Guidance, Navigation and Control
Vol. 3, No. 1 (2023) 2350004 (24 pages)
© Technical Committee on Guidance, Navigation and Control, CSAA and
World Scientific Publishing Co.
DOI: 10.1142/S2737480723500048

Multi-UAV Cooperative Hunting in Cluttered Environments Considering Downwash Effects

Shichen Fan* and Hugh H.-T. Liu†

Institute for Aerospace Studies
University of Toronto, 4925 Dufferin Street
North York, Ontario, Canada M3H 5T6
**shichen.fan@mail.utoronto.ca*
†*hugh.liu@utoronto.ca*

Received 9 February 2023
Revised 23 February 2023
Accepted 27 February 2023
Published 25 April 2023

This paper presents a novel solution to the three-dimensional (3D) cooperative hunting of multiple drones that deals with surrounding a target simultaneously while navigating around obstacles in the cluttered dynamic 3D environment. Meanwhile, drones avoid the airflow downwash force created by the spinning propellers on unmanned aerial vehicles (UAVs) and their effect on the other UAVs. This solution consists of a 3D Simultaneous Encirclement strategy, the cooperative hunting objective with a novel revised particle swarm optimization (PSO*) path planning algorithm, a flocking theory-inspired obstacle avoidance algorithm, and a cascade PI controller. Simulation results with varying conditions were carried out to validate the effectiveness of the proposed solution by successfully taking care of the downwash effects, and having multiple hunter UAVs hunt and encircle a moving or stationary target in a dynamic or static obstacle-rich cluttered environment.

Keywords: Cooperative hunting; downwash effect; UAS.

1. Introduction

UAVs have become a rapidly growing market over the past decade because of their lower cost and lower human risk, compared with traditional human-crewed aircraft. Its flight has different levels of autonomy from the remote control by a human operator to complete autonomous control by onboard computers or a ground station. UAVs have applications in surveillance,[1] search and rescue,[2] fire fighting,[3] intruder capturing[4] and imaging[5] and so on. Cooperative hunting is a pack hunting behavior observed in nature where a group of hunters coordinate their behavior to increase their chance of hunting their prey. When cooperatively hunting, it does not only consider hunter UAVs and a single target UAV, but it also considers a complex

*Corresponding author.

environment. Obstacles, including static obstacles such as buildings and trees and dynamic obstacles like moving vehicles and flying creatures, are present for the UAVs to avoid. In addition, the target behavior and its moving path are unknown and unpredictable. In a multi-agent cooperative hunting scenario, hunter agents are likely to act fearless and actively avoid obstacles and other agents, whereas the target is likely to act randomly like a lost animal without awareness of the hunter UAVs.[6] Hence, for the hunter UAVs to cooperate to hunt in such a complex environment, they must avoid collisions, path around obstacles, and encircle and trace the target. Besides, the downwash effect that exists in a three-dimensional intense flying could lead hunting activity to failure.

Many researches and novel methods of cooperative hunting have been proposed, including hybrid attention-oriented experience replay multi-agent deep deterministic policy gradient (HAER-MADDPG),[7] Differential game approach,[8] a new strategy inspired by the behavior of wolves in hunting,[9] a scheme based on k-WTA and wolf-pack-particle-based model,[10] quadrotor location-based target hunting with formation strategy,[11] formation control based hunting method,[12] a method for multiple unmanned surface vessels collaborative search and hunting,[13] the Deviated Pure Pursuit methodology based strategy,[14] a wolf scouting behavior-inspired cooperative search algorithm,[15] and potential field-enhanced reinforcement learning based strategy.[16] However, these methods so far only focus on two-dimensional (2D) cooperative hunting scenarios. Aerodynamic effects such as the downwash effect and the determination of goal locations make the cooperative hunting problem challenged. This paper's UAV application is 3D cooperative hunting to realize multiple UAVs' coordination to hunt and surround a moving/static target.

In a 3D multi-UAV cooperative hunting event, downwash impact needs to be considered while planning trajectories for each quadrotor. This downwash impact is the force that is created from a high volume of airflow generated from a high-speed propeller rotation.[17] When a trajectory cross scenario exists, the quadrotor flying beneath can experience a downwash force from the quadrotor above. This downwash force may push the quadrotor beneath downward. When the quadrotor is close to the ground, this downwash force may cause a failure of flying by pushing the quadrotor down and hitting the ground. Hence, this downwash impact should be taken with caution in a 3D multi-UAV cooperative hunting event. Very few research works have been done on this downwash effect. Cai Lei proposed a generative adversarial network-based 3D cooperative hunting strategy.[18] Cao Xiang introduced a dynamic prediction of target trajectory-based multi-AUV hunting algorithm.[19] However, both these solutions to 3D cooperative hunting problems do not take downwash effects into consideration. Chih-Chun Chen derives a downwash model from simulating this effect.[17] Figure 1 simulates the airflow created by a propeller.[17]

Chen adopted the downwash force model to predict the downwash impact on the quadrotor UAVs beneath. The downwash force and downwash acceleration were developed and simulated in Chen's paper.[17] By observing the deceleration in the scenario when the top UAV is close to the bottom UAV, a significantly large enough

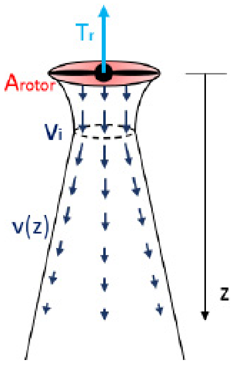


Fig. 1. Air flow created by a propeller.

downwash deceleration is exerted on the bottom UAV that could cause a failure by pushing the bottom UAV to the ground when trajectory crossing of two UAVs exists, leading to a crash. This phenomenon emphasizes that the downwash force has to be considered when planning a trajectory and encircling a target. In other words, the targeted work of this paper is to develop a cooperative hunting solution that avoids the potential failure caused by this downwash impact.

As a continuity of Chen's research on downwash impact, this paper used this downwash force model to predict and add the downwash impact to simulation environments to get closer-to-reality simulation results. Also, this downwash impact is considered when designing the solution to multi-UAV cooperative hunting in three-dimensional space.

In this paper, a novel solution to the 3D cooperative hunting problem when considering downwash impact is presented. This solution consists of a revised PSO-based path planning algorithm, a flocking-based collision-avoidance algorithm, a simultaneous encirclement strategy, and a cascade PI controller. The downwash impact derived from Chen's paper is considered when designing this solution.

The rest of this paper is organized as follows. Section 2 presents a revised PSO path planning algorithm to plan trajectories for hunter UAVs to move toward an escaping target UAV and a comparative study with standard PSO. Section 3 derives a flocking-based collision-avoidance algorithm that generates a repulsive acceleration within a cylindrical range around the hunter UAV itself to prevent potential collision and downwash impact from its neighboring hunter UAVs. Section 4 proposes a simultaneous encirclement strategy that determines the goal locations for hunter UAVs and achieves a simultaneous encirclement of an escaping target UAV. Section 5 includes a Cascade PI Controller so that the hunter UAVs can reach the generated next locations and follow the planned trajectories. Section 6 demonstrates the simulation results of the cooperative hunting of multi-UAV in static/dynamic cluttered environments while the downwash effect exists. Section 7 does comparative studies between the proposed solution and the existing solution. Section 8 concludes.

2. Path Planning Algorithm

This section details the proposed revised Particle Swarm Optimization (PSO*) path planning algorithm, which is an essential module of the solution to multi-agent cooperative hunting in 3D. After that, an introduction to the standard Particle Swarm Optimization (PSO) path planning algorithm is included in this section, followed by comparative studies between the two algorithms to highlight the necessity of improving the standard PSO algorithm. As the goal of the hunting problem in a dynamic/static environment is to find a sub-optimal hunting trajectory while sacrificing as little computational cost as possible, the comparative studies focus on the computational cost and trajectory length of the algorithm performance. In this section, the simulation results of these two algorithms are included and compared from these two perspectives.

2.1. *PSO* path planning algorithm*

The path planning algorithm proposed here is the PSO* algorithm derived from the standard PSO algorithm to improve computational efficiency. Unlike the standard PSO algorithm which generates random particles around the agent's current location, the proposed PSO* path planning algorithm draws particles around intermediate waypoints. The concept of PSO* is to find a sub-optimal trajectory at a low computational cost. In a cooperative hunting problem, a trajectory is said to be sub-optimal if the trajectory is sub-optimized and of close-to-shortest. This path-planning algorithm is a population-based stochastic optimization technique that finds a solution to problems based on the movement and revolution of a particle swarm. Information is exchanged among every particle, as shown in Fig. 2. This example presents the exchange of information in a 4-particle case. This manner of the exchange of information can be expanded to more particles.

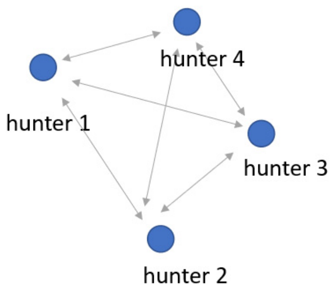


Fig. 2. Information exchanges of PSO* in a 4-particle case.

A straight line is drawn connecting hunter agents' current positions at time t, $\mathbf{P}_{\mathrm{s}}(\mathbf{t}) = [\mathbf{X}_{\mathrm{s}}(\mathbf{t}), \mathbf{Y}_{\mathrm{s}}(\mathbf{t}), \mathbf{Z}_{\mathrm{s}}(\mathbf{t})]$, and their target locations at time t, $\mathbf{P}_{\mathbf{t}}(\mathbf{t}) = [\mathbf{X}_{\mathbf{t}}(\mathbf{t}), \mathbf{Y}_{\mathbf{t}}(\mathbf{t}), \mathbf{Z}_{\mathbf{t}}(\mathbf{t})]$, then pick a number, n_{inter}, of equally spaced intermediate waypoints at time t, $\mathbf{P_{int}(t)} = [\mathbf{X_{int}(t)}, \mathbf{Y_{int}(t)}, \mathbf{Z_{int}(t)}]$ on this line. Within a range from each intermediate waypoint, $\mathbf{P_{int}(t)}$,

$$\mathbf{P_{int}(t)} = \mathbf{P_s(t)} + \frac{(\mathbf{P_t(t)} - \mathbf{P_s(t)})}{(n_{\text{inter}} + 1)} \times k, \quad k = 1, 2, \ldots, n_{\text{inter}} \tag{1}$$

radius as indicated by Eq. (2), a number, n_{particle}, of random particles are generated. This procedure is graphically described by Fig. 3. This way of particle generation makes particle swarm all initialized within a certain range of the direct path from the hunter agent from its goal location on purpose so that it can save efforts of finding a potential trajectory by limiting the area for the generation of particles. A potential drawback of this method is that the trajectory created might not be the true optimal trajectory. This is because no particle is generated outside $\mathbf{r}$ from the intermediate waypoints. Hence, the trajectory generated shall be considered a sub-optimal flight path.

$$\mathbf{r} = \frac{\|(\mathbf{X_t(t)}, \mathbf{Y_t(t)}, \mathbf{Z_t(t)}) - (\mathbf{X_s(t)}, \mathbf{Y_s(t)}\mathbf{Z_s(t)})\|}{2 \times (1 + n_{\text{inter}})}. \tag{2}$$

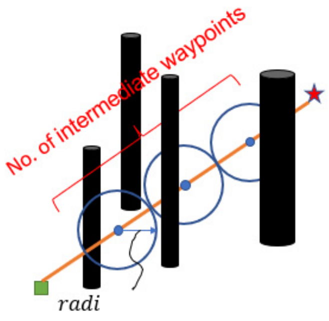


Fig. 3. Particle generation of PSO*.

Every randomly generated around each intermediate waypoint is treated as particle set i. The Cartesian coordinates and velocities of particle set i at time t are $\mathbf{P_i(t)} = [\mathbf{X_i(t)}, \mathbf{Y_i(t)}, \mathbf{Z_i(t)}]$ and $\mathbf{V_i(t)} = [\mathbf{V_{xi}(t)}, \mathbf{V_{yi}(t)}, \mathbf{V_{zi}(t)}]$. The position of the particle set i

$$\mathbf{P_i(0)} = \mathbf{P_{int}(0)} + \mathbf{V_{rad}} \times \begin{Bmatrix} \cos\boldsymbol{\phi} \times \sin\boldsymbol{\theta} \\ \sin\boldsymbol{\phi} \times \sin\boldsymbol{\theta} \\ \cos\boldsymbol{\theta} \end{Bmatrix} \tag{3}$$

is initialized by spherical coordinates with $\mathbf{P_{int}(0)}$ as the center. $\mathbf{V_{rad}}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ are random floating numbers in the range of $[\mathbf{0.0}, \mathbf{radius})$, $[0.0, \pi)$ radians, and $[0.0, 2\pi)$ radians, respectively. The velocity of the particle set i, $\mathbf{V_i(t = 0)}$, is initialized to be zeros.

With the generated particle sets, a list of trajectories is initialized by doing 3D spline interpolations between the start location, every particle set, and the goal location. At every iteration, the velocity $\mathbf{V_i}$ and position $\mathbf{P_i}$ of each generated particle set i are updated according to Eqs. (4) and (5). The velocity $\mathbf{V_i}$ and position $\mathbf{P_i}$ of each generated particle are updated according to the following equations:

$$\mathbf{V_i}(t+1) = \omega\mathbf{V_i}(t) + c_1r_1(\mathbf{pbest(i,t)} - \mathbf{P_i}(t)) + c_2r_2(\mathbf{gbest(t)} - \mathbf{P_i}(t)), \tag{4}$$

$$\mathbf{P_i}(t+1) = \mathbf{P_i}(t) + \mathbf{V_i}(t+1), \tag{5}$$

where $i = 1, 2, 3, \ldots, N$ and N is the total number of particle sets; $t = 1, 2, 3, \ldots,$ Max_Iteration.

$\mathbf{pbest(i,t)}$ stands for the optimal local solution of the particle set i while $\mathbf{gbest(t)}$ represents the optimal global solution among all particle sets. r_1, r_2, used to avoid premature convergences, are random numbers selected from [0,1]. c_1 and c_2 are positive constants. c_1 weighs the importance of the particle set is own previous experiences. c_2 weighs the importance of the global learning of the swarm. When advancing particle sets by updating their velocities and positions, local optimal and globally optimal solutions are considered for the hunter agent to avoid getting stuck into the local optimum. ω stands for the inertia weight, a value chosen from [0,1]. A lower ω means an increased chance of finding a global optimum but is more time-consuming.

A cost function, $\mathbf{J}$, is created to find the trajectory of the shortest length and the least obstacle collision possibility. The trajectory is separated into a number of line segments. The cost of violation is calculated based on the Euclidean distance between the lth line segment and the kth obstacle located within hunter agent j's sensing range d_j by Eq. (7). r_k represents the radius of the kth obstacle. n_o is the number of obstacles sensed by the hunter agent j. n_l serves as the total number of line segments. $\mathbf{L}$ stands for a vector of the trajectory lengths of hunter agents. β is a constant number used to exaggerate the impact of $\mathbf{C_{vio}}$.

$$\mathbf{J} = \mathbf{L} + \beta * \mathbf{C_{vio}}, \tag{6}$$

where $\mathbf{C_{vio}}$ is a vector that consists of the cost of violation of the ith particle set $C_{\text{vio}}(i)$.

$$C_{\text{vio}}(i) = \sum_{k=1}^{n_o} \text{mean}\left(\max\left\{1 - \frac{d_l}{r_k}, 0\right\}\right), \tag{7}$$

where n_{par} stands for the total number of particle sets, $i = 1, 2, \ldots, n_{\text{par}}$, and $l = 1, 2, \ldots, n_l$.

This cost of violation takes any potential collisions with obstacles into consideration. Equation (7) is graphically explained in Fig. 4.

Tangent lines are drawn to the circle around the first intermediate waypoint from the agent's current position as shown in Fig. 5. When there is a nearby obstacle that both tangent lines are through the obstacle, the current PSO* solution does not work efficiently in this scenario. This phenomenon is because all particles are generated within the range of *radius* from the intermediate waypoints. One further if-else step is added to the current PSO* algorithm to deal with this scenario. Specifically, each

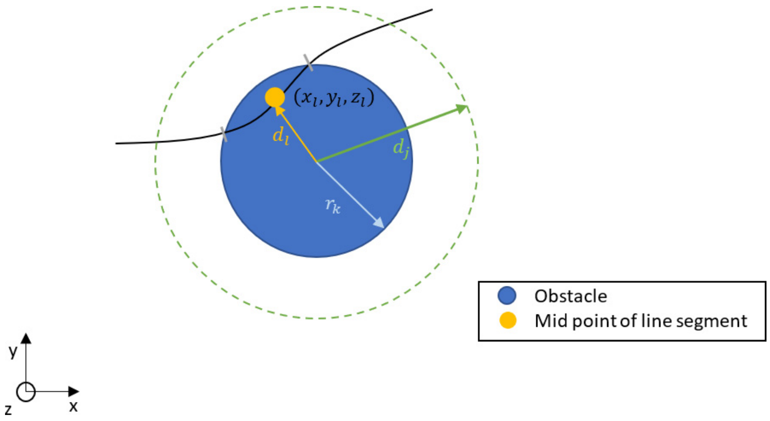


Fig. 4. Cost of violation.

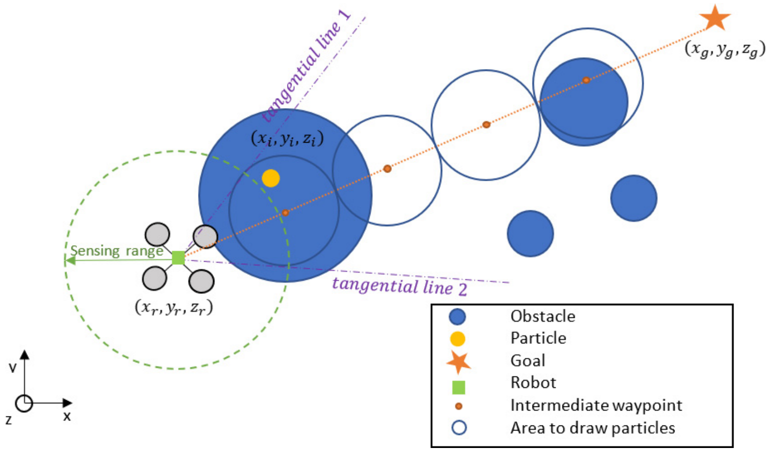


Fig. 5. Tangent lines.

agent checks if there is any obstacle, that of size large enough that is through by both tangent lines, laying on its moving way toward the goal location, within the agent's sensing range. If yes, the agent will act as moving by the tangent line of this obstacle until passing by it in the first place, then use the second step of the proposed PSO* algorithm to plan its trajectory towards its goal location, as shown in Fig. 6.

position limits are

$$\{[x_{\min}, x_{\max}], [y_{\min}, y_{\max}], [z_{\min}, z_{\max}]\}$$

and speed limits are

$$\{[v_{x,\min}, v_{x,\max}], [v_{y,\min}, v_{y,\max}], [v_{z,\min}, v_{z,\max}]\}.$$

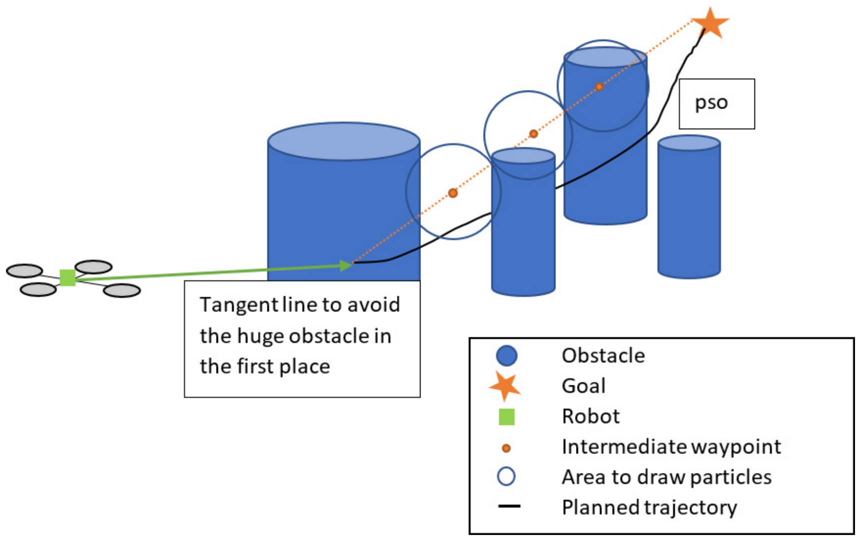


Fig. 6. Trajectory developed by PSO*.

These limits are considered while updating particles' position and velocity. When particle set i's next positions $\mathbf{P_i(t+1)}$ or next velocities $\mathbf{V_i(t+1)}$ exceed these limits, particle set i updates itself to the corresponding limits.

A flow chart describing how the PSO* path planning algorithm works is shown in Fig. 7.

2.2. *Standard PSO: Comparative algorithm*

Different from the proposed PSO* path planning algorithm which draws particles around intermediate waypoints, the standard PSO algorithm generates random particles, (x'_{r_i}, y'_{r_i}), within the range of maximum step size, $V_{r_{\max}}$, around agent i's initial position (x_r, y_r), as shown in Fig. 8. Same as PSO*, the velocity $\mathbf{V_i}$ and position $\mathbf{P_i}$ of each generated particle are updated according to Eqs. (4) and (5). Inertia weight ω is determined by a linear equation Eq. (10), where ω_{start} and ω_{end} represent the start value and end value of the inertia weight ω.

The Cartesian coordinates of the randomly generated particles (x'_{r_i}, y'_{r_i}) are

$$x'_{r_i} = x_r + R * \cos\theta_i, \tag{8}$$

$$y'_{r_i} = y_r + R * \sin\theta_i, \tag{9}$$

where R is a random floating number in the range of [0.0, radi), θ_i is a random floating number in the range of $[0.0,\, 2\pi)$ radians

$$\omega = \omega_{\text{start}} - \frac{(\omega_{\text{start}} - \omega_{\text{end}})}{\text{Max\_Iteration}} * t. \tag{10}$$

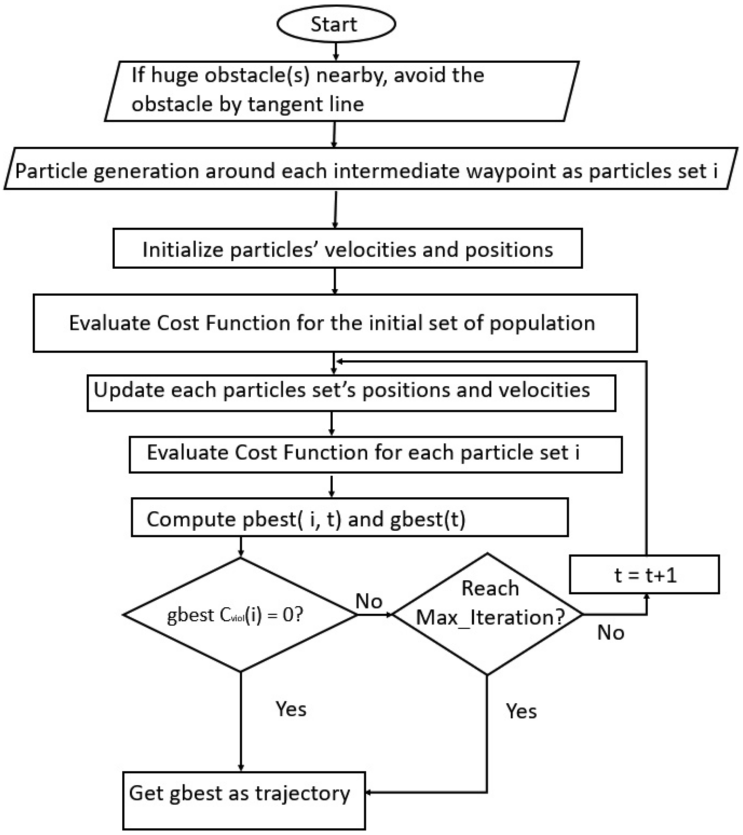

Start
If huge obstacle(s) nearby, avoid the obstacle by tangent line
Particle generation around each intermediate waypoint as particles set i
Initialize particles' velocities and positions
Evaluate Cost Function for the initial set of population
Update each particles set's positions and velocities
Evaluate Cost Function for each particle set i
Compute pbest(i, t) and gbest(t)
gbest C_viol(i) = 0?
No
Reach Max_Iteration?
No
t = t+1
Yes
Yes
Get gbest as trajectory

Fig. 7. Flowchart of PSO*.

The cost function used here is

$$J = \sqrt{(x'_{r_i} - x_r)^2 + (y'_{r_i} - y_r)^2} + \sqrt{(x'_{r_i} - x_g)^2 + (y'_{r_i} - y_g)^2} + \text{Penalty}(i), \quad (11)$$

where goal location is (x_g, y_g).

All obstacles are examined for potential collisions with both the straight path from the robot to the particle and the straight path from the particle to its goal location. If any obstacle is on the straight paths, these paths will cut the obstacle into two separate arc lengths. The penalty of particle i checks every obstacle in the environment. If any obstacle is on the direct lines, penalty Penalty(i) selects the shorter arc length of every qualified obstacle and sums them up. It is assumed that all obstacles are neither adjacent nor overlapped in the environment. Hence, the equation of Penalty(i) is

$$\text{Penalty}(i) = \sum_{\text{obstacles}} \min\{\text{arc}_j | j \in 1, 2\} \quad (12)$$

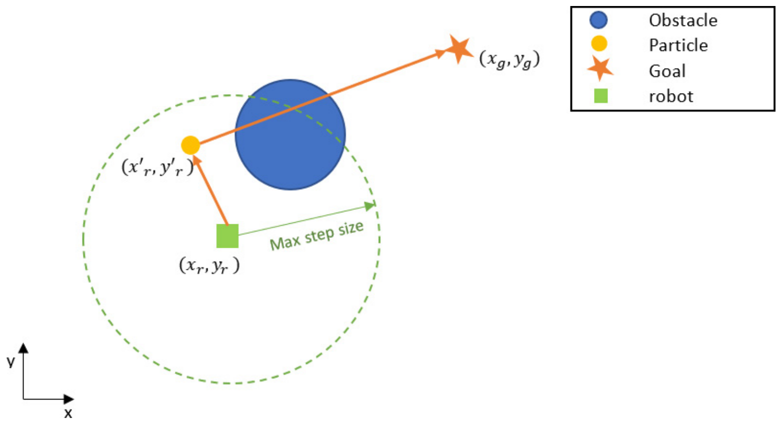


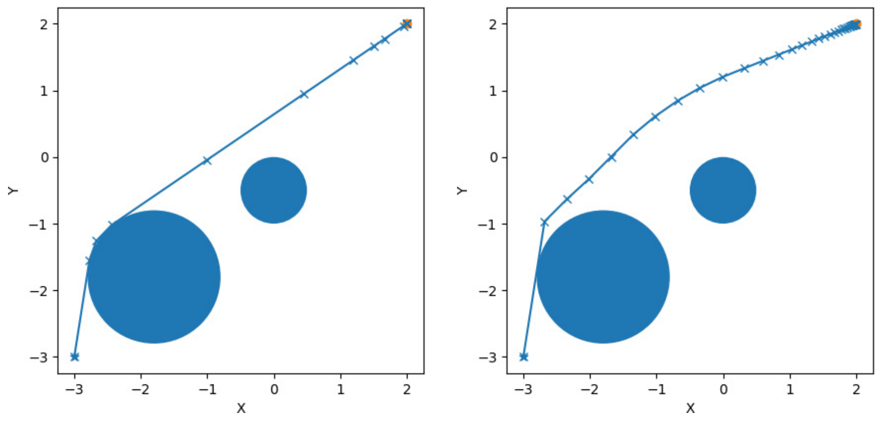
Fig. 8. Standard PSO.

2.3. *Comparative studies*

Simulation results are collected when obstacles are at a different distance from the agent while the agent's starting location and goal location keep the same.

Single obstacle scenario	Trajectory length ratio		Wall time ratio	
$xs = 0, ys = 0, xt = 0, yt = 0$	Standard PSO	PSO*	Standard PSO	PSO*
xobs = 1, yobs = 1, robs = 1	1	1	45.9	1
xobs = 2, yobs = 2, robs = 1	0.98	1	74.9	1
xobs = 3, yobs = 3, robs = 1	0.98	1	45.0	1
xobs = 6, yobs = 6, robs = 1	0.96	1	133.6	1

By comparing execution time and trajectory length when the obstacle is at different distances from the hunter agent and its goal location, we observed that PSO* has much higher computational efficiency than the standard PSO without sacrificing too much trajectory length. When increasing the number of obstacles as shown in Fig. 9, same trend was observed. The significant improvement in computational time is because the ways of particle generation are quite different. The standard version of the particle swarm optimization path planner initializes particles around the hunter agent's current position without taking the goal location into consideration. This way leaves many particles at the opposite direction, so the computational time spent on these particles is wasted because a particle in the opposite direction creates a trajectory of longer length. PSO* creates particles within a certain range along the straight line connecting the hunter agent's current position and goal location. This method can save lots of computational time because the particles created in this way will likely find shorter lengths of trajectories.



(a) Standard PSO (b) PSO*

Fig. 9. Simulation results when multi-obstacles are in environment.

Trajectory length ratio		Wall time ratio	
Standard PSO	PSO*	Standard PSO	PSO*
0.97	1	157.3	1

3. Collision-Avoidance Algorithm

A collision-avoidance controller is developed to solve the downwash impact existing in 3D cooperative hunting. This control law is adopted from the flocking concept proposed by Reza Olfati-Saber.[6] The hunter agents should avoid collisions with nearby flock-mates and attempt to match velocity with nearby flock-mates. This method enables hunter agents to avoid collisions with obstacles in the hunting space and, at the same time, move toward the escaping target agent. As downwash impact can be viewed approximately in the shape of a cylinder, the region where the collision-avoidance algorithm comes into effect adopts the idea of a cylindrical shape. This cylindrical shape is chosen to model the region beneath a UAV that affects other UAVs. The avoidance controller is adopted,[20] as graphically presented in Fig. 10. The cylinders in gray and light blue stand for the active regions, R_{c_i} and R_{c_j}, where this collision-avoidance controller comes into effect. When any of agent i's neighboring agents j invades agent i's active region R_{c_i}, a detection of xy-conflicts is triggered when agent j is within agent i's radial sensing range r_{sens} from agent i on a xy-horizontal plane. Likewise, detection of z-conflicts is triggered when agent j is within axial sensing range h_{sens} from agent i in altitude. When xy-conflicts or z-conflicts are detected, three-dimensional repulsive forces (or accelerations), u_i, are

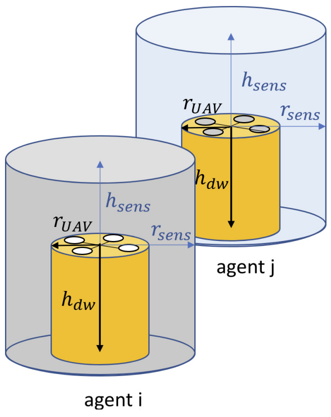


Fig. 10. Reserved cylinder to detect xy-conflict or z-conflict.

generated by agent i so as to repel itself from its neighbors. The yellow cylinders represent the region where downwash effects exist, so they need to be taken care of with caution.

Specifically, this collision-avoidance controller is designed to be

$$\mathbf{u_i} = \mathbf{f_i^g} + \mathbf{f_i^d} \tag{13}$$

that applies a control input made up of two terms. In the first term,

$$\mathbf{f_i^g} = a_1 \sum_{j \in N_i} (\phi_\alpha(\|\mathbf{q_j} - \mathbf{q_i}\|_\sigma) \times \mathbf{n_{i,j}}) \tag{14}$$

is a gradient-based term that repels agent i itself from its neighbors j to avoid potential collisions. a_1 is a scaling factor for the position differences between agent i and its neighbors. $\mathbf{n_{i,j}}$ is the unit vector along the line connecting agent i's position $\mathbf{q_i}$ to agent j's position $\mathbf{q_j}$. The repulsive force ϕ_α is determined by position differences Δp, and

$$\phi_\alpha(\Delta p) = \rho\left(\frac{\mathbf{\Delta p}}{\mathbf{d}_\beta}\right) \times (\sigma(\mathbf{\Delta p} - \mathbf{d}_\beta) - 1), \tag{15}$$

where

$$\sigma(\mathbf{z}) = \frac{\mathbf{z}}{\sqrt{1 + \mathbf{z}^2}} \tag{16}$$

To provide agent i a smooth repulsion, bump function $\rho(\mathbf{\Delta p})$ is designed. It consists of two terms

$$\rho_{(\Delta p_{\text{hor}})} = \begin{cases} 1, & \Delta p_{\text{hor}} \in [0, r_{\text{bet}}), \\ \frac{1}{2} \times \left[1 + \cos\left(\pi \times \frac{\Delta p_{\text{hor}} - r_{\text{bet}}}{1 - r_{\text{bet}}}\right)\right], & \Delta p_{\text{hor}} \in [r_{\text{bet}}, 1], \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

$$\rho_{(\Delta p_{\text{ver}})} = \begin{cases} 1, & \Delta p_{\text{ver}} \in [0, h_{\text{bet}}), \\ \frac{1}{2} \times \left[1 + \cos\left(\pi \times \frac{\Delta p_{\text{ver}} - h_{\text{bet}}}{1 - h_{\text{bet}}}\right)\right], & \Delta p_{\text{ver}} \in [h_{\text{bet}}, 1], \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

where r_{bet} and h_{bet} are real numbers between 0 and 1. The second term in Eq. (13),

$$\mathbf{f}_{\mathbf{i}}^{\mathbf{d}} = a_2 \sum_{j \in N_i} (b_{i,j}(\mathbf{q_j}, \mathbf{q_i}) \times (\mathbf{p_j} - \mathbf{p_i})) \tag{19}$$

is a velocity consensus term to keep agent i flying in the same direction as its neighboring agent(s) j. This term enables multiple hunter agents to hunt the single moving target agent. The heterogeneous adjacency term

$$b_{i,j}(\mathbf{q_j}, \mathbf{q_i}) = \rho\left(\frac{\|\mathbf{q_j} - \mathbf{q_i}\|_\sigma}{d_\beta}\right), \tag{20}$$

where $\|\mathbf{z}\|_\sigma$ is saturated distance norm

$$= \frac{\sqrt{1 + \epsilon \times \|\mathbf{z}\|^2} - 1}{\epsilon} \tag{21}$$

and same bump function $\rho(\boldsymbol{\Delta}\mathbf{p})$ to support a smooth repulsion force variation. $\mathbf{d}_\beta$ is the distance at which both the gradient-based term $\mathbf{f}_{\mathbf{i}}^{\mathbf{g}}$ and the velocity consensus term $\mathbf{f}_{\mathbf{i}}^{\mathbf{d}}$ become zero.

4. Simultaneous Encirclement Strategy

To decrease the escaping rate of the moving target agent, hunter agents should encircle the target agent simultaneously and evenly in the three-dimensional hunting environment. Hence, this section consists of two components, simultaneous encirclement and goal location determination.

A simultaneous encirclement strategy is added for hunter agents to perform when they are close to the target. However, as it is also essential to keep the computational efficiency as simple as possible for the hunter agents so they would not need to spend

Algorithm 1. Simultaneous Encirclement

```
1: Monitor positions of hunter agents and targets.
2: If not all hunter agents are within range of $a$ from target:
   the hunter agents that are within range of $a$ stay where they are while waiting
   for the others to plan trajectories and move toward the target
   Else:
   All hunter agents plan their trajectories and move toward the target to achieve
   an encirclement of the target
```

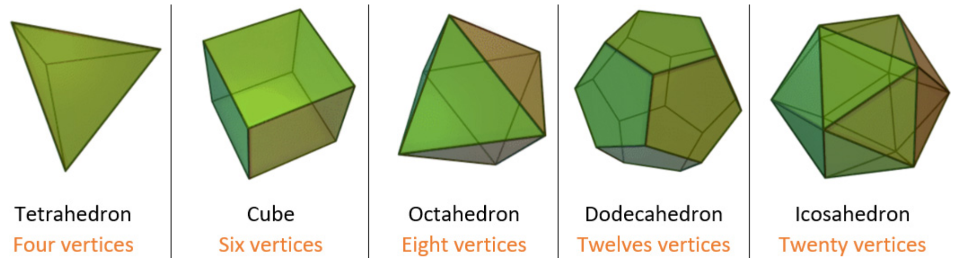


Fig. 11. Platonic solid in three-dimensional space.

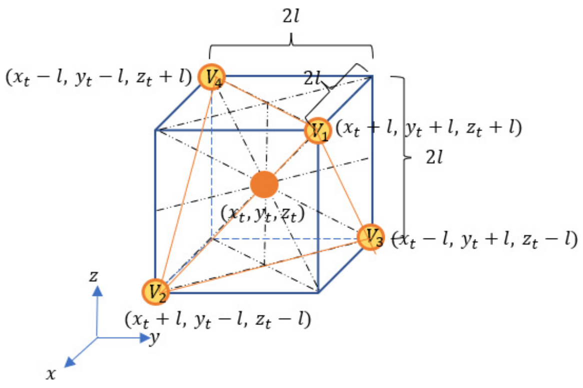

$2l$
$(x_t - l,\ y_t - l,\ z_t + l)$
V_4
$2l$
V_1
$(x_t + l,\ y_t + l,\ z_t + l)$
$2l$
(x_t, y_t, z_t)
V_3
$(x_t - l,\ y_t + l,\ z_t - l)$
V_2
$(x_t + l,\ y_t - l,\ z_t - l)$
z
y
x

Fig. 12. Vertices selection for four hunter agents in three-dimensional.

too much time calculating their next location while hunting a target is flying, a simple simultaneous strategy is designed with a pseudo-code as Algorithm 1.

In addition to simultaneous encirclement, hunter agents should have their goal locations determined so they can plan their trajectories to encircle the target agent. As mentioned before, hunter agents be of evenly allocated around the target is important. The idea of platonic solids is adopted to determine the vertices for hunter agents so that the agents evenly form a platonic solid around the target. Figure 11 provides some examples of this idea in a three-dimensional environment.[a] The tetrahedron centered at the target agent (x_t, y_t, z_t) is picked for a 4-hunter agent case. As downwash effects exist in a three-dimensional hunting environment, if the bottom UAV is hovering right underneath the top UAV, the bottom one experiences a downward push from the top and this might lead the bottom UAV, which never reaches its final goal location so a failure of encirclement. Hence, the Cartesian coordinates of the vertices in inertia space are determined by taking this phenomenon into consideration. Figure 12 provides an illustration of the selected vertices marked with cartesian coordinates (labeled by circles in yellow) and the center of the

[a] https://math.ucr.edu/home/baez/platonic.html.

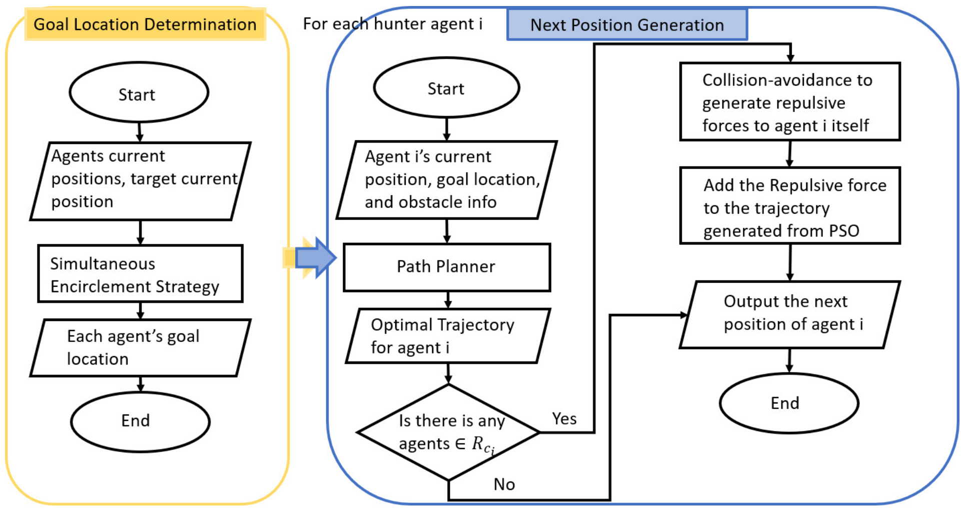

Goal Location Determination
Start
Agents current positions, target current position
Simultaneous Encirclement Strategy
Each agent's goal location
End
For each hunter agent i
Next Position Generation
Start
Agent i's current position, goal location, and obstacle info
Path Planner
Optimal Trajectory for agent i
Is there is any agents ∈ R_{c_i}
Yes
No
Collision-avoidance to generate repulsive forces to agent i itself
Add the Repulsive force to the trajectory generated from PSO
Output the next position of agent i
End

Fig. 13. Flow chart for the trajectory generation with collision-avoidance algorithm and simultaneous encirclement strategy.

tetrahedron (which is the location of the target agent and labeled by circles in orange).

With the vertices being determined, an algorithm should be designed as a method to assign these vertices to each hunter agent. When multiple agents' trajectories are crossed, the downwash effect might lead to the failure of hunting by pushing the bottom UAV downward, which might hit the ground if this bottom UAV is too close to the ground. Hence, the Hungarian algorithm is added and designed to assign vertices to hunter agents as goal locations by minimizing the trajectory crossing phenomenon. The cost matrix is described by Eq. (22). a_i stands for the unit position vector of agent i and v_j stands for the unit position vector of the vertices j. This cost matrix is to find the combination of agents and determined vertices so that the angle θ_{ij} between a_i and v_j is minimized.

$$\mathbf{C} = \begin{bmatrix} a_1 \cdot v_1 & a_1 \cdot v_2 & a_1 \cdot v_3 & a_1 \cdot v_4 \\ a_2 \cdot v_1 & a_2 \cdot v_2 & a_2 \cdot v_3 & a_2 \cdot v_4 \\ a_3 \cdot v_1 & a_3 \cdot v_2 & a_3 \cdot v_3 & a_3 \cdot v_4 \\ a_4 \cdot v_1 & a_4 \cdot v_2 & a_4 \cdot v_3 & a_4 \cdot v_4 \end{bmatrix}$$
$$= \begin{bmatrix} \cos(\theta_{11}) & \cos(\theta_{12}) & \cos(\theta_{13}) & \cos(\theta_{14}) \\ \cos(\theta_{21}) & \cos(\theta_{22}) & \cos(\theta_{23}) & \cos(\theta_{24}) \\ \cos(\theta_{31}) & \cos(\theta_{32}) & \cos(\theta_{33}) & \cos(\theta_{34}) \\ \cos(\theta_{41}) & \cos(\theta_{42}) & \cos(\theta_{43}) & \cos(\theta_{44}) \end{bmatrix}. \tag{22}$$

A flow chart in Fig. 13 summarizes the path planning algorithm with the collision-avoidance algorithm and simultaneous encirclement strategy.

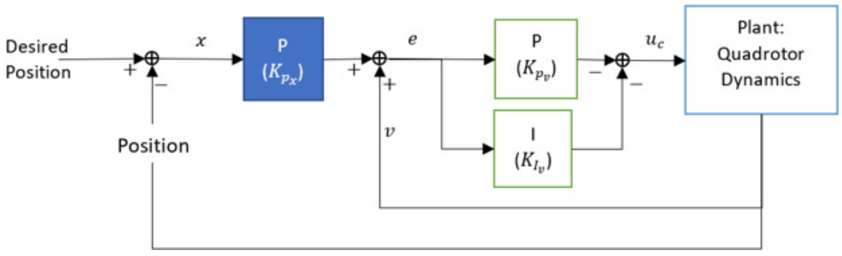


Fig. 14. Cascade PI controller.

5. Controller

A Cascade PI Controller shown in Fig. 14 is adopted here.

This cascade PI controller acts as an outer loop controller to yield desired angles and thrust by sending its outputs to the inner loop attitude control. The inputs for the position controller are the differences between hunter agents' positions and their desired positions differences ($\mathbf{x}$) and velocities ($\mathbf{v}$), and the controller outputs $\mathbf{u_c}$, consists of pitch angles (θ_c), roll angles (ϕ_c), and thrust force (Thrust).

$$\mathbf{e} = \mathbf{K_{p_x}}\mathbf{x} + \mathbf{v}, \tag{23}$$

$$\mathbf{u_c} = \begin{bmatrix} \theta_c \\ \phi_c \\ \text{Thrust} \end{bmatrix} = -\mathbf{K_{p_v}}\mathbf{e} - \mathbf{K_{I_v}}\int \mathbf{e}dt, \tag{24}$$

$$\mathbf{u_c} = -\mathbf{K_{p_x}}\mathbf{K_{I_v}}\int \mathbf{x}dt - (\mathbf{K_{p_x}}\mathbf{K_{p_v}} + \mathbf{K_{I_v}})\mathbf{x} - \mathbf{K_{p_v}}\mathbf{v}, \tag{25}$$

where gain matrices $\mathbf{K_{P_x}}$, $\mathbf{K_{P_v}}$, and $\mathbf{K_{I_v}}$ are tuned to gain good performance.

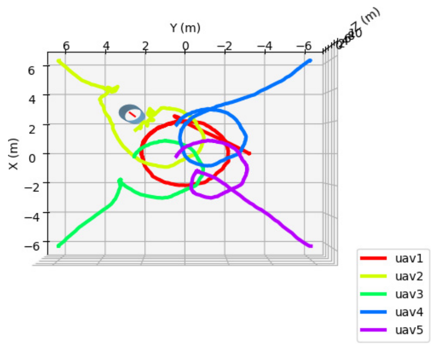
$$\mathbf{K_{p_x}} = \begin{bmatrix} 0.95 & 0 & 0 \\ 0 & 0.95 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}, \tag{26}$$

$$\mathbf{K_{p_v}} = \begin{bmatrix} 0.08 & 0 & 0 \\ 0 & 0.08 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}, \tag{27}$$

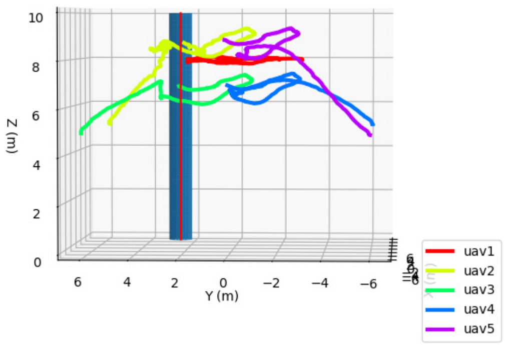
$$\mathbf{K_{I_x}} = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.02 \end{bmatrix}. \tag{28}$$

6. Simulation Results

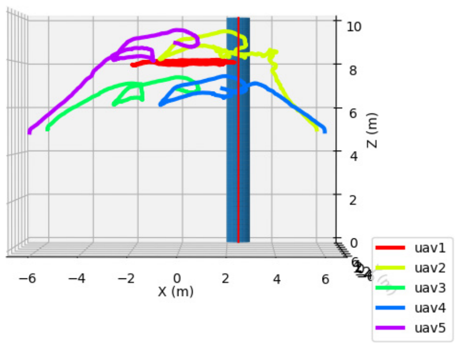
As mentioned in the previous section, the Gazebo simulator and pure python environment are chosen as the simulators. This section presents the simulation results of



(a) xy



(b) yz



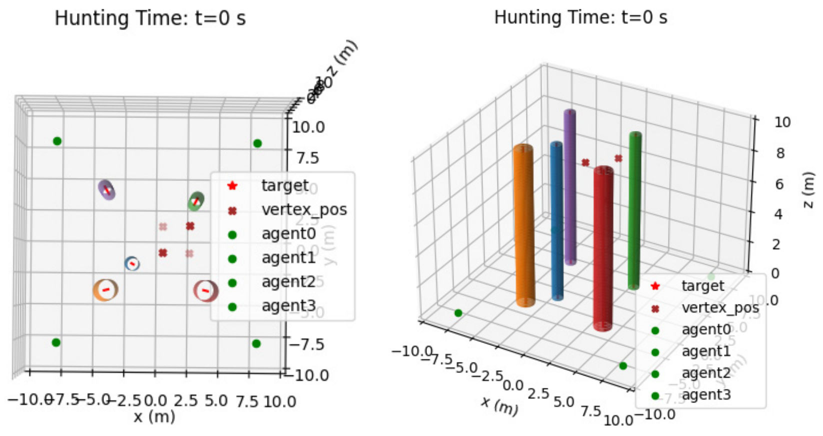
(c) xz

Fig. 15. Gazebo Environment, 4 agents hunting single moving agent in static environment.

two complicated cooperative hunting scenarios, multiple hunters and one moving target in a cluttered static/dynamic environment. It is worth mentioning that for the scenario with the target moving and the Gazebo simulator being selected to provide the simulation results, the target's moving speed is set to be around $0.1\,\mathrm{m/s}$. This speed is a pretty slow-moving speed for an escaping target drone. Still, it is chosen because of the limited computational capabilities of the computer being used and the high computational capability that the Gazebo simulator requires. For the other scenario where pure python is selected to provide a stimulating environment, the target can escape at a much higher speed.

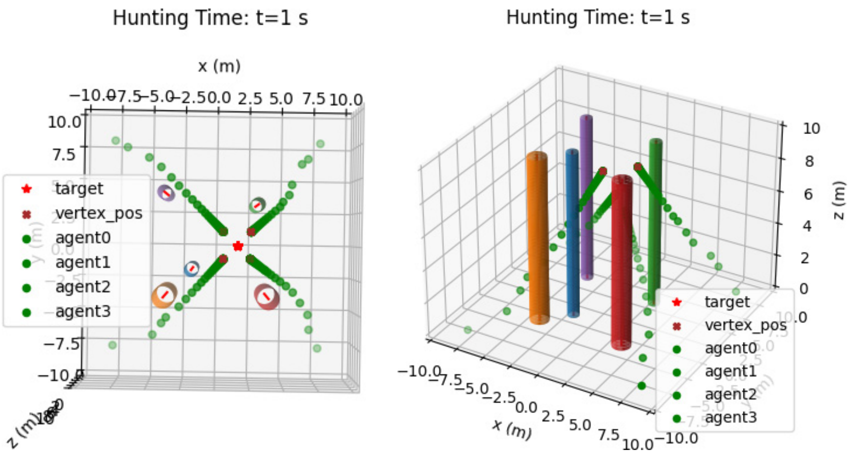
Scenario 1. Multi-agent hunting single moving agent in a static obstacle environment.

Target UAV (labeled by red) moves in the hunting environment in Fig. 15. This UAV moves in a circular path of radius $= 2\,\mathrm{m}$, at a constant speed of $v = 0.1\,\mathrm{m/s}$. Hunter UAVs' initial locations are (6, 6, 5) m, (−6, 6, 5) m, (−6, −6, 5) m, and (6, −6, 5) m. A static cylindrical obstacle is of radius $= 0.5\,\mathrm{m}$ and height $= 10\,\mathrm{m}$, at location $x = 2.5\,\mathrm{m}$, $y = 2.5\,\mathrm{m}$. The simulation results are shown in Fig. 15. It is collected from the Gazebo simulator.



(a) xy (b) xyz

Fig. 16. Time instant 1: $t = 0\,\mathrm{s}$.



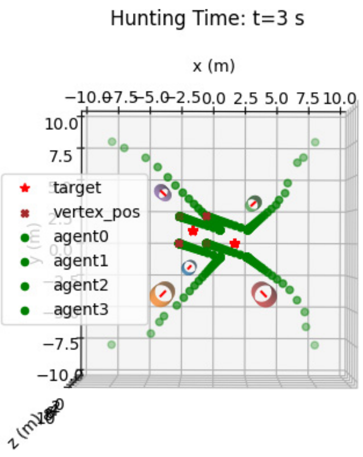
(a) xy (b) xyz

Fig. 17. Time instant 1: $t = 1\,\mathrm{s}$.

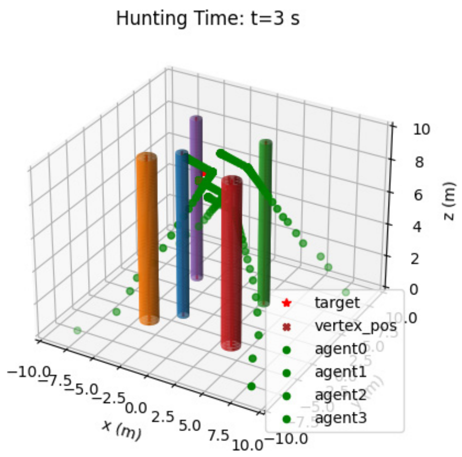
We observe that the hunter agents plan their collision-free trajectories from their initial positions separately to avoid the downwash impact and trace and encircle the moving target in the shape of a tetrahedron while preventing downwash impact successfully.

Scenario 2. Multi-agent hunting single moving agent in a dynamic environment.

The target agent moves waypoint by waypoint in this scenario. The target agent's initial location is (1.5, 0.0, 8.0) m, followed by (−1.5, 1.0, 8.0) m, (−1.5, −1.0, 8.0) m,

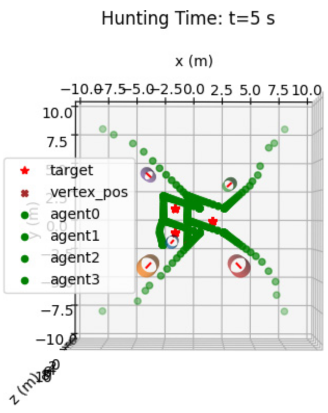


(a) xy

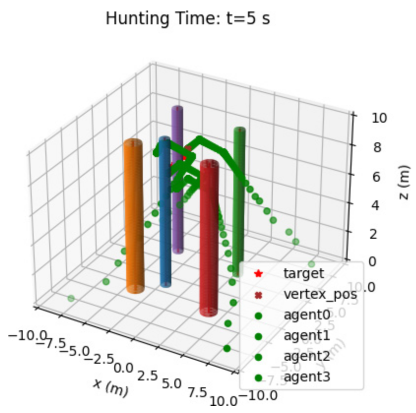


(b) xyz

Fig. 18. Time instant 1: $t = 3\,\mathrm{s}$.



(a) xy

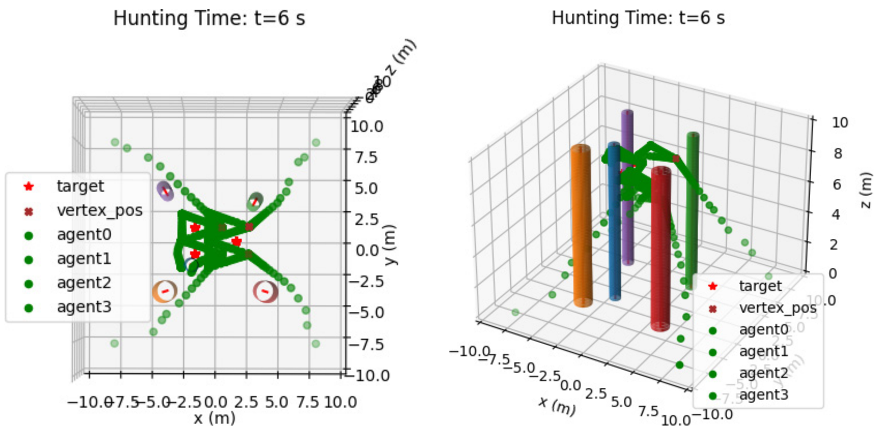


(b) xyz

Fig. 19. Time instant 1: $t = 5\,\mathrm{s}$.

and (1.5, 0.0, 8.0) m. Randomly located obstacles move at the same constant speed during the cooperative hunting procedure. The hunter agents' initial locations are (8.0, 8.0, 0.0) m, (−8.0, 8.0, 0.0) m, (8.0, −8.0, 0.0) m, and (−8.0, −8.0, 0.0) m. Python serves as the simulator for this cooperative hunting activity. The simulation at different time instants is plotted and included in Figs. 16–20.

With the existence of the downwash impact in this three-dimensional hunting space, all hunter agents are observed to succeed in planning their collision-free trajectories, encircling and tracing the moving target while avoiding collisions with the moving obstacles in the environment.



(a) xy (b) xyz

Fig. 20. Time instant 1: $t = 6\,\text{s}$.

7. Comparison Studies

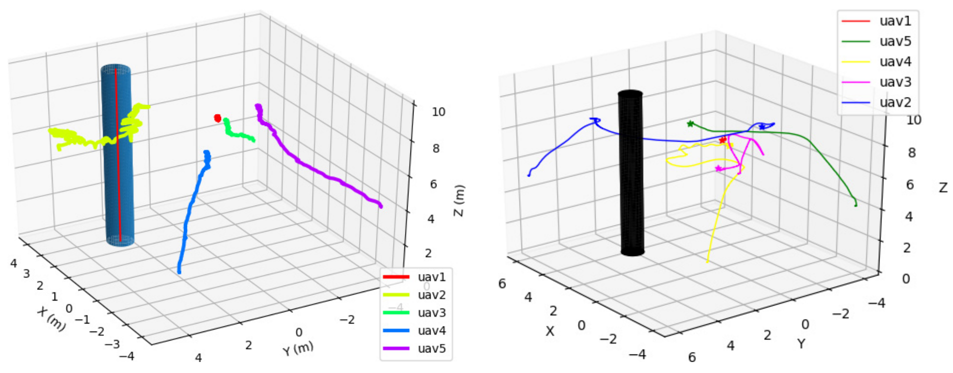
This section applies the proposed solution and the existing solution, the leader–follower controller, in three-dimensional cooperative hunting problems to compare these two solutions' performances. This section starts with a brief introduction to the leader–follower controller and follows with a comparative study between the proposed solution and this controller in a complicated cooperative hunting situation.

7.1. *Leader–follower controller in three-dimensional space*

An existing solution to the 3D cooperative hunting problem of multi-agent in a cluttered environment is proposed in Ge's paper.[21] This author solves this problem through dynamic control, a distributed sliding mode position controller, along with the PX4 PID attitude onboard controller. The solution consists of a distributed fixed-time observer, an obstacle avoidance algorithm, and a sliding mode position controller.

A distributed fixed-time observer enables communications of information among neighbors via topology networks. The design of this controller is by the rule that the leader hunter obtains the information, including position and velocity, of the target, while the other agent acting as the followers are not necessarily accessible to this information when hunting the target. Instead, the hunters use this fixed-time observer to estimate the information of the target.

To avoid a collision from the obstacles in the environment of cooperative hunting, the idea of an artificial potential field is adopted. Especially, a repulsive potential function is created based on the relative distance between the center of the hunter agent and the center of an obstacle. The repulsive force from the obstacle is obtained by taking the negative gradient of this potential function. Summing up, the repulsive



(a) Proposed Solution

(b) Distributed Sliding Mode Position Controller

Fig. 21. Comparison when starting locations are (4, 4, 5) m, (4, −4, 3) m, (−4, 4, 3) m, (−4, −4, 4) m, target hovering at (0, 0, 8) m, a static cylindrical obstacle of radius = 0.5 m and height = 10 m, at location $x = 2.5$ m, $y = 2.5$ m.

forces from all the obstacles make up the total repulsive force the agent should create to achieve collision-free in the cluttered environment.

A sliding mode position controller with the PX4 onboard PID attitude controller works as means for multiple hunter agents to trace and surround a single target. The desired trajectory generated from the above two components based on the current positions and attitudes of the hunter agents is sent to this sliding mode position controller to obtain the desired input u. u is converted to the desired attitude, including the roll angle, pitch angle, and thrust. After that, these desired attitudes are sent to the PX4 onboard PID attitude controller used to manipulate the quadrotor.

7.2. *Comparative study*

This section has four hunter agents who cooperatively hunt a target in a complicated scenario in a Gazebo simulation environment to compare the performance of the proposed solution with the leader–follower controller solution. In particular, the leader–follower controller solution is applied to cooperative hunting in a cluttered environment.

To make the simulations comparable, velocity limits ($V_{\max} = \pm 5.0\,\mathrm{m/s}$), acceleration limits ($a_{\max} = \pm 5.0\,\mathrm{m/s^2}$), and time interval ($dt = 0.2\,\mathrm{s}$) are set to be the same for both solutions.

In the scenario where an obstacle is laying on the hunter agents' way toward their goal locations, two groups can both path around the block and arrive at their final goal locations to encircle the target agent as shown in Fig. 21. Compared with the proposed solution that develops a trajectory of 53.94 m, the distributed sliding mode position controller's generated trajectory is 65.51 m. In other words, the proposed solution provides a shorter hunting route and faster convergence than the leader–follower controller solution for the hunter agents. Zigzagging is because of the combined repulsive force from the collision-avoidance algorithm and the attractive force from the PSO* path planner. This phenomenon leads to a future task — designing a solution to smooth the trajectories.

8. Conclusion

This paper serves as a continuity of Chen's research on the downwash impact in a three-dimensional UAV-dense environment and an extension on cooperative hunting of multi-UAV scenarios when considering downwash impact. As the downwash impact can push the UAV beneath downward, this impact can lead to a crash and hunting failure if the UAV is close to the ground. Hence, it is essential to take downwash impact into consideration while planning trajectories and determining goal locations for hunter UAVs. To achieve this goal, a solution that consists of a novel revised PSO path planning algorithm, a collision-avoidance controller, a simultaneous encirclement strategy, and a cascade PI controller, is proposed in this paper. The simulation results of this solution in the complicated cooperative hunting scenarios have been conducted. Results indicate successful hunting and encirclement of the escaping target when downwash impact exists. The proposed solution is then compared with the leader–follower controller, and the proposed solution has better cooperative hunting performance.

References

1. S. Samaras *et al.*, Deep Learning on Multi sensor Data for Counter UAV Applications — A Systematic Review, *Sensors* **19**(22), 4837 (2019).
2. K. G. Panda, S. Das, D. Sen and W. Arif, Design and Deployment of UAV-aided Post-disaster Emergency Network, *IEEE Access* **7**, 102985–102999 (102999).
3. J. A. Shaffer, E. Carrillo and H. Xu, Hierarchal Application of Receding Horizon Synthesis and Dynamic Allocation for UAVs Fighting Fires, *IEEE Access* **6**, 78868–78880 (78880).
4. E. A. Mitishita and N. L. Santos de Salles Graca, The Influence of Redundant Images in Uav Photogrammetry Application, in *IEEE International Geoscience and Remote Sensing Symposium* (2018), pp. 7894–7897, https://ieeexplore.ieee.org/document/8517423.
5. J. T. K. Ping *et al.*, Generic Unmanned Aerial Vehicle (UAV) for Civilian Application — A Feasibility Assessment and Market Survey on Civilian Application for Aerial Imaging, in *2012 IEEE Conference Sustainable Utilization and Development in Engineering and Technology (STUDENT)*, pp. 289–294, https://ieeexplore.ieee.org/document/6408421.
6. R. Olfati-Saber, Flocking for Multi-agent Dynamic Systems: Algorithms and Theory, *IEEE Trans. Automat. Contr.* **51**(3), 40120 (2006), doi: 10.1109/TAC.2005.864190.
7. L. Yu *et al.*, Hybrid Attention-oriented Experience Replay for Deep Reinforcement Learning and Its Application to a Multi-robot Cooperative Hunting Problem, *Neurocomputing* **523**, 44–57 (57), doi: 10.1016/j.neucom.2022.12.020.
8. B. A. Baba and B. Bilgehan, Optimisation of Multi Robots Hunting Game, in *AIP Conference Proceedings*, Vol. 2325, No. 1 (American Institute of Physics, 2021), https://aip.scitation.org/doi/10.1063/5.0040282.

9. O. Hamed and M. Hamlich, Improvised Multi-robot Cooperation Strategy for Hunting a Dynamic Target, in *2020 International Symposium of Advanced Electrical and Communication Technologies (ISAECT)* (IEEE, 2020), pp. 1–4, doi: 10.1109/ISAECT50560.2020.9523684.
10. R. Hu *et al.*, A new scheme for cooperative hunting tasks with multiple targets in dynamic environments, in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2021), pp. 1816–1822, doi: 10.1109/ROBIO54168.2021.9739257.
11. D. Xia, S. Guo, L. Shi, H. Xing, X. Hou, Z. Li and M. Zhou, Quadrotor location-based target hunting strategy for multiple amphibious spherical robots, in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)* (IEEE, 2020), pp. 1263–1268, doi: 10.1109/ICMA49215.2020.9233537.
12. L. Li *et al.*, A research of multiple autonomous underwater vehicles cooperative target hunting based on formation control, in *2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)* (IEEE, 2021), pp. 22–27, doi: 10.1109/CACRE52464.2021.9501365.
13. G. Wu *et al.*, Review of multiple unmanned surface vessels collaborative search and hunting based on swarm intelligence, *Int. J. Adv. Robot. Syst.* **19**(2), 172988062210918 (2022), doi: 10.1177/17298806221091885.
14. C. de Souza, P. C. Garcia and B. Vidolov, Local interaction and navigation guidance for hunters drones: A chase behavior approach with real-time tests, *Robotica* **40**(8), 2697–2715 (2022).
15. H. Jinqiang, W. Husheng, Z. Renjun, M. Rafik and Z. Xuanwu, Self-organized search-attack mission planning for UAV swarm based on wolf pack hunting behavior, *J. Syst. Eng. Electron.* **32**(6), 1463–1476 (2021), doi: 10.23919/JSEE.2021.000124.
16. Z. Zhang *et al.*, Multi-robot cooperative pursuit via potential field-enhanced reinforcement learning, in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022), pp. 8808–8814, doi: 10.1109/ICRA46639.2022.9812083.
17. C.-C. Chen and H. H.-T. Liu, Adaptive modelling for downwash effects in multi-UAV path planning, *Guidance, Navigation and Control* **01**(04), 2140005 (2021), doi: 10.1142/S2737480721400057.
18. L. Cai and Q. Sun, Multiautonomous underwater vehicle consistent collaborative hunting method based on generative adversarial network, *Int. J. Adv. Robot. Syst.* **17**(3), 172988142092523 (2020), doi: 10.1177/1729881420925233.
19. X. Cao and X. Xu, Hunting algorithm for multi-AUV based on dynamic prediction of target trajectory in 3D underwater environment, *IEEE Access* **8**, 138529–138538 (2020), doi: 10.1109/ACCESS.2020.3013032.
20. S. Fan, G. Song, C. C. Chen, P. Shi and H. H. Liu, "Multi-UAV cooperative hunting using PSO in 3D cluttered environment," in *Advances in Guidance, Navigation and Control. ICGNC 2022*, Lecture Notes in Electrical Engineering, Vol. 845 (Springer, Singapore, 2023), doi: 10.1007/978-981-19-6613-2_352.
21. G. Song and S. W. X. Peng, Distributed target-encirclement control of multiple quadrotors with a leader–follower framework, *ICIC Express Letters* **16**(9), 973–982 (2021).

Shichen Fan received her Bachelor's degree in Mechanical Engineering (Mechatronics option) and MA.Sc. degree in Aerospace Engineering from the University of British Columbia and the University of Toronto Institute for Aerospace Studies, in 2020 and 2022, respectively. Her research interests include multi-UAV path planning, downwash impact, and multi-UAV cooperative hunting. She is the corresponding author of this paper.

Hugh H-.T. Liu received his Bachelor's degree in automatic control from the Shanghai Jiao Tong University, Shanghai, China, in 1991, Master's degree in flight control and simulation from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1994, and Ph.D. in mechanical engineering from the University of Toronto, Canada, in 1998. He is currently Professor at the Institute for Aerospace Studies, University of Toronto, where he has been a Faculty Member since 2000. He is an internationally leading researcher in the area of aircraft systems and control. He is also currently the Principal Investigator of the Natural Science and Engineering Research Council of Canada-Collaborative Research and Training Experience Program on Unmanned Aerial Vehicles and Director of the Centre for Aerial Robotics Research and Education, Ottawa, ON. His research interests include aircraft systems and control, autonomous unmanned systems, as well as integrated modeling and simulations.