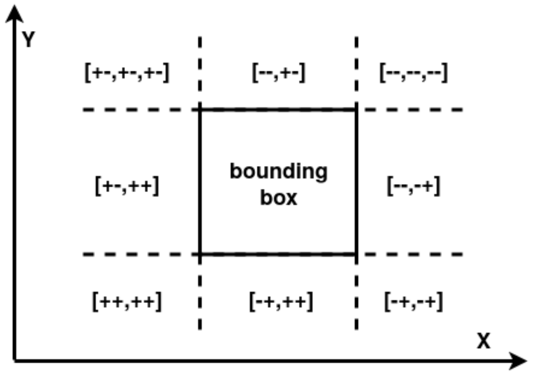
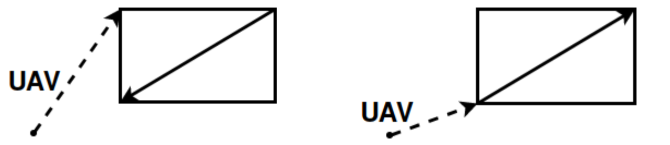


Fig. 7. 2D overview of potential intrusion scenarios

a) Three safe-corners: The algorithm compares the distance from the UAV towards these two safe-corners, the one with a shorter distance is chosen as the alternative waypoint, except for the cases when the corresponding trajectory towards the corner results in the UAV travelling in the opposite direction of the moving obstacle in both x and y-axis (Fig. 8a).

b) Two safe-corners: The algorithm determines the alternative waypoint depending on the number of opposite direction between the UAV's potential trajectory and the estimated obstacle's trajectory. The preferable corner could have two opposite directions; the less preferable choice is the one with only one opposite sign. In this way, it allows the UAV to travel in the opposite direction of the moving obstacle to avoid the potential collision.



(a) chose the longer distance corner to maximise safety when the obstacle is moving towards the UAV

(b) chose the shorter distance corner to avoid the obstacle

Fig. 8. Determine the safe-corner when there are three safe-corners, where the solid arrow represents the predicted obstacle's future trajectory, the dashed arrow represents the proposed UAV's trajectory

IV. Experiments and Evaluation

A. Space Awareness

1) Experiment Setup: Table I includes the experiments designed to test the algorithm's ability to avoid the restricted area, with four scenarios of how the potential trajectory intersects with the restricted area. Each scenario is given with a 3D target point and a restricted rectangle area defined by $x_{min}, x_{max}, y_{min}, y_{max}$, the UAV is expected to avoid the restricted area and reach the target point from the original point $(0, 0, 0)$. Cases 1.1 and 1.4 include the scenario that the potential trajectory intersects with two parallel boundaries, and the difference is the implemented shortest path exit strategy. In case 1.1, it is the equal distance for the UAV to avoid the area by taking any of the two corners. In case 1.4, the UAV is expected to calculate the distance from its current position towards each safe-corner, and take the short path to exit the restricted area. Case 1.2 covers the scenarios that the trajectory intersects with two adjacent boundaries, and the UAV is expected to avoid the area by approaching to the corner where those two boundaries intersect. Lastly, case 1.3 represents the scenario that the UAV is located on the boundary of the restricted area, and the potential trajectory will not intersect with the restricted area, the ideal result would be the UAV to keep a safe distance from the boundary.

TABLE I
Restricted area experiment scenarios

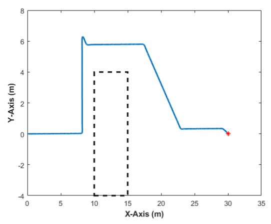
Case No	Target			Restricted Area			
	x	y	z	x_{min}	x_{max}	y_{min}	y_{max}
1.1	30	0	2	10	15	-4	4
1.2	10	3	2	5	15	-4	0
1.3	20	8	2	10	25	-4	4
1.4	25	3	2	10	15	-4	4

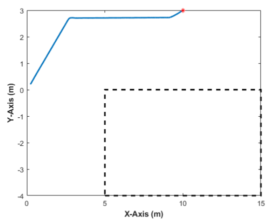
2) Results and Evaluation: As it is shown in Fig. 9, the restricted area is represented as the dashed line, the solid line represents the executed trajectory in 2D, and the target is marked with the asterisk sign. The UAV reached the target by finding the appropriate corner when the trajectory intersects with two adjacent boundaries (Fig. 9b), and kept a safe distance from the restricted area when the UAV is too close to the area (Fig. 9c). When the potential trajectory intersects with two parallel boundaries, the UAV reached the target by avoiding the restricted area with a shorter distance (Fig. 9d).

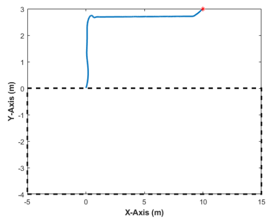
In Fig. 9a and 9d, the trajectory went over the safe distance when the UAV was trying to turn at approximately 8 in the x-axis, which is due to the implemented position controller restricts the initial attitude error less than 90 degrees to obtain the stability of the complete system. The properties of the position controller can be found in [11]. A possible solution to improve the controller instability is to introduce an intermediate waypoint with smaller turning angle when a sharp manoeuvre is required. However, this restricts the flexibility of the UAV's movement.

B. Static Obstacle Avoidance

1) Experiment Setup: For static obstacle avoidance, potential collision is evaluated by constructing a 3D axis-aligned bounding box from the UAV's current position to the proposed intermediate waypoint. Ideally, In an open space with fewer obstacles, large waypoint step size results in faster converging to the target position. However, in a crowded working environment, a small step size is needed to detect all obstacles without missing any obstacles. This potentially increase the converging time of the navigation. Secondly, higher map resolution gives a better representation of the complicated part of the working


(a) $Case1.1$: two parallel boundary with equal distance to safe-corners


(b) $Case1.2$: two adjacent boundary


(c) $Case1.3$: the trajectory does not intersect with the bounding box

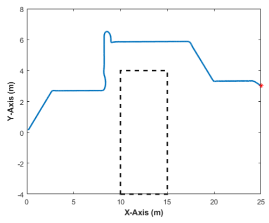

(d) $Case1.4$: two parallel boundaries with a shorter corner to exit

Fig. 9. Restricted Area results in 2D

environment but requires more computation capacity. Fig. 10 illustrats a 3D outdoor environment, the UAV is expected to find a collision-free trajectory to the user-defined target $(30, 10, 2)$ from $(0, 0, 0)$. Experiments are designed to evaluate how Octomap resolution and waypoint step size can affect the performance of the algorithm. Additionally, as the primary solution of searching for alternative waypoint is to 'look-up', a roofed environment is introduced to test the algorithm's feasibility.

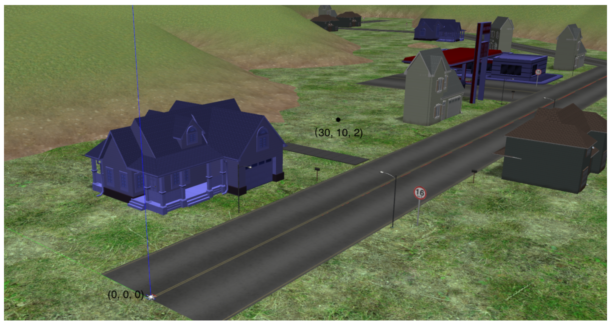
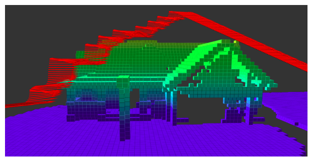


Fig. 10. 3D map with static obstacles and user-defined target

2) *Results and Evaluation:* Fig. 11 shows the executed trajectory for both outdoor and roofed environment. The trajectory is concluded to be collision-free as the UAV has successfully detected and avoided the surrounding obstacles during the navigation. The performance of the algorithm is evaluated by recording the total flight duration consumed for the UAV to reach the target position, with the results given in Table II.


(a) open space trajectory

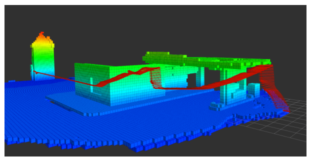

(b) roofed environment trajectory

Fig. 11. The executed trajectory in 3D

TABLE II
ALGORITHM PERFORMANCE WITH DIFFERENT OCTOMAP RESOLUTION AND NEXT WAYPOINT STEP SIZE

Case No	Waypoint step (m)	Map resolution (m)	Duration (s)
2.1	0.5	0.05	297
2.2	0.5	0.1	65
2.3	0.5	0.15	53
2.4	0.5	0.2	51
2.5	0.5	2	49
2.6	0.5	3	failed
3.1	0.1	0.5	failed
3.2	0.5	0.5	47
3.3	1	0.5	29
3.4	1.5	0.5	23

The results show that the algorithm's performance is improved significantly when the resolution changes from 0.05 to 0.1, a large number of iterations for obstacle detection takes much time with a low-resolution value (high resolution); hence, the duration has a big jump. The duration changes slightly when the resolution value increases from 0.1 to 2; the number of iterations for collision is reduced. However, in Case 2.6, it failed to find an initial waypoint at the beginning of the simulation. This is because the perception unit in mapping system detects the ground land as an obstacle, and marked the space as occupied with large resolution value. The size of the obstacle is larger than its actual size; hence no initial waypoint at the beginning of the simulation, even the UAV is in an open space. An ideal solution would be the improvement in the mapping system, to segment the ground land from the rest of the static obstacles.

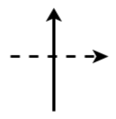
For the waypoint step size, the results show that a large step size improves on the converging time of the algorithm. However, in Case 3.1, the algorithm failed to find a trajectory when the step size is 0.1 metre. This is because the bounding box used for collision checking is constructed from UAV's current position towards the next waypoint with the length of step size; due to rounding and discretisation effects, nodes may be traversed that have float coordinates appearing outside of the float bounding box.

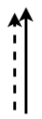
C. Dynamic Obstacle Avoidance

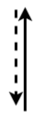
1) Experiment Setup: In order to determine the appropriate t for velocity estimation, the dynamic obstacle is simulated as a second UAV; The velocity of the dynamic obstacle is evaluated in terms of the global coordinates of the obstacle at the current time and last time as well as the time interval. From the result, the estimated maximum velocity is extremely larger than its true value when the duration is shorter than $0.035s$. A possible explanation would be the selected duration of t is shorter than the minimum time T_p required to process the velocity estimation, which results in the system failed to update the obstacle's position. On the other hand, the estimated velocity is smaller than the actual velocity when the t exceeds approximately $0.5s$. This is due to the obstacle is programmed to reach a sequence of intermediate waypoints rather travelling straight between two waypoints with a constant velocity, which involves acceleration and deacceleration. Larger t does not provide a good representation of the obstacle's instantaneous velocity. Hence, the duration of $0.04s$ is chosen for the obstacle's velocity estimation.

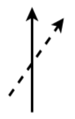
TABLE III
CASES DESIGNED FOR DYNAMIC OBSTACLE AVOIDANCE WITH START AND ENDING COORDINATES FOR BOTH THE UAV AND THE MOVING OBSTACLE

CASE *No.*	**UAV start** **x, y, z** (m)	**UAV target** **x, y, z** (m)	**Obstacle start** **x, y, z** (m)	**Obstacle target** **x, y, z** (m)
4.1	[0, -6, 2]	[0, 4, 2]	[-6, 0, 2]	[6, 0, 2]
4.2	[0, -6, 2]	[0, 10, 2]	[0, 0, 2]	[0, 6, 2]
4.3	[0, -6, 2]	[0, 10, 2]	[0, 6, 2]	[0, 0, 2]
4.4	[0, -10, 2]	[0, 10, 2]	[-3, -3, 2]	[3, 3, 2]
4.5	[0, -10, 2]	[0, 10, 2]	[3, 3, 2]	[-3, -3, 2]


(a) case 4.1


(b) case 4.2


(c) case 4.3


(d) case 4.4

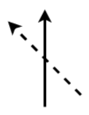
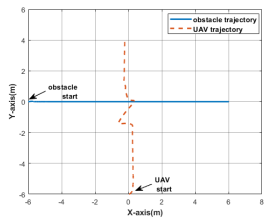

(e) case 4.5

Fig. 12. 2D overview of the obstacle's trajectory and UAV's potential trajectory, where the dashed arrow denotes the UAV's potential trajectory and solid arrow denotes the obstacle's trajectory

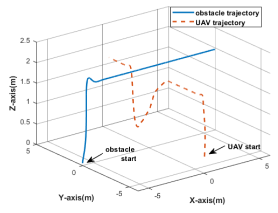
Fig. 12 shows the five-set experiments designed to test the algorithm's feasibility of dynamic obstacle avoidance. The obstacle's trajectory is designed to be different from the UAV's potential trajectory, with different start and ending coordinates (Table III). The UAV is expected to reach the target position without colliding with the moving obstacle.

2) Results and Evaluation:

a) Case 4.1: As it is shown in Fig. 13, the obstacle travelled in positive x-axis direction from -6 to +6; the UAV is flying in the direction of y-axis, and they have the same altitude in z-axis. As shown in Fig.13a, when the UAV found the potential collision with a dynamic obstacle, it flies to the opposite direction of the obstacle towards negative x-axis. Another solution, as shown in Fig.13b, is when the UAV found the dynamic obstacle, it slightly drops its altitude in z-axis, and then back to original altitude.



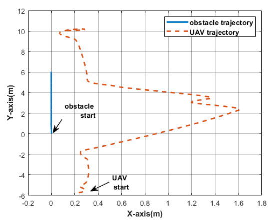
(a) 2D trajectory comparison



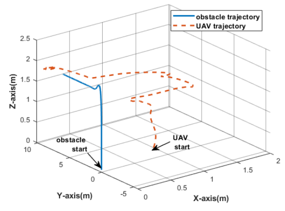
(b) 3D trajectory comparison

Fig. 13. Case 4.1: 2D and 3D trajectory of obstacle and UAV

b) Case 4.2: As it is shown in Fig. 14, the UAV and the identified obstacle are flying in the parallel directions, but their boundary boxes intersect each other. To avoid the potential collision, the UAV must move out of the boundary box of the obstacle. As shown in Fig. 14a, in the 2D space, the UAV is parallel with the obstacle along y-axis; hence, the UAV moves towards positive x-axis. In the 3D space, shown in Fig.14b, the UAV is parallel with the obstacle along z-axis, to avoid the potential collision, the UAV is moving towards positive x-axis.



(a) 2D trajectory comparison

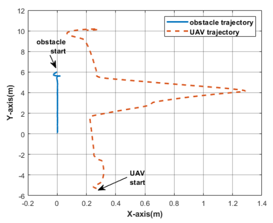


(b) 3D trajectory comparison

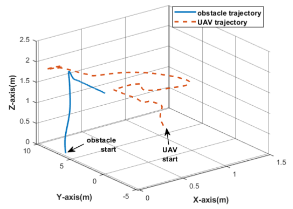
Fig. 14. Case 4.2: 2D and 3D trajectory of obstacle and UAV

c) Case 4.3: As it is shown in Fig. 15, Similarly to Case 4.2, the algorithm determined the corner with a shorter distance.

d) Case 4.4: As it is shown in Fig. 16, the UAV will fly to the bounding box corner with a higher number of opposite direction to the obstacle's velocity. Hence, the UAV is slightly flying away from its original direction but along the opposite direction of the identified obstacle. As shown in Fig. 16a and 16b, the UAV is flying towards negative x-axis and y-axis, while the obstacle is flying towards positive x-axis and y-axis.

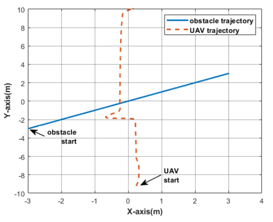


(a) 2D trajectory comparison

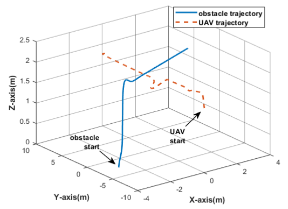


(b) 3D trajectory comparison

Fig. 15. Case 4.3: 2D and 3D trajectory of obstacle and UAV



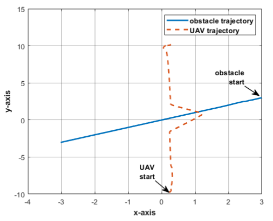
(a) 2D trajectory comparison



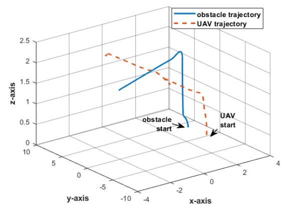
(b) 3D trajectory comparison

Fig. 16. Case 4.4: 2D and 3D trajectory of obstacle and UAV

e) Case 4.5: As it is shown in Fig. 17, similarly to case 4.4, the algorithm found a collision-free path by going to the negative direction on both x and y axes, which is the opposite direction of obstacle's movement.



(a) 2D trajectory comparison



(b) 3D trajectory comparison

Fig. 17. Case 4.5: 2D and 3D trajectory of obstacle and UAV

V. Conclusion

We have developed a comprehensive navigation system for an autonomous UAV. The navigation system consists of four sub-systems: mapping, localisation, cognition and control systems. Three case studies were designed to verify the feasibility of the algorithms to avoid a restricted area, static obstacle and dynamic obstacle. The experiments on the three cases demonstrate that the UAV is able to reach the target under all the three cases of environments. All simulated components are designed to match their real-world counterparts' dynamics and properties. Ideally, it can be transferred to a real UAV without any changes. The simulation system provides a platform for future robotic research.

As the simulation system is implemented in a modular way, it provides an easy platform for adding on modulars of designed algorithms. This will benefit for test and verification of the designed algorithms for autonomous navigation. Moreover, the system has good readability, maintainability and extendability. In this work, we did not address the obstacle detection. In future, we will investigate obstacle detection with machine learning and vision techniques, as well as multiple dynamic obstacles avoidance in an uncertain environment.

References

[1] B. Yenne, Attack of the drones: A history of unmanned aerial combat. Zenith Imprint, 2004.
[2] F. Gao, W. Wu, and S. Shen, Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments, Journal of Field Robotics, 36(4), pp.710-733, 2019.
[3] J. Peterson, H. Chaudhry, K. Abdelatty, J. Bird, and K. Kochersberger, Online aerial terrain mapping for ground robot navigation. Sensors, 18(2), p.630, 2018.
[4] C. Xia, Intelligent Mobile Robot Learning in Autonomous Navigation, Ph.D. dissertation, Ecole Centrale de Lille, 2015. [Online]. Available:https://tel.archives-ouvertes.fr/tel-01298608
[5] 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.
[6] T. Mac, C. Copot, R. De Keyser, and C. Ionescu, The development of an autonomous navigation system with optimal control of an UAV in partly unknown indoor environment, Mechatronics, 49, pp.187-196, 2018.
[7] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, Rotors – a modular gazebo MAV simulator framework, in Robot Operating System (ROS). Springer, pp. 595–625, 2016.
[8] A. Finn and S. Scheding, Developments and challenges for autonomous unmanned vehicles. Springer, 2012.
[9] T. Shima and S.Rasmussen, UAV cooperative decision and control: challenges and practical approaches, SIAM, 2009.
[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, Octomap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots, 2013, software available at http://octomap.github.com.
[11] T. Lee, M. Leoky, and N. H. McClamroch, Geometric tracking control of a quadrotor uav on SE (3), in 49th IEEE Conference on Decision and Control, pp. 5420-5425, IEEE, 2010.
[12] R. V. Kulkarni and G. K. Venayagamoorthy, Bio-inspired algorithms for autonomous deployment and localization of sensor nodes, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 40, no. 6, pp. 663-675, 2010.
[13] A. Nguyen and B. Le, 3d point cloud segmentation: A survey, in 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), pp. 225-230, Nov 2013.
[14] G.Szauer, Game Physics Cookbook, Packt Publishing Ltd, pp. 116-120, 2017.