IN: AIAA SCITECH 2024 Forum, 8-12 January 2024, Orlando, USA, Paper number AIAA 2024-2091
DOI: 10.2514/6.2024-2091

Dynamic Path Planning of UAV in Three-dimensional Complex Environment Based on Interfered Fluid Dynamical System

Komsun Tamanakijprasart*, Sabyasachi Mondal†, and Antonios Tsourdos‡
Cranfield University, Cranfield MK430AL, United Kingdom

The difficulties of path planning for unmanned aerial vehicles (UAVs) grow with the increase of static obstacles. Moreover, the presence of dynamic obstacles piles up the computation burden, as the UAVs need to dynamically replan and compute a new path to avoid them within an expanding search space. Existing studies on dynamic path planning have primarily evaluated algorithms in low-fidelity simulations and focused on improving computational efficiency. Nonetheless, these efforts remain insufficient for practical applications due to the limited computing powers of onboard processors, coupled with the significantly more cluttered nature of real-world environments. This paper introduces a dynamic autorouting program featuring the Interfered Fluid Dynamical System (IFDS) for adaptive path planning and a novel safeguarding function to ensure safety distances during obstacle avoidance. The proposed strategy brings a dynamic path planning framework, allowing UAVs to adaptively reroute to avoid areas and obstacles that will change throughout the flight in complex environments.

I. Nomenclature

CCA3D	=	three-dimensional Carrot-Chasing Algorithm
IFDS	=	Interfered Fluid Dynamical System
Γ	=	Obstacle modeling function
$\boldsymbol{M}$	=	modulation matrix
$\bar{\boldsymbol{u}}$	=	interfered fluid flow
ξ	=	UAV's position
a, b, c	=	object's axis length
ω	=	weighting coefficient
p, q, r	=	shape index parameters
ρ_0	=	repulsive parameter
σ_0	=	tangential parameter
SF	=	shape-following parameter
$\boldsymbol{t}$	=	unit tangential vector along the outward surface of the obstacle
$\boldsymbol{n}$	=	unit normal vector along the outward surface of the obstacle

II. Introduction

In recent years, the route planning technology for unmanned aerial vehicles has been advancing rapidly. As the research for practical implementation is increasingly demanded, numerous requirements and constraints, such as real-time computation and complex 3D dynamic environments, must be considered. This turns the path planning task into a very difficult problem as the computational complexity rises significantly, and the dynamic obstacle avoidance in a 3D complex environment has not received much attention. Traditional path planning methods, including A*, Probabilistic Roadmap (PRM), and exploring random Trees (RRT), or even novel computer intelligence methods such as Genetic

*MSc student in Autonomous Vehicle Dynamics and Control, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, College Road, Cranfield, Bedford, MK43 0AL, United Kingdom. Email: komxun@gmail.com

†Lecturer in Autonomous Systems, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, Email: sabyasachi.mondal@cranfield.ac.uk

‡Head of the Autonomous and Cyber-Physical Systems Centre, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, Email: a.tsourdos@cranfield.ac.uk, AIAA Senior Member.

Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are no longer sufficient for meeting real-world constraints and requirements.

In addition to the aforementioned methods, many novel methods involving modification and combination of the traditional methods have been proposed in recent studies. Authors in [1] propose solving a real-time collision avoidance by using Generalized Vector Explicit Guidance (GENEX) combined with the barrier function. Mondal et al. [2] used it along with differential geometric guidance law to achieve formation flying. GENEX, first introduced in [3], is a generalized form of an explicit optimal guidance algorithm in which the control energy is minimized and was originally designed for a missile interceptor [4]. Although proven to be successful in avoiding static obstacles in both 2D and 3D scenarios, the path generated by GENEX is suitable for avoiding static obstacles only.

The multi-objective path planning (MOPP) framework for UAV is investigated in [5], where multiple constraints are considered, and the information from offline and online paths has been jointly exploited. A similar spirit as the level set method and the search space cut down A* algorithm has been applied for offline and online path planning, respectively. An effective path meeting safety and travel time requirements were generated while avoiding dynamic obstacles. An interesting concept of search space cut down has also been proposed to improve efficiency further. Nonetheless, the algorithm has only been evaluated in 2D scenarios. Authors in [6] proposed an improved sparse hierarchical lazy theta* (SHLT*) algorithm to efficiently plan a near-optimal path in real-time under an unknown 3D space. Here, spatial modeling was employed, where the space was divided into different levels. The SHLT* algorithm can quickly recalculate the new coarse path when the UAV encounters obstacles. Though the simulation did not include moving obstacles, this method has a high potential to deal with dynamic obstacles in real-time. In [7], the modified genetic algorithm (MGA) was proposed to improve the exploration capability of the standard genetic algorithm. However, even though the dynamic environment was simulated in 2D, the computing time went as high as 0.70 seconds for the 25-obstacles case, creating a high doubt in real-time capability in a 3D complex environment. Authors in [8] proposed an efficient path planning via a comprehensively improved particle swarm optimization (CIPSO), claiming to perform better than MGA and all other variations of PSOs. Despite this improved computational efficiency, the average running time for CIPSO is still about 1.5 seconds for a 3D environment, which is unsuitable for real-time applications and high-speed UAVs. Having noticed this issue, [9] compared and executed improved GA, improved PSO, and improved grey wolf optimization (IGWO) on the CAN bus. It is revealed that the IGWO algorithm has the highest feasibility score with the lowest execution time. In [10] and [11], UAV path plannings based on IGWO were simulated; however, avoiding dynamic obstacles in a 3D complex environment still has not been investigated.

To the best of our literature survey, most novel path-planning methods are efficient in either a 2D plane or a 3D plane without dynamic obstacles. Transforming from a 2D problem to a 3D problem and from static to dynamic obstacle avoidance will significantly increase the computational cost. The path planning problem for a 3D complex environment with dynamic obstacles is hardly considered in most literature. The SHLT* algorithm and the IGWO algorithm have a high potential for dealing with this problem; however, there is a lack of research on these methods. Among novel path planning techniques, the Interfered Fluid Dynamical System (IFDS) method [12], [13] appears to have the highest practicality in terms of low-cost computation, real-time execution, 3D complex environments handleability, and dynamic obstacle avoidance. Moreover, the success of the hybrid IFDS algorithm in dealing with the aforementioned problem has been reported in many pieces of literature [14–18].

In order to meet all mission requirements, the following criteria will be determined for each candidate algorithm: 1) dynamic obstacle capability, 2) 3D environment capability, 3) real-time computation, and 4) optimal path. Table 1 compares the criteria fulfillment of different path-planning methods. The versatility of the IFDS algorithm has been demonstrated across various applications. For instance, an IFDS-based route planning approach for helicopters is proposed in [19] for an agricultural plant protection mission. The helicopter's flight performance and constraints were combined with the IFDS algorithm to determine the feasibility of the generated path. Another significant advancement is observed in [20], where the IFDS serves as the foundation for a multi-agent formation (MAF) framework by introducing the kinematic model and constraints combined. The receding horizon control was also added to improve obstacle avoidance performance and safety. In [21], maneuver control for obstacle avoidance (MCOA) of fixed-wing UAVs was refined by implementing a deep-reinforcement-learning-based reactive online decision-making mechanism built on top of the IFDS guidance law.

The current research trend is focused on integrating IFDS with complementary optimization algorithms. Reference [17] combined IFDS with quartic Bézier curve optimization to generate a trajectory satisfying kinematic constraints (e.g. maximum turn rate and climb rate) for improving smoothness and safety. In [16], the improved GA (IGA) incorporating GWO is used to optimize the feasible IFDS path for autonomous underwater vehicles (AUV). The use of a Neural Network has been proposed in [15] to adjust the IFDS parameters adaptively according to the environmental information.

Table 1 Path planning methods comparison: ✓- criteria satisfied, ✗- criteria not satisfied, ? - lack of research

No.	Path Planning Method	Dynamic	3D	Real-time	Optimal
1	GENEX [1, 3]	✗	✓	✓	✓
2	MGA [7]	?	✓	✗	✓
3	CIPSO [8]	✓	✓	✗	✓
4	SHLT* [6]	?	✓	✓	✗
5	IGWO [10, 11]	?	✓	?	✓
6	IFDS [12, 13]	✓	✓	✓	✗
7	Hybrid IFDS [14–18]	✓	✓	✓	✓

Lastly, an adaptive interfered fluid dynamic system algorithm (AIFDS) integrated with various reinforcement learning algorithms is proposed in [18] to enhance adaptability and overall path planning performance.

In this paper, the core components of the dynamic autorouting program include the IFDS algorithm, the safeguarding function, and the path-following algorithm. These components have been coded and simulated using MATLAB R2022b. The IFDS algorithm has found extensive applications in solving three-dimensional path planning problems. This study introduces a novel contribution by proposing the integration of a safeguarding function into the IFDS framework, ensuring that the generated path consistently maintains the required safe distance from obstacles. This function is analytically derived from the IFDS formulas, providing safety assurances irrespective of the initial tuning parameters of the IFDS. The proposed method offers the dual advantage of generating a safe and smooth path while keeping computational complexity to a minimum.

III. Problem Formulation

A. Aim and Objectives

The aim of this study is to develop an algorithm that is capable of rerouting obstacles autonomously onboard. The rerouting functionality is essential to avoid obstacles that may emerge throughout the course of the flight. These obstacles include static obstacles, dynamic obstacles, and pop-up threats such as congestion, events, fires, etc. The problem scenario is conceptually depicted in Fig. 1. These obstacles and threats are formulated as geometrical constraints in this problem. Furthermore, several assumptions have been taken into account, as outlined in Table 2.

Table 2 Problem details

Topic	Content
Requirements	• The route must avoid both static and dynamic obstacles • The route needs to adapt to areas that will change throughout the flight
Constraints	• Static obstacles such as terrain and buildings • Dynamic obstacles such as moving aircraft or other drones • Pop-up threats such as congestion, fire, etc.
Assumptions	• The algorithm will be computed onboard the UAV • The area map is given • The location of the static obstacles and dynamic obstacles are known • The behaviors of pop-up threats are unknown • The UAV's true position is known

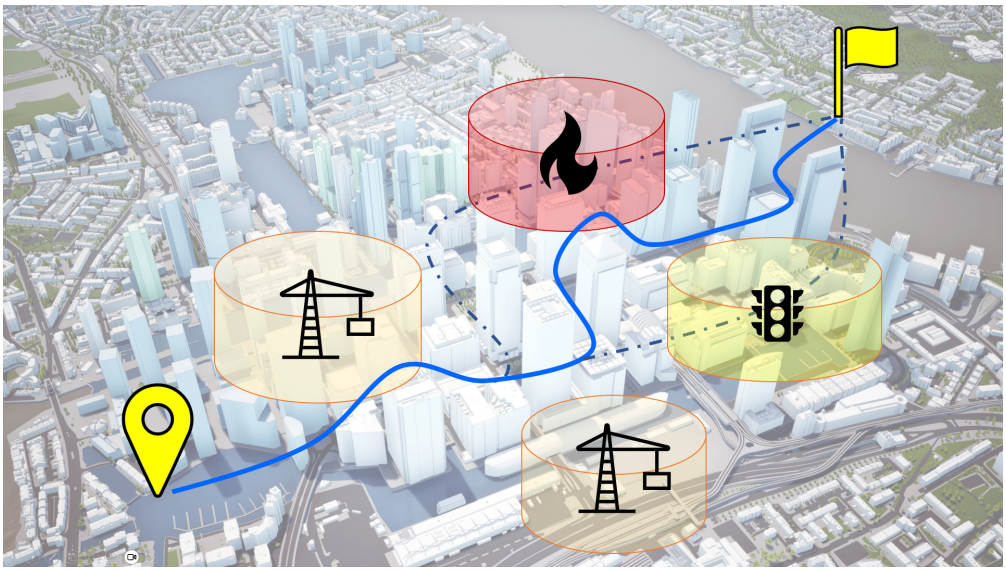

Fig. 1 Problem scenario.

B. Modeling of Environment

While real-world objects and terrain, including mountains, trees, buildings, and aircraft, often have complex shapes, they can be simplified into basic geometric forms such as cylinders, cones, spheres, parallelepipeds, or combinations. This simplification allows the obstacle avoidance problem to be formulated for the IFDS method as solving for fluid streamlined solutions that avoid these simplified shapes. To begin, the object modeling function Γ is introduced to represent basic geometric shapes, expressed as follows:

$$\Gamma(\xi) = \left(\frac{x - x_0}{a}\right)^{2p} + \left(\frac{y - y_0}{b}\right)^{2q} + \left(\frac{z - z_0}{c}\right)^{2r} = 1 \tag{1}$$

where $\xi = (x, y, z)$ is the UAV position, $\xi_0 = (x_0, y_0, z_0)$ is object's origin point, (a, b, c) are object's axis length, and (p, q, r) are index parameters which give various shapes as follow:

$$\text{object} = \begin{cases} \text{sphere}, & \text{if } p = q = r = 1. \\ \text{cylinder}, & \text{if } p = q = 1 \text{ and } r > 1. \\ \text{cone}, & \text{if } p = q = 1 \text{ and } r < 1. \\ \text{parallelepiped}, & \text{if } p > 1, q > 1 \text{ and } r > 1. \end{cases} \tag{2}$$

The object modeling function $\Gamma(\xi)$ acts as a boundary equation of the obstacle where $\Gamma(\xi) = 1$ when evaluated at the object's surface, and $\Gamma(\xi) > 1$ when measured at points away from the object. The expansion of the $\Gamma(\xi)$ value occurs in three dimensions, reflecting the distribution corresponding to the shape of the obstacle. The distribution of $\Gamma(\xi)$ for various shapes are shown in Fig. 2 - 4

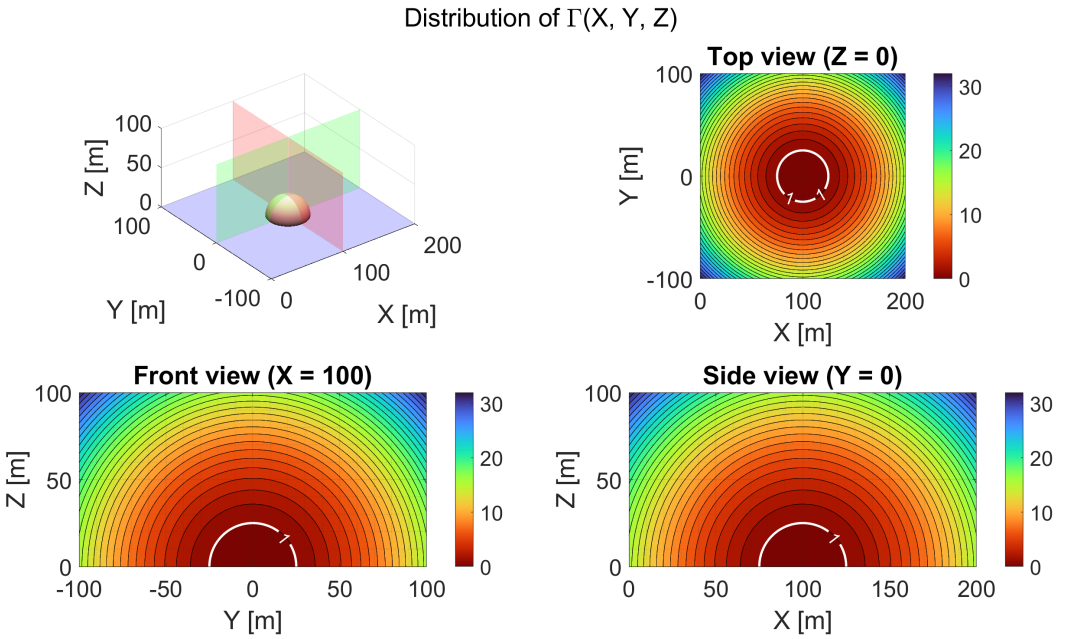

Fig. 2 The distribution of $\Gamma(\xi)$ for a sphere.

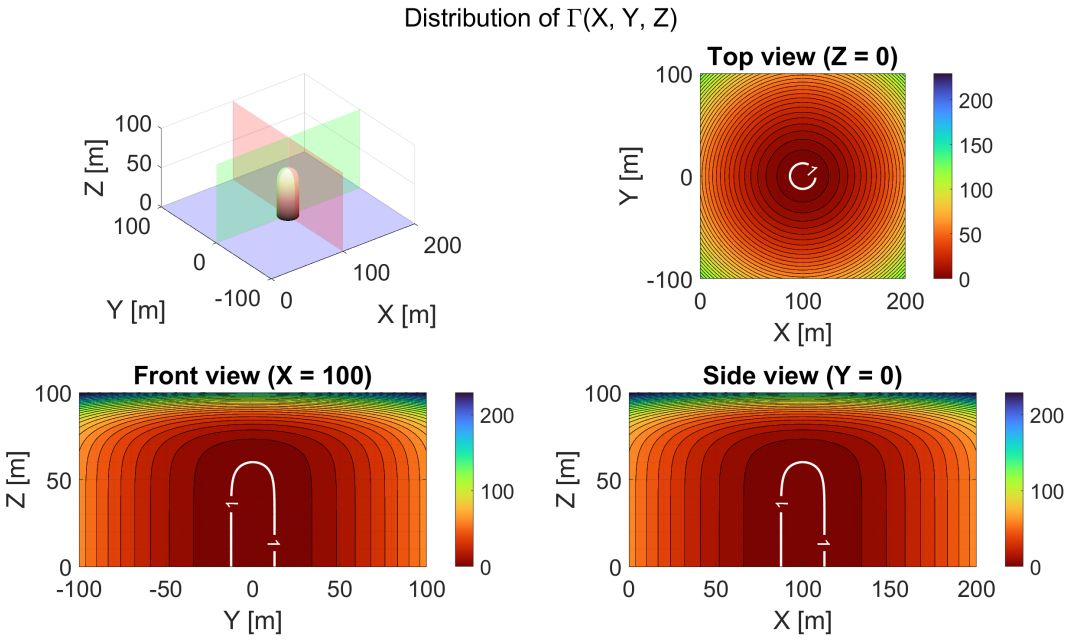

Fig. 3 The distribution of $\Gamma(\xi)$ for a cylinder.

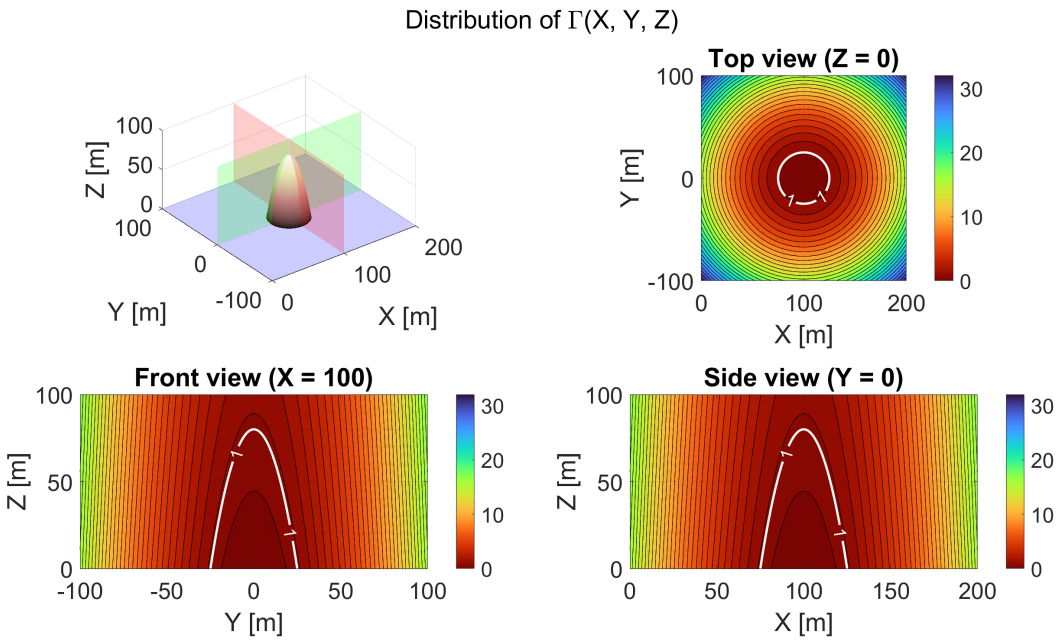

Fig. 4 The distribution of $\Gamma(\xi)$ for a cone.

IV. Interfered Fluid Dynamical System

A. Modeling of interfered fluid flow

In the modeling of interfered fluid flow, the position of the UAV is represented by $\xi = (x, y, z)$, and the object's origin point is denoted as (x_0, y_0, z_0). The UAV's trajectory is determined by iteratively updating its state ξ using the modified velocity field of the interfered fluid, denoted as $\bar{\boldsymbol{u}}(\xi)$. This velocity field is adjusted to navigate around the boundaries of obstacles within the field. The update of the UAV's position is given by the following equations:

$$\xi_{k+1} = \xi_k + \bar{\boldsymbol{u}}(\xi)\Delta t$$

$$\bar{\boldsymbol{u}}(\xi) = \boldsymbol{M}(\xi)\boldsymbol{u}(\xi) = \boldsymbol{u}(\xi) + \boldsymbol{v}(\xi)$$

Here, $\bar{\boldsymbol{u}}(\xi)$ is calculated as the sum of the original fluid velocity $\boldsymbol{u}(\xi)$ and an additional velocity component $\boldsymbol{v}(\xi)$ resulting from the obstacle. The magnitude of this additional velocity $\boldsymbol{v}(\xi)$ increases as the UAV approaches the obstacle. Let $d(\xi)$ be the distance between the current UAV position ξ and the final destination $\xi_d = (x_d, y_d, z_d)$. The fluid strength, representing the UAV's cruising speed, is denoted as a constant C. The original fluid flow can be conceptualized as a sink located at the destination, represented by:

$$\boldsymbol{u}(\xi) = \left[\frac{C(x - x_d)}{d(\xi)}, \quad \frac{C(y - y_d)}{d(\xi)}, \quad \frac{C(z - z_d)}{d(\xi)}\right]^\intercal$$

$$d(\xi) = \sqrt{(x - x_d)^2 + (y - y_d)^2 + (z - z_d)^2}$$

The original fluid velocity is then modified by the modulation matrix $\boldsymbol{M}(\xi)$, which perturbs the flow in response to obstacles. The definition of $\boldsymbol{M}(\xi)$ is given as:

$$\boldsymbol{M}(\xi) = \boldsymbol{I} - \frac{\boldsymbol{n}(\xi)\boldsymbol{n}(\xi)^\intercal}{|\Gamma(\xi)|^{\frac{1}{\rho(\xi)}}\boldsymbol{n}(\xi)^\intercal\boldsymbol{n}(\xi)} + \frac{\boldsymbol{t}(\xi)\boldsymbol{n}(\xi)^\intercal}{|\Gamma(\xi)|^{\frac{1}{\sigma(\xi)}}\|\boldsymbol{t}(\xi)\|\|\boldsymbol{n}(\xi)\|} \tag{3}$$

where $\boldsymbol{n}(\xi)$, $\boldsymbol{t}(\xi)$, $\rho(\xi)$, and $\sigma(\xi)$ are defined as follows:

$$\begin{aligned}
\boldsymbol{n}(\xi) &= \left[\frac{\partial \Gamma(\xi)}{\partial x}, \quad \frac{\partial \Gamma(\xi)}{\partial y}, \quad \frac{\partial \Gamma(\xi)}{\partial z}\right]^{\mathsf{T}} \\
\boldsymbol{t}(\xi) &= \left[\frac{\partial \Gamma(\xi)}{\partial y}, \quad -\frac{\partial \Gamma(\xi)}{\partial x}, \quad 0\right]^{\mathsf{T}} \\
\rho(\xi) &= \rho_0 \exp\left(1 - \frac{1}{d_0(\xi) d(\xi)}\right) \\
\sigma(\xi) &= \sigma_0 \exp\left(1 - \frac{1}{d_0(\xi) d(\xi)}\right)
\end{aligned}$$

ρ_0 is the repulsive parameter, σ_0 is the tangential parameter, and $d_0(\xi)$ is the distance between the obstacle's boundary and the UAV's current position ξ. Additionally, $\boldsymbol{n}(\xi)$ and $\boldsymbol{t}(\xi)$ are the unit normal vector and the unit tangential vector along outward on the surface of the obstacle, respectively. ρ_0 defines the range in which the path responds to the presence of an obstacle. A greater ρ_0 results in a larger deflection to the original flow. Conversely, σ_0 governs the degree of flexibility for lateral path deviations. A higher σ_0 value makes the path more adaptable to sideways deviations. The influence of the tuning parameters ρ_0 and σ_0 is visually demonstrated in Fig. 5.

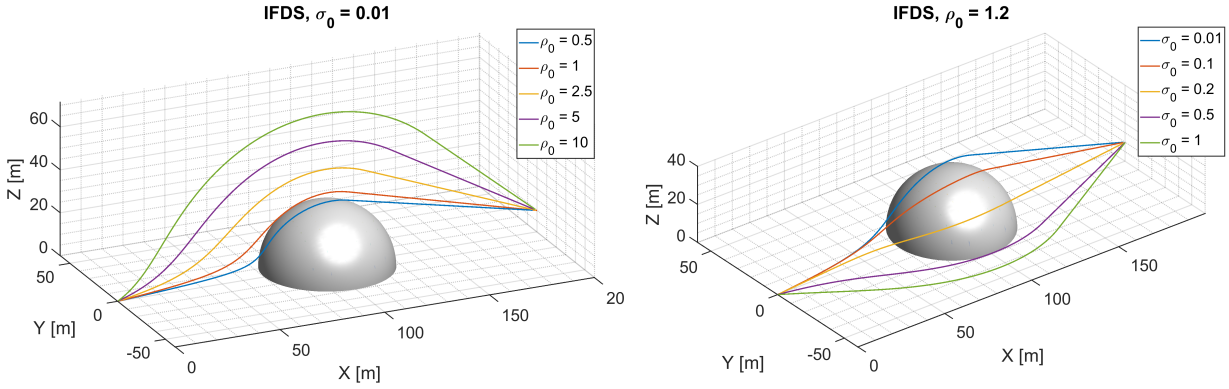

Fig. 5 3D paths with varied IFDS parameters

In some cases, following the shape of an obstacle is not desired. Authors in reference [13] proposed the shape-following parameter SF $\in \{0, 1\}$ so that SF = 1 when the shape-following is demanded, and SF = 0 when it is not. The effect of this feature is shown in Fig. 6. The modulation matrix from equation 3 is then redefined as:

$$\boldsymbol{M}(\xi) = \begin{cases} \boldsymbol{I} - \dfrac{\boldsymbol{n}(\xi)\boldsymbol{n}(\xi)^{\mathsf{T}}}{|\Gamma(\xi)|^{\frac{1}{\rho(\xi)}}\boldsymbol{n}(\xi)^{\mathsf{T}}\boldsymbol{n}(\xi)} + \dfrac{\boldsymbol{t}(\xi)\boldsymbol{n}(\xi)^{\mathsf{T}}}{|\Gamma(\xi)|^{\frac{1}{\sigma(\xi)}}\|\boldsymbol{t}(\xi)\|\|\boldsymbol{n}(\xi)\|}, & \text{if } \boldsymbol{n}(\xi)^{\mathsf{T}}\boldsymbol{v}(\xi) < 0 \text{ or SF} = 1 \\ \boldsymbol{I}, & \text{if } \boldsymbol{n}(\xi)^{\mathsf{T}}\boldsymbol{v}(\xi) \geq 0 \text{ and SF} = 0 \end{cases}$$

B. Avoidance of multiple obstacles

For K obstacles in the environment, let the object modeling function of the k^{th} obstacle be $\Gamma_k(\xi) = 1, k = 1, 2, ..., K$ and the interfered fluid flow for the k^{th} obstacle is

$$\bar{\boldsymbol{u}}_k(\xi) = \boldsymbol{M}_k(\xi)\boldsymbol{u}(\xi)$$

$$\boldsymbol{M}_k(\xi) = \boldsymbol{I} - \frac{\boldsymbol{n}_k(\xi)\boldsymbol{n}_k(\xi)^{\mathsf{T}}}{|\Gamma_k(\xi)|^{\frac{1}{\rho_k(\xi)}}\boldsymbol{n}_k(\xi)^{\mathsf{T}}\boldsymbol{n}_k(\xi)} + \frac{\boldsymbol{t}_k(\xi)\boldsymbol{n}_k(\xi)^{\mathsf{T}}}{|\Gamma_k(\xi)|^{\frac{1}{\sigma_k(\xi)}}\|\boldsymbol{t}_k(\xi)\|\|\boldsymbol{n}_k(\xi)\|}$$

Let $\omega_k(\xi)$ be the weighting coefficient of the k^{th} obstacle, and $\bar{\omega}_k(\xi)$ be the normalized relative weighting coefficient:

$$\omega_k(\xi) = \prod_{i=1, i\neq k}^{K} \frac{\Gamma_i(\xi) - 1}{(\Gamma_k(\xi) - 1) + (\Gamma_i(\xi) - 1)}$$

$$\tilde{\omega}_k(\xi) = \frac{\omega_k(\xi)}{\sum_{i=1}^{K} \omega_i(\xi)}$$

The weighted average of the velocity field is used to ensure that all obstacles in the area are avoided. The interfered fluid flow for multiple obstacle avoidance can be seen in Fig. 7.

$$\boldsymbol{M}(\xi) = \sum_{k=1}^{K} \tilde{\omega}_k(\xi) \boldsymbol{M}_k(\xi)$$

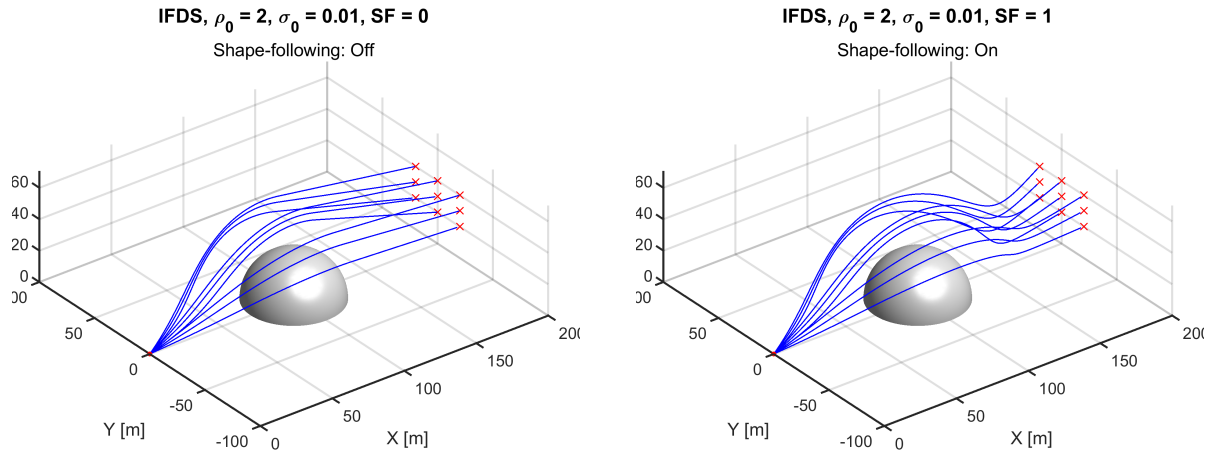

Fig. 6 The effect of the shape-following parameter on the IFDS paths. Left: SF = 0. Right: SF = 1.

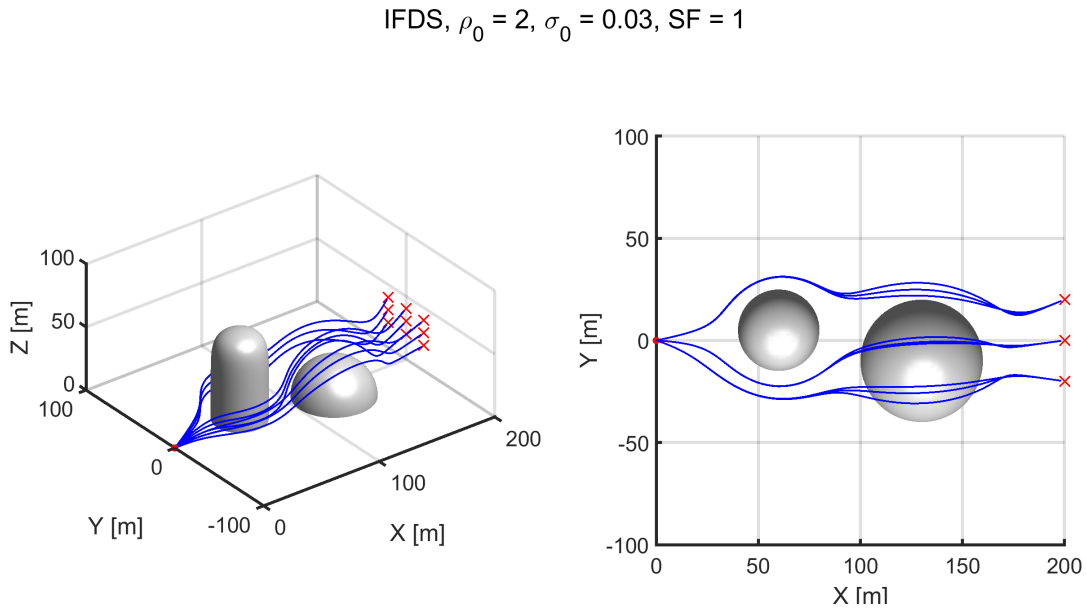

Fig. 7 Paths generated from the IFDS algorithm for avoiding multiple obstacles.

V. Safeguarding

Regulations and guidelines governing UAV operations have been established to mitigate the risks associated with drone flights. Among these regulations, one crucial requirement is maintaining a safe distance from obstacles to reduce the likelihood of accidents. The Drone Code by the Civil Aviation Authority (CAA) emphasizes maintaining a safe distance from obstacles, as illustrated in Fig. 8. It is imperative to develop robust and reliable methods that enable UAVs to navigate complex environments with a high level of autonomy while adhering to these safety regulations. The presented safeguarding function addresses this pressing challenge by providing an effective means for UAVs to avoid obstacles safely at a specified range. By leveraging the boundary function for arbitrary shapes, the function facilitates real-time path planning for UAVs to adjust their flight paths dynamically and maintain a safe distance from obstacles. Generalizing the approach, the safeguarding function is formulated as a function of IFDS tuning parameters, ρ_0, facilitating adaptable protection for various scenarios. The effectiveness of the safeguarding function is demonstrated through comparative analysis of path trajectories, revealing the significant improvement in maintaining a safe distance from the object's surface. The presented safeguarding function ensures path safety even when the tuning parameter, ρ_0, is set to a small value, enhancing the reliability and safety of IFDS operations.

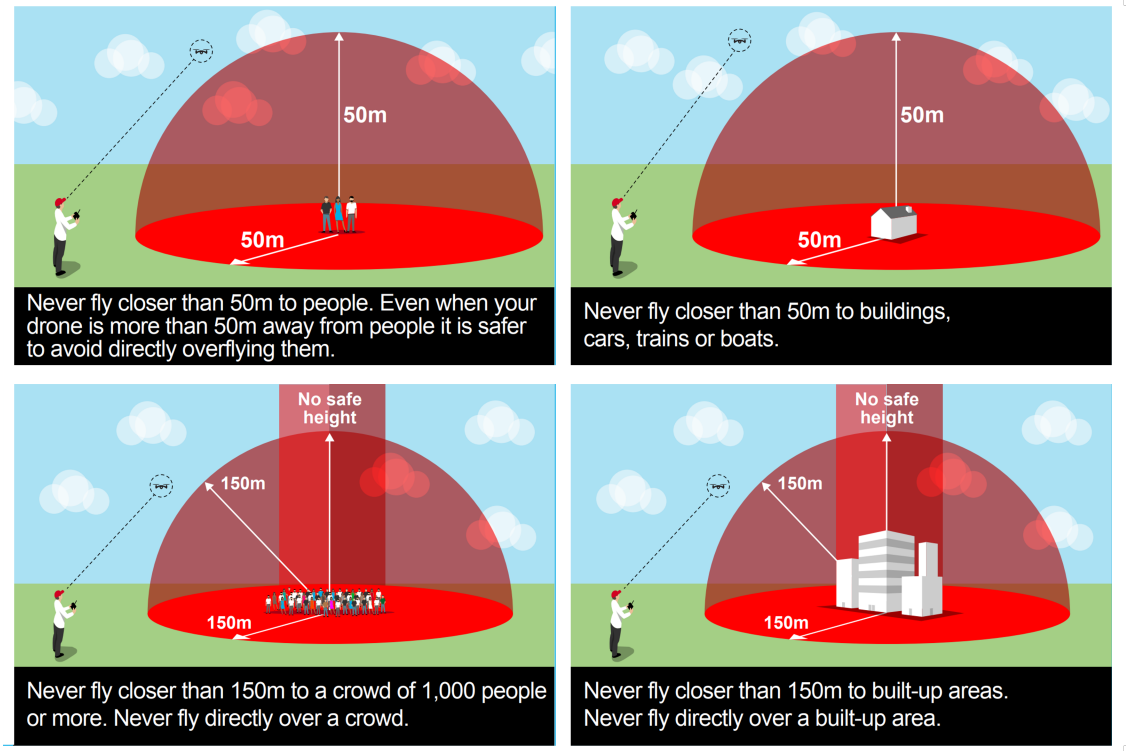

Fig. 8 Drone code - Safety guidelines for operating drones (dronesafe.uk)

A. Safeguarding Function Derivation

The boundary function for arbitrary shapes is given as

$$\Gamma(\xi) = \left(\frac{x - x_0}{a}\right)^{2p} + \left(\frac{y - y_0}{b}\right)^{2q} + \left(\frac{z - z_0}{c}\right)^{2r}$$

where the object's surface is located at $\Gamma(\xi) = 1$. For a sphere, $p = q = r = 1$ and $a = b = c$ is its radius; the boundary function then becomes

$$\Gamma(\xi) = \left(\frac{x - x_0}{a}\right)^{2} + \left(\frac{y - y_0}{a}\right)^{2} + \left(\frac{z - z_0}{a}\right)^{2} \tag{4}$$

Consider a sphere located at the same origin (x_0, y_0, z_0) with the radius $a + \delta_g$ and the boundary function $\Gamma^*(\xi)$, the location of its surface can be expressed from equation 4 as follow:

$$\Gamma^*(\xi) = \left(\frac{x - x_0}{a + \delta_g}\right)^2 + \left(\frac{y - y_0}{a + \delta_g}\right)^2 + \left(\frac{z - z_0}{a + \delta_g}\right)^2 = 1$$
$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = (a + \delta_g)^2 \tag{5}$$

Substituting (5) into (4) yields

$$\Gamma(\xi) = \left(\frac{a + \delta_g}{a}\right)^2$$

This is the value of $\Gamma(\xi)$ at a distance $a + \delta_g$ away from the object's origin, as illustrated in Fig. 9. The relationship between $\Gamma(\xi)$ and $\Gamma^*(\xi)$ is found to be:

$$\Gamma^*(\xi) = \Gamma(\xi) - \left(\frac{a + \delta_g}{a}\right)^2 + 1 \tag{6}$$

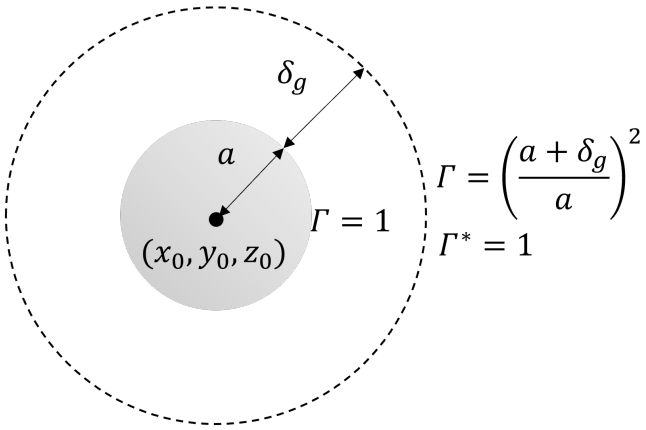

Fig. 9 Illustration of the boundary function at distance δ_g from the object's surface.

The safeguarding can be performed by replacing the boundary function $\Gamma(\xi)$ with $\Gamma^*(\xi)$ as expressed in equation (6). Nonetheless, to generalize this process, the safeguarding function will be expressed as a function of IFDS tuning parameters. Let's consider

$$|\Gamma(\xi)|^{\frac{1}{\rho^*}} = |\Gamma^*(\xi)|^{\frac{1}{\rho}}$$
$$|\Gamma(\xi)|^{\frac{1}{\rho^*}} = \left|\Gamma(\xi) - \left(\frac{a + \delta_g}{a}\right)^2 + 1\right|^{\frac{1}{\rho}}$$

Since $\rho = \rho_0 \exp\left(1 - \frac{1}{d_0 d}\right)$, the safeguarding function can be expressed in term of the IFDS tuning parameter ρ_0 as follow:

$$\rho_0^*(\xi) = \frac{\ln|\Gamma(\xi)|}{\ln\left|\Gamma(\xi) - \left(\frac{a + \delta_g}{a}\right)^2 + 1\right|}\rho_0 \tag{7}$$

The safeguarding function (7) ensures the path will not get closer than δ_g from the object's surface, even if the ρ_0 was initially set to have a very small value.

B. The Effect of Safeguarding Function

The effect of the safeguarding function is shown in Fig. 10 - 11. In Fig. 10, ρ_0 was set to be 0.5, causing the path to get very close to the object's surface. With the safeguarding function, ρ_0 was modified along the path, keeping the distance of at least $\delta_g = 10\ m$ away from the object's surface as shown in Fig. 11.

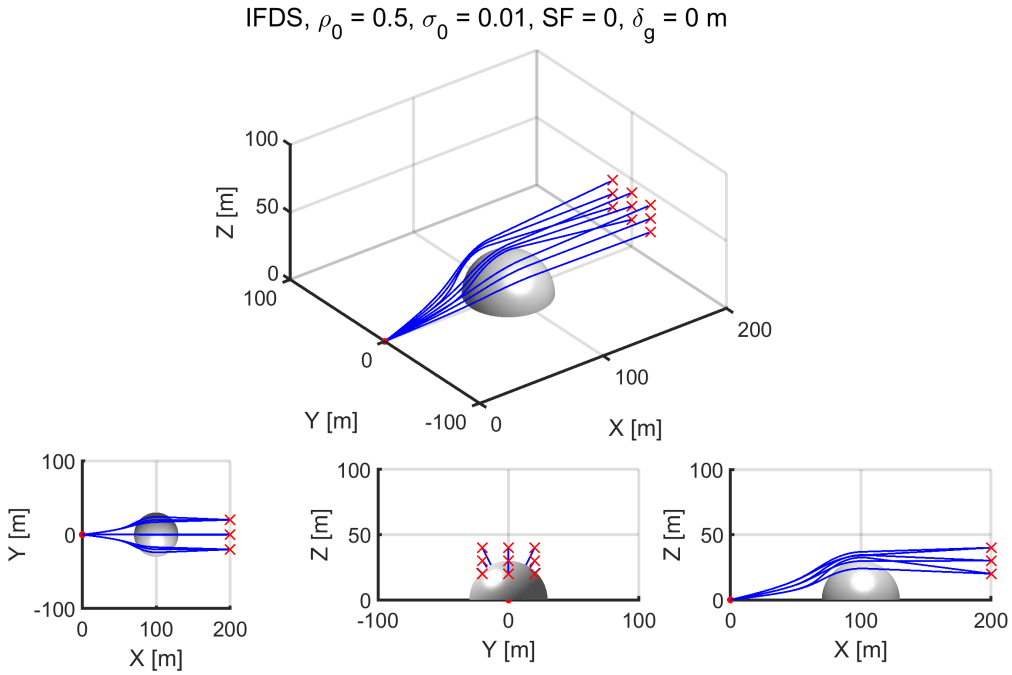

Fig. 10 Paths from IFDS without the safeguarding function.

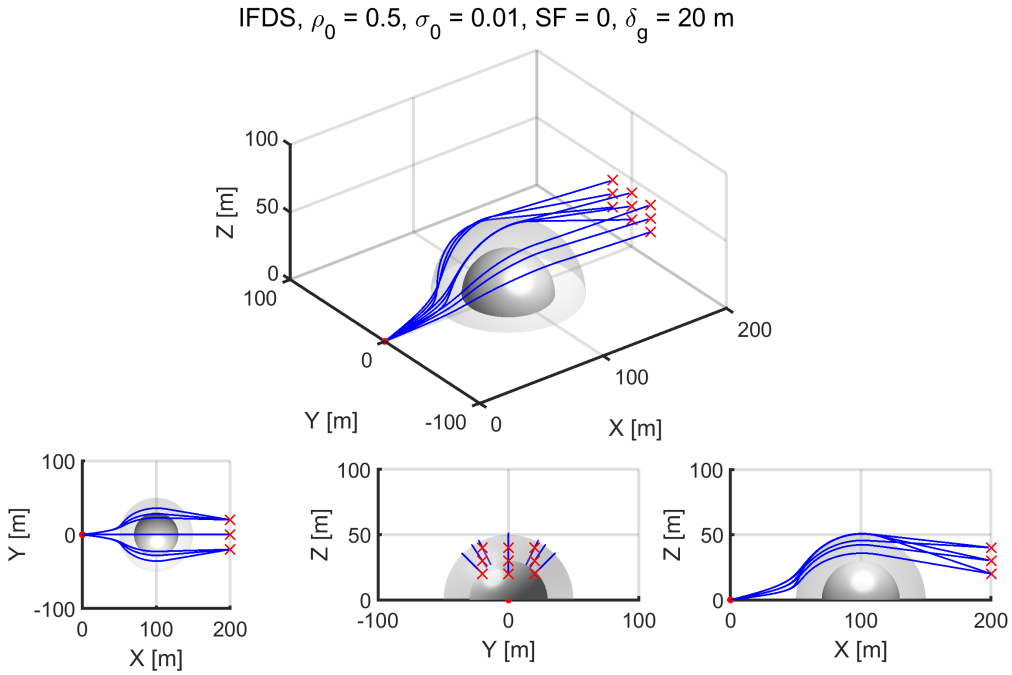

Fig. 11 Paths from IFDS with the safeguarding function.

VI. Results and Discussions

A. Scenario Creation

We have presented a simulation study for two scenarios that describe multiple static and dynamic obstacles of different shapes. The location and properties of the obstacles are shown in Table 9. Therefore, the shape and edge information are known. The dynamic obstacles' position changes over time according to the sinusoid shown in the table. Scenario 1 describes the static obstacles, while scenario 2 includes the dynamic obstacles.

Table 3 Obstacles setups for different scenarios.

Scenario	Type of obstacle	Shape of obstacle	Origin of obstacle $(x_0,\ y_0,\ z_0)$	Properties of obstacle
1	Static	Cylinder	$(60,\ 5,\ 0)$	$r = 15\ m, \quad h = 50\ m$
	Static	Sphere	$(120,\ -10,\ 10)$	$r = 15\ m$
	Static	Cylinder	$(168,\ 0,\ 0)$	$r = 12.5\ m, \quad h = 80\ m$
2	Static	Sphere	$(100,\ 0,\ 0)$	$r = 15\ m$
(Example 1)	Dynamic	Cylinder	$(100 + 50\sin(t/8),\ 50\cos(t/8),\ 0)$	$r = 10\ m, \quad h = 80\ m$
	Dynamic	Cylinder	$(100 - 50\sin(t/8),\ -50\cos(t/8),\ 0)$	$r = 10\ m, \quad h = 50\ m$
2	Static	Cylinder	$(60,\ 5,\ 0)$	$r = 15\ m, \quad h = 50\ m$
(Example 2)	Static	Cylinder	$(110,\ -10,\ 0)$	$r = 12.5\ m, \quad h = 80\ m$
	Dynamic	Cylinder	$(80,\ 50\sin(t/5),\ 0)$	$r = 10\ m, \quad h = 60\ m$
	Dynamic	Sphere	$(160,\ -20 - 20\sin(t/2),\ 40 + 20\cos(t/2))$	$r = 15\ m$

Before the UAV flies, it creates the path using IFDS, considering the known static obstacles. Then, it starts following the path using a path-following algorithm. In this study, we employ the widely recognized Carrot-chasing algorithm (CCA). Throughout the flight, the UAV dynamically adjusts its path onboard in response to encountering dynamic obstacles.

B. Simulation setups

Given that the simulated area spans 200×200 square meters, the UAV's cruising speed has been set at 10 m/s. This moderate speed ensures that the UAV's reactions to the dynamic environment are adequately captured. Therefore, the simulation time is set within the range of 20 to 50 seconds, depending on the time required by the UAV to attain the 1-meter threshold for reaching the destination. Finally, the starting and final locations are specified in Table 4, and the UAV's parameters are detailed in Table 5.

Table 4 Simulation setups.

Parameter	Variable name	Value
Simulation time	$rtsim$	20 - 50 s
Threshold for reaching destination	$targetThresh$	$1\ m$
Starting location	$(X_{ini},\ Y_{ini},\ Z_{ini})$	$(0,\ 0,\ 0)\ m$
Target destination	$(X_{final},\ Y_{final},\ Z_{final})$	$(200,\ 0,\ 10)\ m$
Safeguarding distance	δ_g	$10\ m$

Table 5 UAV parameters setups.

Parameter	Variable name	Value
UAV's cruising speed	C	$10\ m/s$
UAV's initial location	$(x_i,\ y_i,\ z_i)$	$(0,\ -20,\ 5)\ m$
UAV's initial yaw and pitch angle	$(\psi_i,\ \gamma_i)$	$(0,\ 0)\ rad$
UAV's minimum turning radius	$(Ryaw_{min},\ Rpitch_{min})$	$(10,\ 10)\ m$

C. Tuning parameters initialisation

With the safeguarding function active, the initial repulsive parameter ρ_0 has been configured within the range of 1 to 2.5. This adjustment aims to achieve the most direct path possible. Meanwhile, the tangential parameter σ_0 is employed to govern the lateral deviation of the path and has been configured at a value of 0.01. This setting is chosen to prioritize longitudinal mobility, specifically enabling the UAV to fly over obstacles. In these scenarios, the shape-following functionality has been deactivated due to its lack of necessity. The specific tuning parameters for the IFDS algorithm are outlined in Table 6.

Table 6 IFDS tuning parameters setups.

Parameter	Variable name	Value
Initial repulsive parameter	ρ_0	$1 - 2.5$
Initial tangential parameter	σ_0	0.01
Shape-following parameter	SF	0

D. Three-dimensional Carrot-Chasing Algorithm

The Carrot-Chasing Algorithm (CCA) is a well-known path-following technique, widely used in two-dimensional applications [22], [23], [24]. The three-dimensional CCA (CCA3D) builds upon the classic CCA while considering the UAV's three-dimensional position and orientation. Given a predefined path consisting of waypoints, the algorithm calculates the feasible trajectory that the UAV can follow while respecting pitch and yaw angle limitations. By utilizing trigonometric relations and vector operations, the algorithm ensures smooth and continuous path tracking, allowing the UAV to navigate effectively in complex environments. The CCA3D is outlined in Algorithm 1.

The algorithm begins with parameter initialization, setting various essential parameters for subsequent computation. These parameters include two consecutive waypoints $\boldsymbol{W}i$, $\boldsymbol{W}_{i+1}$, UAV's current position $\boldsymbol{p}$, its yaw and pitch angle ψ, γ, and its velocity v_a. Tuning parameters for CCA3D include the look-ahead distance δ and the CCA gain κ. Following initialization, the algorithm undertakes distance calculations R_u, which measures the distance between the UAV's current position p and the current waypoint $\boldsymbol{W}_i$. Additionally, it computes R_w, corresponding the distance between the current waypoint $\boldsymbol{W}_i$ and the next waypoint $\boldsymbol{W}_{i+1}$. Then, it calculates angles θ_1, θ_2, θ_{u_1}, and θ_{u_2} which define the orientation between waypoints and the UAV's position.

The next phase involves trajectory point computation. Here, the algorithm carefully calculates a new point (x_t, y_t, z_t) within the 3D space. This point is positioned at a distance of $R + \delta$ from the current waypoint $\boldsymbol{W}_i$ and aligns with the direction towards the subsequent waypoint $\boldsymbol{W}_{i+1}$. This calculation is critical in enabling the UAV's trajectory to converge to the path while still adhering to specified pitch and yaw angle constraints. Moving further, the algorithm determines the desired pitch angle (ψ_d) and yaw angle (γ_d) that extend from the UAV's present position to the newly computed trajectory point. These angles are essential for controlling the UAV's orientation during its flight. In the final stages, the algorithm calculates acceleration commands (u_1 and u_2) based on these desired pitch and yaw angles. These commands are regulated by the CCA gain κ, ensuring that the UAV's trajectory tracking remains smooth and feasible.

Algorithm 1 Three dimensional Carrot-Chasing Algorithm

Initialize: $\boldsymbol{W}_i = (x_i, y_i, z_i), \boldsymbol{W}_{i+1} = (x_{i+1}, y_{i+1}, z_{i+1}), \boldsymbol{p} = (x, y, z), \psi, \gamma, \delta, \kappa, v_a$

$R_u = ||\boldsymbol{W}_i - \boldsymbol{p}|| = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$

$R_w = ||\boldsymbol{W}_{i+1} - \boldsymbol{W}_i|| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$

$\theta_1 = atan2(y_{i+1} - y_i, x_{i+1} - x_i)$

$\theta_2 = atan2(z_{i+1} - z_i, \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2})$

$\theta_{u_1} = atan2(y - y_i, x - x_i)$

$\theta_{u_2} = atan2(z - z_i, \sqrt{(x - x_i)^2 + (y - y_i)^2})$

if $R_u \neq 0$ **then**

    $\alpha = acos(\boldsymbol{R}_u \cdot \boldsymbol{R}_w, R_u R_w)$

else

    $\alpha = 0$

end if

$R = \sqrt{R_u^2 - (R_u \sin\alpha)^2}$

$(x_t, y_t, z_t) \leftarrow (R + \delta)(\cos\theta_2 \cos\theta_1, \cos\theta_2 \sin\theta_1, \sin\theta_2)$

$\psi_d = atan2(y_t - y, x_t - x)$

$\gamma_d = atan2(z_t - z, \sqrt{(x_t - x)^2 + (y_t - y)^2})$

$u_1 = maxlim(\kappa(\psi_d - \psi)v_a)$

$u_2 = maxlim(\kappa(\gamma_d - \gamma)v_a)$

In this study, a look-ahead distance δ of 20 meters has been chosen. This moderate value is employed to prevent path oscillations. Additionally, the gain parameter κ has been set to 50, ensuring swift convergence of the trajectory to the designated path. The tuning values for the CCA3D algorithm are outlined in Table 7.

Table 7 CCA3D tuning parameters setups.

Parameter	Variable name	Value
Gain	κ	50
Look-ahead distance	δ	20 m

E. Scenario 1: Multiple Static Obstacles Avoidance

The path generated by the UAV using the IFDS is depicted in Fig. 12 as a dashed blue line. A transparent cover shows the safety distance around the actual obstacles. It can be observed that integrating the safeguarding function with IFDS provides a path that consistently maintains a safe distance from the edges of obstacles. The UAV, represented by a white circle, follows the path using CCA3D and successfully reaches its destination while safely avoiding the static obstacles.

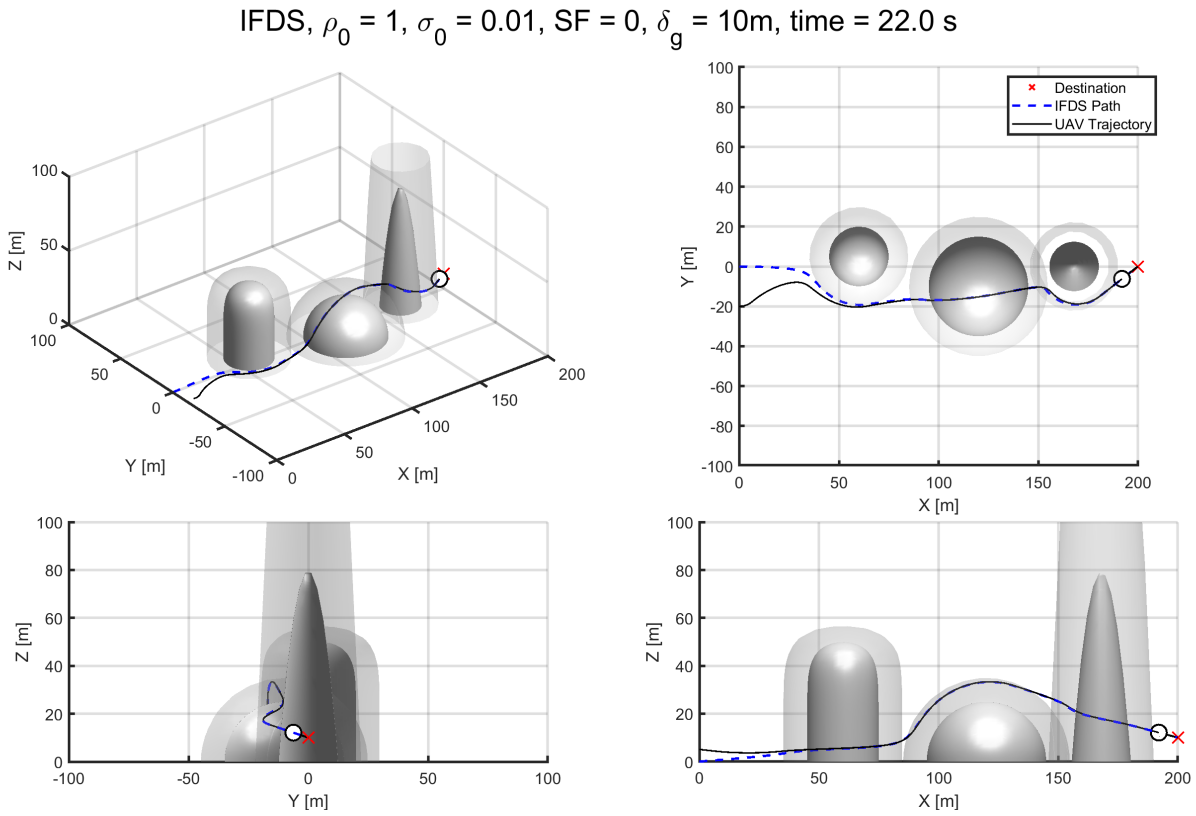

Fig. 12 Scenario 1: Multiple Static Obstacles Avoidance

F. Scenario 2: Multiple Static and Dynamic Obstacles Avoidance

The evaluation of the dynamic autorouting algorithm in the context of dynamic obstacles includes two distinct examples; each focused on assessing its performance concerning both ground and airborne obstacles. The first example, illustrated in Figs.13 and 14, replicated a scenario where grounded obstacles executed circular motions. On the other hand, the second example, depicted in Figs.15 and 16, simulated a situation where one obstacle patrolled linearly along the ground while another remained airborne. Across both tests, the dynamic autorouting algorithm adapted to the evolving environment, successfully guiding the UAV to its destination without any collisions with obstacles.

1. Example 1

In response to dynamic changes in the environment or obstacles, the IFDS algorithm adjusts its approach. Instead of computing the path from the initial starting location to the destination, it calculates the path onboard from the UAV's real-time position to the destination. This adjustment enables the path to adapt to the changing surroundings and obstacles dynamically. In Fig. 13, the UAV starts to follow the IFDS path, and when it encounters a dynamic obstacle, it recalculates its path onboard. A notable transition in the path's lateral alignment is observable, shifting from the right side to the left side of the obstacle within the time interval of $t = 12$ to 14 s as shown in Fig. 14.

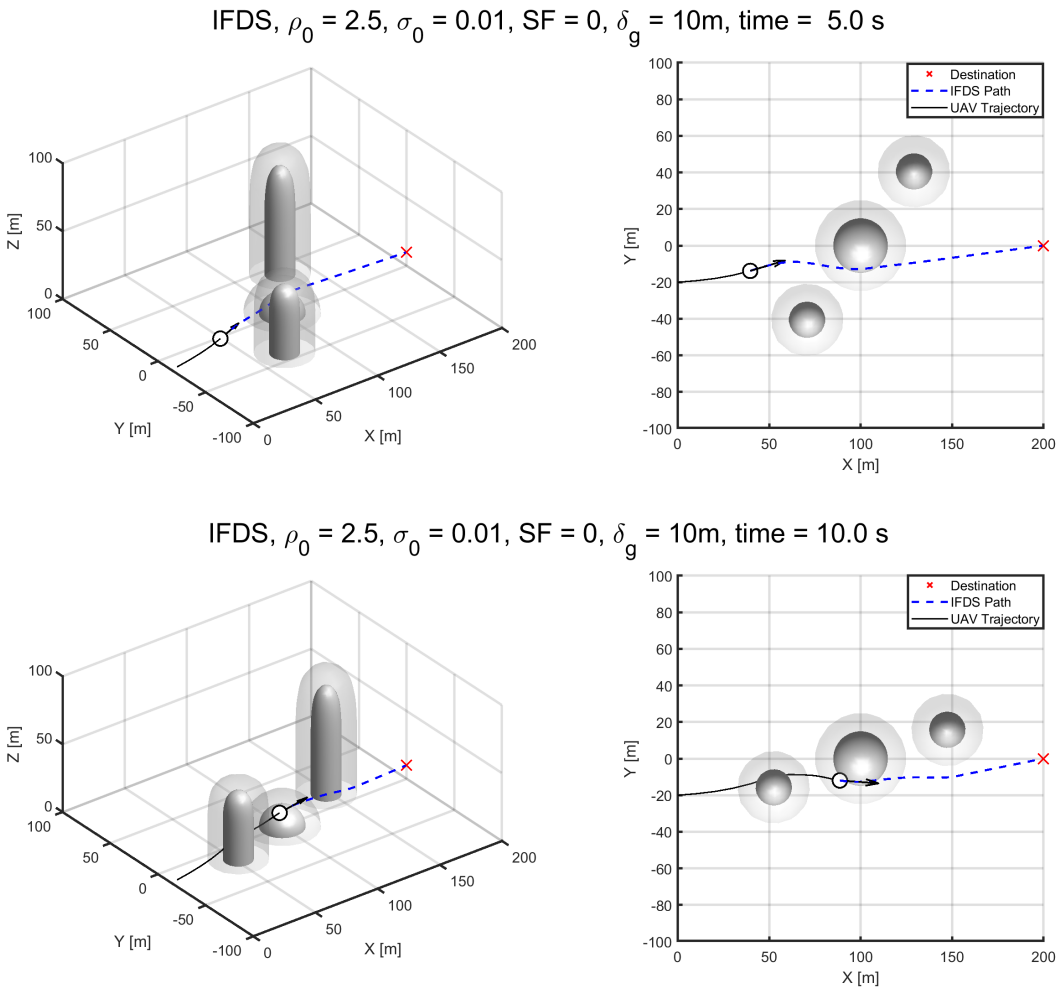

Fig. 13 Scenario 2: Example 1, $t = 5 - 10$ s

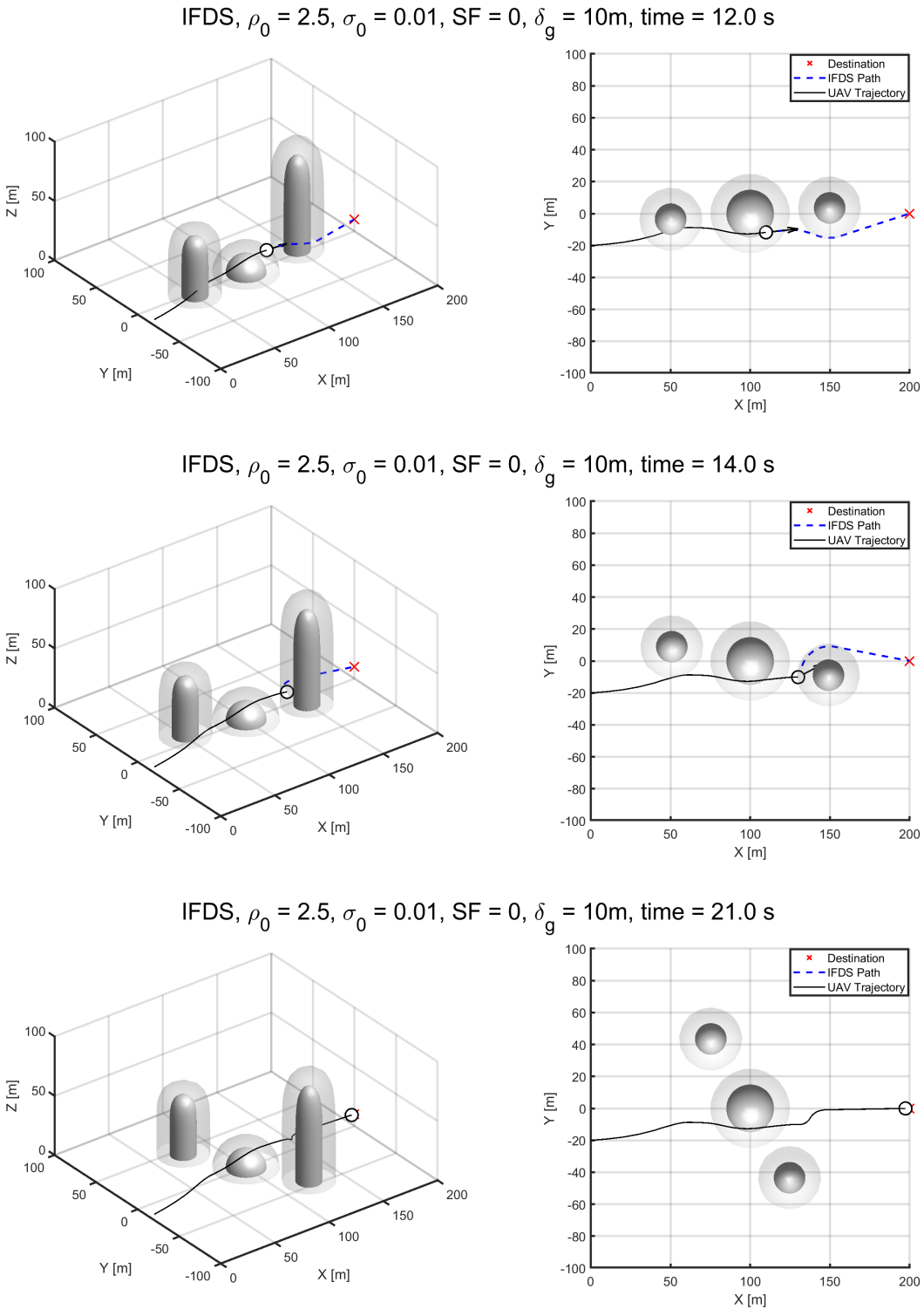

Fig. 14 Scenario 2: Example 1, $t = 12 - 21\ s$

2. Example 2

In contrast, Figs. 15 to 16 illustrate that the adaptive adjustments in the path are not limited solely to lateral movements but also extend longitudinally when dealing with airborne obstacles. In Fig. 15, as the UAV encounters the second obstacle, the path dynamically transitions from the right side to the left side of the obstacle between time $t = 4$ and 6 s. Furthermore, the path demonstrates three-dimensional adaptability for the airborne obstacle during the interval of time $t = 13$ to 18 s. This distinctly highlights the dynamic autorouting algorithm's remarkable flexibility in response to varying obstacle configurations and trajectories.

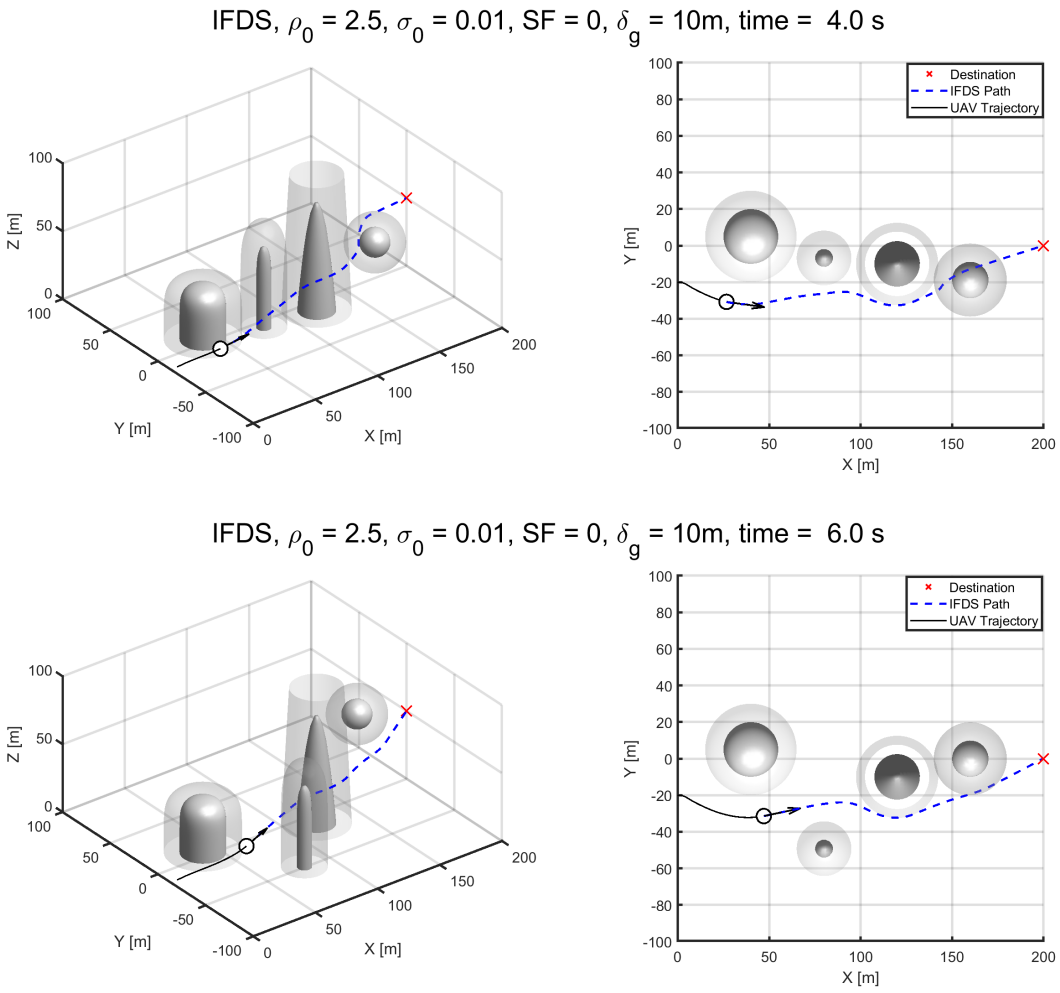

Fig. 15 Scenario 2: Example 2, $t = 4 - 6$ s

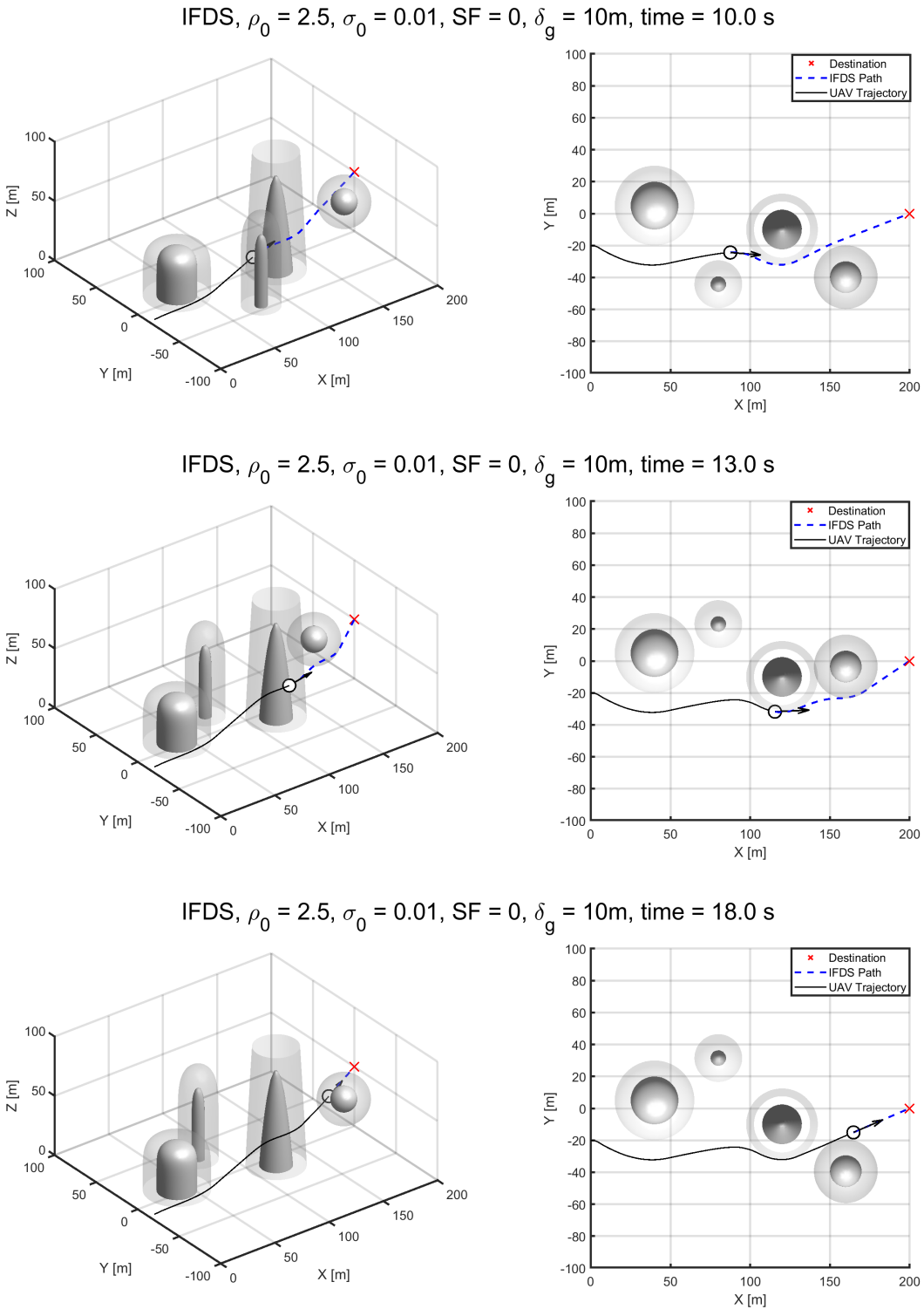

Fig. 16 Scenario 2: Example 2, $t = 10 - 18\ s$

G. Real-time Performance

The real-time performance of the proposed dynamic autorouting algorithm is evaluated. The specifications of the computer employed to conduct the simulations in this study are detailed in Table 8. The computation time is quantified using MATLAB's built-in timing functions *tic* and *toc*. To ensure accurate measurements, each test has been executed multiple times, as the background tasks of the computer can lead to slight variations in timing measurements. The average maximum computation times for the tests are summarized in Table 9. Overall, all tests are completed in milliseconds, affirming the real-time capabilities of the dynamic autorouting algorithm. The computation time is influenced by the number of obstacles present in the field. In scenario 2, with four obstacles, the calculation duration reaches up to 67 ms, while in scenario 1 with three obstacles, it takes only 35.8 ms.

It is important to highlight that in the first scenario, where there are no moving obstacles, the IFDS algorithm executes only once under the assumption that the path is pre-calculated before UAV departure. Thus, this calculation does not impose a burden on the UAV's onboard processor, and the UAV only follows this path throughout its flight. Conversely, when dealing with dynamic obstacles in the second scenario, the IFDS algorithm recalculates the path from the UAV's current position every second. Although this continuous update ensures that the path adapts to the UAV's real-time position, it does impact the workload on the onboard computer. Nonetheless, the path length gradually diminishes as time progresses, leading to reduced computing times and an overall decrease in the average computation time. A detailed breakdown of the computing times for each time step in the second scenario can be found in Fig. 17.

Table 8 Laptop specifications.

Device specifications	
Processor	AMD Ryzen 9 5900HX 3.30 GHz
Installed RAM	16 GB
System type	64-bit, x64-based processor
Operating system	Windows 11 Home 22H2
Program	MATLAB R2022b

Table 9 Averaged maximum computed time of the IFDS with safeguarding in various scenarios.

Scenario	Type of obstacles	Number of obstacles	Average maximum computing time (ms)
1	Static	3	35.8
2 (Example 1)	Static + Dynamic	3	40.7
2 (Example 2)	Static + Dynamic	4	67.0

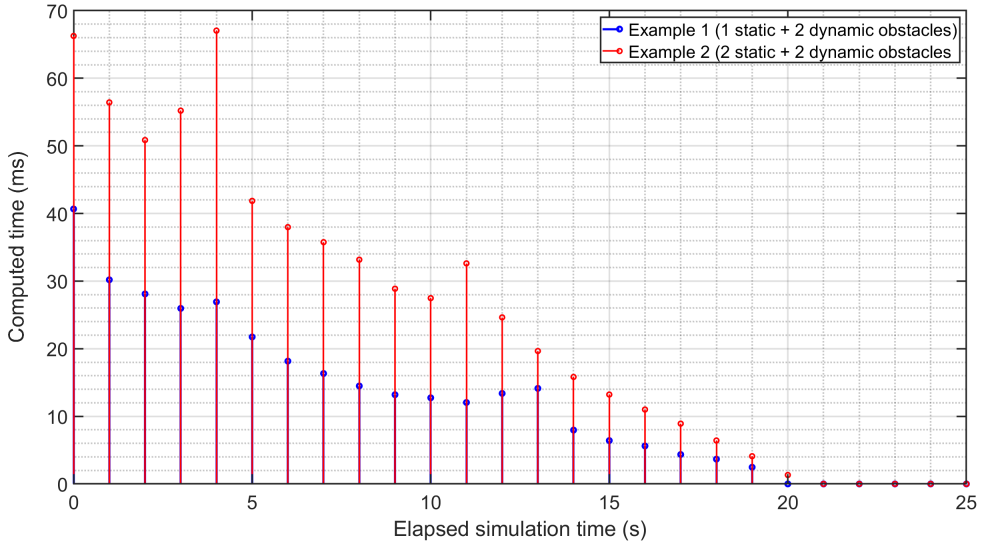

Fig. 17 Computed time of the IFDS with safeguarding for avoiding dynamic and static obstacles.

VII. Conclusion

This paper focuses on dynamic path planning in a three-dimensional complex environment utilizing the Interfered Fluid Dynamical System (IFDS). The IFDS algorithm has handled the main drawback of many novel path planning methods — the slow computing time, which is difficult to improve especially with the limitation of the onboard computer. The IFDS algorithm's effectiveness is widely acknowledged in existing literature, offering attributes like high computational efficiency, real-time feasibility, and practical applicability. To further enhance its practicality, the safeguarding function has been introduced. The safeguarding function guarantees a consistent, safe distance between the generated path and object surfaces, regardless of the tuning of the IFDS algorithm. In addition, the path following using a three-dimensional carrot-chasing algorithm (CCA3D) has been adopted to generate a feasible UAV trajectory. This algorithm guarantees trajectory adherence to UAV kinematic constraints, encompassing parameters like maximum turning radius, pitch, and yaw angle limits.

A simulation study has been conducted for two scenarios, featuring both multiple static obstacles and dynamic obstacles within a $200 \times 200 \times 200$ m space to evaluate the performance of the proposed dynamic autorouting algorithm. The results of the simulation study reveal that the algorithm can effectively handle static and dynamic obstacles together. Real-time execution also remains achievable even in scenarios featuring rapidly moving obstacles on both the ground and in the air. The average maximum computing time consistently stays within milliseconds and is directly influenced by the number of obstacles present in the field. Impressively, it consistently navigates UAVs to their intended destinations without collision incidents while adaptively avoiding moving obstacles while maintaining the desired safety distance from the obstacles.

References

[1] Hueso, J. S., Mondal, S., Tsourdos, A., and Chadwick, A., "Real-Time Collision Avoidance Trajectory Planner Using Generalized Vector Explicit Guidance," 2023. https://doi.org/10.2514/6.2023-1734, URL https://arc.aiaa.org/doi/10.2514/6.2023-1734.

[2] Mondai, S., and Padhi, R., "Formation Flying using GENEX and Differential geometric guidance law," *IFAC-PapersOnLine*, Vol. 48, No. 9, 2015, pp. 19–24.

[3] Ohlmeyer, E. J., and Phillips, C. A., "Generalized Vector Explicit Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 2, 2006, pp. 261–268. https://doi.org/10.2514/1.14956, URL https://doi.org/10.2514/1.14956.

[4] Mondal, S., and Padhi, R., "Selection of optimal time-to-go in generalized vector explicit guidance," *2015 IEEE Conference on Control Applications (CCA)*, IEEE, 2015, pp. 756–761.

[5] Yin, C., Xiao, Z., Cao, X., Xi, X., Yang, P., and Wu, D., "Offline and Online Search: UAV Multiobjective Path Planning Under Dynamic Urban Environment," *IEEE Internet of Things Journal*, Vol. 5, 2018, pp. 546–558. https://doi.org/10.1109/JIOT.2017.2717078.

[6] Wu, C., Huang, X., Luo, Y., Leng, S., and Wu, F., "An Improved Sparse Hierarchical Lazy Theta Algorithm for UAV Real-Time Path Planning in Unknown Three-Dimensional Environment," *International Conference on Communication Technology Proceedings, ICCT*, Vol. 2020-October, 2020, pp. 673–677. https://doi.org/10.1109/ICCT50939.2020.9295690.

[7] Elhoseny, M., Tharwat, A., and Hassanien, A. E., "Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm," *Journal of Computational Science*, Vol. 25, 2018, pp. 339–350. https://doi.org/10.1016/j.jocs.2017.08.004.

[8] Shao, S., Peng, Y., He, C., and Du, Y., "Efficient path planning for UAV formation via comprehensively improved particle swarm optimization," *ISA Transactions*, Vol. 97, 2020, pp. 415–430. https://doi.org/10.1016/j.isatra.2019.08.018, URL https://linkinghub.elsevier.com/retrieve/pii/S0019057819303532.

[9] Jamshidi, V., Nekoukar, V., and Refan, M. H., "Real time UAV path planning by parallel grey wolf optimization with align coefficient on CAN bus," *Cluster Computing*, Vol. 24, 2021, pp. 2495–2509. https://doi.org/10.1007/s10586-021-03276-6.

[10] Ding, Q., and Xu, X., "Improved GWO Algorithm for UAV Path Planning on Crop Pest Monitoring," *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 7, 2022, p. 5. https://doi.org/10.9781/ijimai.2022.07.002.

[11] Liu, X., Long, Y., Li, T., and Huang, T., "3D Path Planning of Unmanned Aerial Vehicle Based on Improved Grey Wolf Optimization Algorithm," 2023, pp. 4285–4290. https://doi.org/10.1109/CAC57257.2022.10055505.

[12] Wang, H., Lyu, W., Yao, P., Liang, X., and Liu, C., "Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system," *Chinese Journal of Aeronautics*, Vol. 28, 2015, pp. 229–239. https://doi.org/10.1016/J.CJA.2014.12.031.

[13] Yao, P., Wang, H., and Su, Z., "UAV feasible path planning based on disturbed fluid and trajectory propagation," *Chinese Journal of Aeronautics*, Vol. 28, 2015, pp. 1163–1177. https://doi.org/10.1016/J.CJA.2015.06.014.

[14] Yao, P., Wang, H., and Su, Z., "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerospace Science and Technology*, Vol. 47, 2015, pp. 269–279. https://doi.org/10.1016/J.AST.2015.09.037.

[15] Wang, Y., Wang, H., Wen, J., Lun, Y., and Wu, J., "Obstacle avoidance of UAV based on neural networks and interfered fluid dynamical system," Institute of Electrical and Electronics Engineers Inc., 2020, pp. 1066–1071. https://doi.org/10.1109/ICUS50048.2020.9274988.

[16] Yao, P., and Zhao, S., "Three-Dimensional Path Planning for AUV Based on Interfered Fluid Dynamical System under Ocean Current (June 2018)," *IEEE Access*, Vol. 6, 2018, pp. 42914–42916. https://doi.org/10.1109/ACCESS.2018.2861468.

[17] Celestini, D., Primatesta, S., and Capello, E., "Trajectory Planning for UAVs Based on Interfered Fluid Dynamical System and Bézier Curves," *IEEE Robotics and Automation Letters*, Vol. 7, 2022, pp. 9620–9626. https://doi.org/10.1109/LRA.2022.3191855.

[18] Zhang, Y., and Wang, H., "Adaptive Interfered Fluid Dynamic System Algorithm Based on Deep Reinforcement Learning Framework," *Lecture Notes in Electrical Engineering*, Vol. 861 LNEE, 2022, pp. 1388–1397. https://doi.org/10.1007/978-981-16-9492-9_139/TABLES/1, URL https://link.springer.com/chapter/10.1007/978-981-16-9492-9_139.

[19] Fan, S., and Chen, M., "Route planning of agricultural plant protection unmanned helicopter based on interfered fluid dynamical system," *Proceedings - 9th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2017*, Vol. 2, 2017, pp. 114–117. https://doi.org/10.1109/IHMSC.2017.141.

[20] Wu, J., Wang, H., Li, N., and Su, Z., "Formation Obstacle Avoidance: A Fluid-Based Solution," *IEEE Systems Journal*, Vol. 14, 2020, pp. 1479–1490. https://doi.org/10.1109/JSYST.2019.2917786.

[21] Wu, J., Wang, H., Liu, Y., Zhang, M., and Wu, T., "Learning-based fixed-wing UAV reactive maneuver control for obstacle avoidance," *Aerospace Science and Technology*, Vol. 126, 2022, p. 107623. https://doi.org/10.1016/j.ast.2022.107623, URL www.elsevier.com/locate/aescte.

[22] "Path following Control for Unmanned Aerial Vehicle Based on Carrot Chasing Algorithm and PLOS," *Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Information Systems, ICAIIS 2020*, 2020, pp. 571–576. https://doi.org/10.1109/ICAIIS49377.2020.9194843.

[23] Bhadani, R., "Path Planning of Unmanned System using Carrot-chasing Algorithm," *CoRR*, Vol. abs/2012.13227, 2020. URL https://arxiv.org/abs/2012.13227.

[24] Tabatabaei, S. A. H., Yousefi-koma, A., Ayati, M., and Mohtasebi, S. S., "Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles," *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, 2015, pp. 784–788. https://doi.org/10.1109/ICRoM.2015.7367882.