aerospace

MDPI

Article

An Adaptive Cylindrical-Pruning Decision Tree Method for Conflict Detection During Low-Altitude Multi-UAV Operations

Xijun Liu, Zelin Chen, Zhaoyang Li * and Dongcheng Luo

College of Aviation Electronic and Electrical Engineering, Civil Aviation Flight University of China (CAFUC), Chengdu 641450, China; liuxijun@cafuc.edu.cn (X.L.); czlcafuc@126.com (Z.C.); ldccafuc@126.com (D.L.)
* Correspondence: lzy2024@cafuc.edu.cn

Abstract

Dense low-altitude multi-UAV operations require conflict detection methods that can rapidly identify potential conflicts while preserving detection reliability under realistic motion constraints. This paper proposes an Adaptive Cylindrical-Pruning Decision Tree Method (AC-DTPM) for short-term conflict detection in dense low-altitude UAV traffic. A realistic UAV motion model is first constructed by considering speed limits, acceleration constraints, climb/descent limits, response delay, positioning error, and wind disturbance. Based on this model, a double-layer cylindrical protection zone is established, including a core conflict zone and an outer warning zone. The proposed AC-DTPM improves the basic Decision Tree Pruning Method through two mechanisms: adaptive time segmentation, which adjusts trajectory-segment length according to UAV speed and local traffic density, and cylindrical warning-zone lower-bound pruning, which eliminates impossible conflict candidates by separately evaluating horizontal and vertical distance lower bounds during tree search. Simulation experiments were conducted for 100–800 UAVs under locally dense and crossing-route scenarios. At 800 UAVs, AC-DTPM reduced the number of evaluated candidate pairs from 601.0 with the basic DTPM to 178.8, corresponding to a 70.3% reduction and a candidate compression ratio of 5.59×10^{-4}. Compared with the basic DTPM, AC-DTPM reduced total detection time by 32.5% and memory cost by approximately 17.0%, while maintaining a missed-detection rate of zero. These results indicate that AC-DTPM provides an interpretable and resource-efficient front-end conflict detection method for scalable low-altitude UAV traffic management.

Keywords: multi-UAV; conflict detection; decision tree pruning; adaptive time segmentation; cylindrical protection zone

Check for updates

Academic Editor: Gokhan Inalhan

Received: 9 June 2026
Revised: 6 July 2026
Accepted: 14 July 2026
Published: 18 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

With the rapid development of low-altitude airspace operations, unmanned aerial vehicles (UAVs) are increasingly being used in logistics delivery, emergency response, infrastructure inspection, urban patrol, and geographic surveying. These applications are evolving from isolated single-UAV missions to dense multi-UAV cooperative operations. In such scenarios, multiple UAVs may operate within the same constrained low-altitude airspace, leading to frequent trajectory interactions and increased conflict risks. Reliable conflict detection is therefore a fundamental component of low-altitude airspace management, because it directly affects trajectory replanning, collision avoidance, operational efficiency, and public safety.

The emergence of the low-altitude economy further increases the need for scalable and intelligent airspace management. Recent studies on intelligent low-altitude mobility have emphasized that future low-altitude systems should integrate perception, communication, trajectory prediction, and intelligent decision-making to support safe large-scale operations [1]. A comprehensive survey of conflict detection and resolution in unmanned aircraft system traffic management also indicated that safety, scalability, uncertainty, and computational efficiency remain central challenges for dense UAV operations [2]. Therefore, low-altitude multi-UAV conflict detection should not only identify potential separation violations, but also provide efficient candidate screening and reliable warning information under realistic motion constraints and uncertain operating conditions.

Existing conflict detection and resolution methods can be broadly divided into uncertainty-aware prediction methods, geometric or optimization-based methods, grid-based search methods, and learning-based methods. Probabilistic and uncertainty-aware methods consider positioning errors, prediction deviations, and control delays, and they provided useful tools for modeling uncertain UAV motion. For example, Chen et al. proposed an in-flight conflict-free trajectory replanning method that considered UAV position uncertainty and energy consumption [3]. Such methods improve the robustness of replanning under uncertainty, but their computational burden may increase rapidly when the number of UAVs and uncertainty sources grows. As a result, they are not always suitable as fast front-end candidate-screening methods for dense low-altitude traffic.

Grid-based and geometric search methods improve computational efficiency by discretizing the airspace or by reducing unnecessary pairwise comparisons. Four-dimensional grid-based approaches can represent spatial and temporal occupancy simultaneously, while decision-tree pruning methods organize predicted trajectories into hierarchical spatiotemporal structures to reduce exhaustive pairwise detection. Monteiro et al. proposed a Decision Tree Pruning Method (DTPM) for four-dimensional trajectory conflict detection and resolution in which trajectory segments were pruned according to bounding-box relationships [4]. Compared with exhaustive pairwise detection, DTPM can reduce the number of candidate comparisons. However, conventional DTPM usually adopts fixed time segmentation and mainly relies on axis-aligned bounding-box overlap. In dense low-altitude scenarios, fixed segmentation cannot fully adapt to different UAV speeds and local traffic densities, while conservative bounding boxes may retain many invalid candidate pairs.

Reinforcement learning has also become an important research direction for UAV conflict resolution and swarm decision-making. Graph-based and multi-agent reinforcement learning methods can learn complex interaction patterns and generate adaptive maneuvering strategies in dynamic environments. Li et al. developed a multiscale-graph-enhanced reinforcement learning method for conflict resolution in dense UAV networks [5]. Isufaj et al. applied graph convolutional reinforcement learning to multi-UAV conflict resolution [6], and Zhao et al. proposed a graph-based multi-agent reinforcement learning method for large-scale UAV swarm control [7]. These studies show the potential of reinforcement learning in cooperative decision-making and conflict resolution. Nevertheless, learning-based methods usually require large-scale training data, generalization validation, and additional online inference resources. More importantly, learned policies do not inherently provide deterministic geometric guarantees for complete conflict detection. For safety-critical low-altitude operations, interpretable and deterministic conflict detection methods are still needed as reliable front-end modules before resolution, replanning, or reinforcement-learning-based decision-making [8,9].

In addition to computational efficiency, the safety model itself should reflect the operational characteristics of low-altitude UAVs [10,11]. Multirotor UAVs usually have stronger horizontal maneuvering capability than vertical maneuvering capability, and

their short-term prediction is affected by speed limits, acceleration constraints, positioning errors, response delays, and wind disturbances [12,13]. Therefore, a cylindrical protection zone model that separately describes horizontal and vertical safety separations is more appropriate than a simple spherical or point-mass model for dense low-altitude UAV operations [14–16]. However, how to combine this anisotropic safety model with efficient candidate screening remains insufficiently addressed in existing DTPM-based detection frameworks [17].

To address this problem, this paper proposes an Adaptive Cylindrical-Pruning Decision Tree Method (AC-DTPM). The method constructs a double-layer cylindrical protection-zone model and introduces two improvements based on DTPM. First, an adaptive time-segmentation mechanism adjusts trajectory segment length according to UAV speed and local airspace density, improving trajectory representation in high-risk regions while reducing redundant segmentation in low-risk regions. Second, a cylindrical warning zone lower-bound pruning mechanism separately evaluates horizontal and vertical distance lower bounds between decision tree nodes. If the safety separation condition is already satisfied in either direction, the corresponding node pair can be pruned before final closest-point-of-approach confirmation.

To provide an overview of the research scope and methodological structure, the conceptual framework of this study is presented in Figure 1.

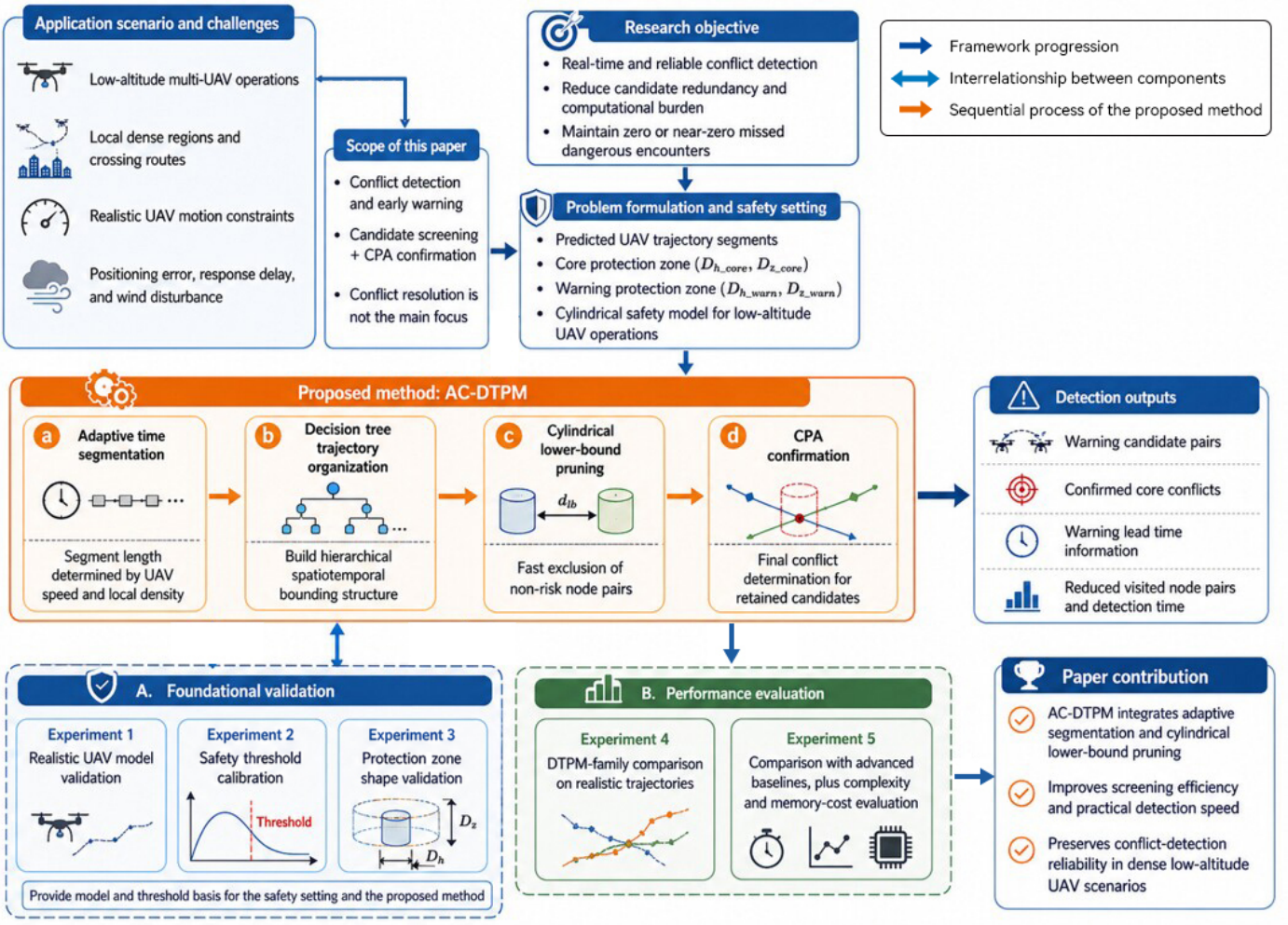


Figure 1. Conceptual framework.

The main contributions of this paper are summarized as follows.

1. A double-layer cylindrical protection zone model is integrated with airspace grid occupancy to support low-altitude UAV safety assessment and candidate screening.
2. An adaptive time-segmentation strategy is developed to improve four-dimensional trajectory representation under different UAV speed and local density conditions.
3. A cylindrical warning zone lower-bound pruning criterion is proposed to reduce invalid candidate pairs by exploiting independent horizontal and vertical safety constraints.
4. Simulation experiments using realistic UAV motion constraints and different UAV scales, parameter settings, and advanced baselines are conducted to evaluate the detection completeness, candidate screening efficiency, real-time performance, and memory cost characteristics of AC-DTPM.

2. Double-Layer Cylindrical Protection Zone Model

UAV conflict determination should not be based solely on ideal point-mass positions, but should also consider the physical size of the aircraft, positioning errors, control response delays, and redundant safety margins. For low-altitude multirotor UAVs, the horizontal maneuvering capability is generally stronger than the vertical maneuvering capability. Therefore, a cylindrical protection zone is more suitable than a spherical protection zone for separately characterizing the horizontal safety distance and the vertical safety height [18]. In this paper, an inner core zone and an outer warning zone are established. The inner core zone is used for strict intrusion determination, while the outer warning zone is used for potential conflict warning and candidate screening [19].

Let the center position of the i-th UAV at time t be given by Equation (1):

$$p_i(t) = [x_i(t), y_i(t), z_i(t)]^T \tag{1}$$

The inner core protection zone is defined by Equation (2):

$$\mathcal{C}_i^{core}(t) = \{q \mid d_h(q, p_i(t)) \le D_h^{core},\ |q_z - z_i(t)| \le D_z^{core}\} \tag{2}$$

The outer warning protection zone can be defined by Equation (3):

$$\mathcal{C}_i^{warn}(t) = \{q \mid d_h(q, p_i(t)) \le D_h^{warn},\ |q_z - z_i(t)| \le D_z^{warn}\} \tag{3}$$

where D_h^{core} and D_z^{core} are the horizontal radius and vertical height of the inner core zone, respectively, while D_h^{warn} and D_z^{warn} are those of the outer warning zone. These parameters satisfy

$$D_h^{\text{warn}} > D_h^{\text{core}},\ D_z^{\text{warn}} > D_z^{\text{core}} \tag{4}$$

Based on the above definitions, the outer zone and the inner zone form a hierarchical safety boundary from loose to strict constraints [20]. During the detection process, if another aircraft only enters the outer warning zone, it indicates that an approaching trend exists, and focused monitoring or further refined determination should be initiated [21]. If the aircraft further intrudes into the inner core zone, it indicates that the safety separation has failed, and the conflict confirmation and resolution stage should be immediately activated.

For an arbitrary grid cell $G_{u,v,w}$, if its center point $\mathbf{q}_{u,v,w}$ falls within the inner core zone of the i-th UAV, the cell is marked as a core occupied cell. If its center point is outside the inner core zone but inside the outer warning zone, the cell is marked as a warning occupied

cell. If neither condition is satisfied, the cell is marked as a free cell [22]. Accordingly, the occupancy state function can be written as

$$\chi_{u,v,w}^{(i)} = \begin{cases} 2, & \mathbf{q}_{u,v,w} \in \mathcal{C}_i^{\text{in}} \\ 1, & \mathbf{q}_{u,v,w} \in \mathcal{C}_i^{\text{out}} \setminus \mathcal{C}_i^{\text{in}} \\ 0, & \mathbf{q}_{u,v,w} \notin \mathcal{C}_i^{\text{out}} \end{cases} \tag{5}$$

When multiple UAVs are present in the airspace, the occupancy states of all aircraft with respect to the same grid cell can be integrated to obtain a global occupancy representation, namely

$$\chi_{u,v,w} = \max_{i=1,2,\ldots,N_{\text{UAV}}} \chi_{u,v,w}^{(i)} \tag{6}$$

where N_{UAV} denotes the number of UAVs in the airspace. The protection-zone modeling considering the difference between horizontal and vertical response capabilities is shown in Figure 2.

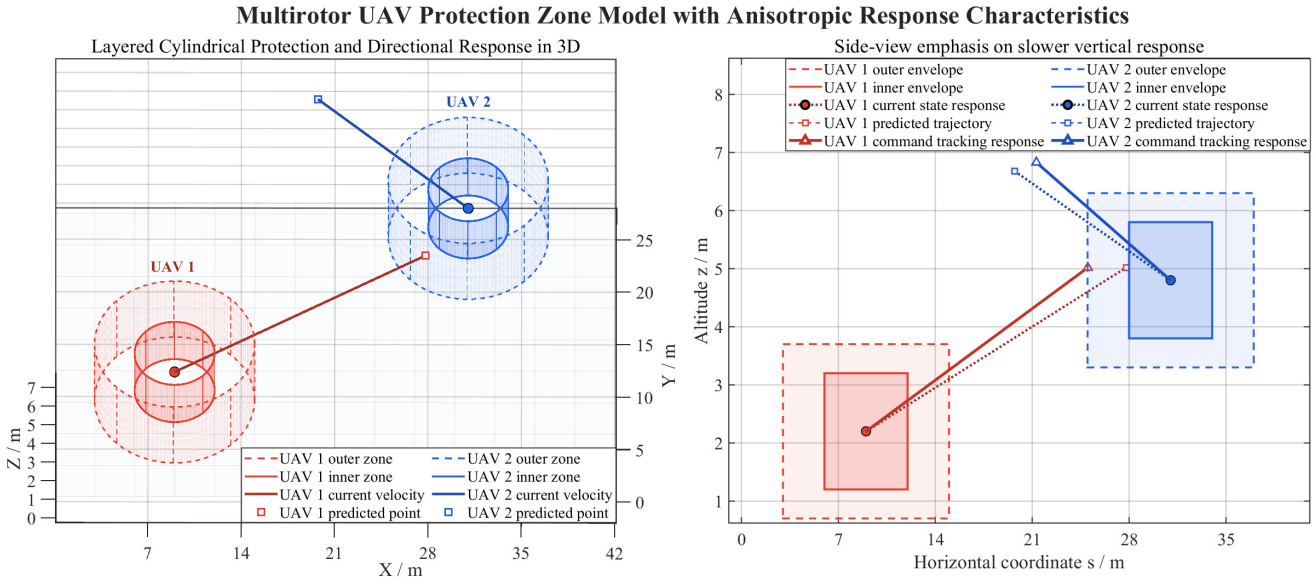


Figure 2. Double-layer cylindrical protection zone.

3. DTPM-Based Conflict Search Method

3.1. Four-Dimensional Trajectory Segment Representation

Let the three-dimensional center position vector and velocity vector of the i-th UAV generated by the realistic UAV motion model at time t be denoted as

$$\mathbf{p}_i(t) = [x_i(t), y_i(t), z_i(t)]^T \tag{7}$$

$$\mathbf{v}_i(t) = [v_{xi}(t), v_{yi}(t), v_{zi}(t)]^T \tag{8}$$

Within the short-term prediction window, the UAV trajectory used in this study is generated by a realistic UAV motion model. Let $\dot{\mathbf{x}}_i(t)$ denote the dynamic state of the i-th UAV. The realistic UAV motion model can be written in a general form as

$$\dot{\mathbf{x}}_i(t) = f_{\text{UAV}}(\mathbf{x}_i(t), \mathbf{u}_i(t-\tau_i), \mathbf{w}_i(t)),\ \mathbf{p}_i(t) = \mathbf{H}_p \mathbf{x}_i(t) + \mathbf{e}_i(t) \tag{9}$$

where f_{UAV} denotes the realistic UAV motion model, $\mathbf{u}_i(t-\tau_i)$ is the delayed control input, τ_i is the response delay, $\mathbf{w}_i(t)$ denotes wind disturbance, $\mathbf{e}_i(t)$ denotes positioning error,

and $\mathbf{H}_p$ extracts the position component from the UAV state. Therefore, $\mathbf{p}_i(t)$ represents the realistic simulated UAV center position used for subsequent trajectory segmentation.

To facilitate four-dimensional conflict search, each realistic UAV trajectory sample and its corresponding time are represented as a four-dimensional trajectory point:

$$\mathbf{q}_i(t) = [x_i(t), y_i(t), z_i(t), t]^T \tag{10}$$

Then, the four-dimensional trajectory of the i-th UAV within the prediction time window can be expressed as

$$\Gamma_i = \{\mathbf{q}_i(t) \mid t \in [t_0, t_0 + T_h]\} \tag{11}$$

To transform the continuous four-dimensional trajectory into a finite set of trajectory segments suitable for decision-tree processing, the prediction time window is divided into M time intervals:

$$t_k = t_0 + k\Delta t,\ k = 0, 1, \ldots, M \tag{12}$$

where

$$M = \left\lceil \frac{T_h}{\Delta t} \right\rceil \tag{13}$$

A four-dimensional trajectory segment can then be constructed using two adjacent four-dimensional trajectory points:

$$S_i^k = (\mathbf{q}_i(t_k), \mathbf{q}_i(t_{k+1})),\ k = 0, 1, \ldots, M-1 \tag{14}$$

Accordingly, the trajectory segment set of the i-th UAV within the prediction time window can be defined as

$$\mathcal{S}_i = \left\{ S_i^k \mid k = 0, 1, \ldots, M-1 \right\} \tag{15}$$

The complete set of four-dimensional trajectory segments for all UAVs is then expressed as

$$\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i \tag{16}$$

3.2. Construction of Trajectory Segment Bounding Boxes

To rapidly determine whether potential overlap exists between different realistic UAV trajectory segments, an axis-aligned bounding box is constructed for each four-dimensional piecewise linear trajectory segment. For the trajectory segment S_i^k, let its starting and ending four-dimensional trajectory points be $\mathbf{q}_i(t_k)$ and $\mathbf{q}_i(t_{k+1})$, respectively. The lower and upper bounds of this trajectory segment in the four dimensions (x), (y), (z), and (t) are defined as

$$\begin{gathered} x_{i,k}^{\min} = \min\{x_i(t_k), x_i(t_{k+1})\},\ x_{i,k}^{\max} = \max\{x_i(t_k), x_i(t_{k+1})\} \\ y_{i,k}^{\min} = \min\{y_i(t_k), y_i(t_{k+1})\},\ y_{i,k}^{\max} = \max\{y_i(t_k), y_i(t_{k+1})\} \\ z_{i,k}^{\min} = \min\{z_i(t_k), z_i(t_{k+1})\},\ z_{i,k}^{\max} = \max\{z_i(t_k), z_i(t_{k+1})\} \\ t_{i,k}^{\min} = t_k,\ t_{i,k}^{\max} = t_{k+1} \end{gathered} \tag{17}$$

For preliminary conservative candidate screening, the original bounding box is expanded using the warning zone safety margin. Let $D_{xy} = D_h^{warn}$ and $D_z = D_z^{warn}$. Then, the safety-expanded four-dimensional bounding box of the trajectory segment is defined as

$$\begin{aligned}\mathbf{b}_{i,k}^{-} &= [x_{i,k}^{\min} - D_{xy}, y_{i,k}^{\min} - D_{xy}, z_{i,k}^{\min} - D_z, t_{i,k}^{\min} - \Delta t_s]^T \\ \mathbf{b}_{i,k}^{+} &= [x_{i,k}^{\max} + D_{xy}, y_{i,k}^{\max} + D_{xy}, z_{i,k}^{\max} + D_z, t_{i,k}^{\max} + \Delta t_s]^T\end{aligned} \tag{18}$$

Thus, the trajectory segment bounding box can be abbreviated as

$$B_i^k = [\mathbf{b}_{i,k}^{-}, \mathbf{b}_{i,k}^{+}] \tag{19}$$

For two trajectory segments S_i^k and S_j^l, if no overlap exists between their four-dimensional bounding boxes, the two segments cannot constitute a conflict within the corresponding time range and can therefore be directly eliminated. The overlap criterion for four-dimensional bounding boxes can be written as

$$B_i^k \cap B_j^l \neq \varnothing \tag{20}$$

When the overlap condition is not satisfied in any one dimension, the trajectory segment pair can be regarded as having no potential conflict:

$$b_{i,k,d}^{-} > b_{j,l,d}^{+} \text{ or } b_{j,l,d}^{-} > b_{i,k,d}^{+},\ d = 1, 2, 3, 4 \tag{21}$$

3.3. Decision Tree-Based Candidate Search Framework

Let the final set of candidate trajectory segment pairs be denoted as $\mathcal{C}_p$. Then, the candidate UAV pair set can be expressed as

$$\mathcal{C}_p = \left\{ (i, j) \mid \exists k, l,\ (S_i^k, S_j^l) \in \mathcal{C}_s,\ i < j \right\} \tag{22}$$

Compared with exhaustive detection, the basic DTPM method does not directly perform CPA calculations for all UAV pairs. Instead, it first generates a smaller candidate UAV-pair set through four-dimensional trajectory segment bounding boxes and decision tree pruning [23]. The candidate compression ratio is defined as

$$\eta_c = \frac{|\mathcal{C}_p|}{N(N-1)/2} \tag{23}$$

where a smaller η_c indicates a more significant reduction in candidate pairs.

4. Improved AC-DTPM Method

The basic DTPM method usually segments the predicted trajectories using a fixed time step. This approach is simple to implement, but it does not account for differences in UAV velocities or variations in local traffic density. In large-scale low-altitude multi-UAV scenarios, different UAVs may have different velocities, relative approaching states, and numbers of neighboring aircraft. If all UAVs adopt the same time step, two types of problems may arise. First, in high-speed or dense regions, the trajectory segments may span excessively large intervals, resulting in overly conservative four-dimensional bounding boxes and an increased number of candidate UAV pairs. Second, in low-speed or sparse regions, the trajectories may be divided into excessively fine segments, producing redundant trajectory segments and increasing the burden of decision-tree construction and node search. Therefore, an adaptive time-segmentation mechanism is introduced into the basic DTPM framework, allowing the trajectory segment length to be dynamically adjusted according to UAV velocity and local density.

When there are N UAVs in the airspace, the total number of trajectory segments to be processed by the basic DTPM method is

$$N_s = NM \tag{24}$$

It can be seen that, when Δt is small, the number of trajectory segments increases rapidly, and the number of decision tree nodes and the search overhead increase accordingly. When Δt is large, the spatial span of a single trajectory segment becomes longer, and the coverage range of its bounding box expands. As a result, many trajectory segments that would originally be impossible to conflict are still retained as candidate objects. For the i-th UAV, the spatial length of a single trajectory segment can be expressed as

$$l_i = \|\mathbf{v}_i\|\Delta t \tag{25}$$

where $\|\mathbf{v}_i\|$ denotes the speed of the UAV. It can be seen from the above equation that, under the same time step, a high-speed UAV corresponds to a longer trajectory segment, whose four-dimensional bounding box is more likely to overlap with the bounding boxes of other trajectory segments, thereby increasing the number of invalid candidate UAV pairs. Therefore, a fixed time step can hardly balance the detection accuracy in high-speed regions and the computational efficiency in low-speed regions.

4.1. Adaptive Time-Segmentation Mechanism

Adjusting the time step solely according to velocity still cannot reflect the degree of local traffic congestion. In dense low-altitude operation scenarios, even if the UAV velocity is not high, finer time segmentation is still required to improve the resolution of conflict search when a large number of neighboring targets exist around the UAV. Therefore, a local density factor is further introduced in this paper.

Taking the current position of the i-th UAV as the center, a local statistical neighborhood is constructed. If the i-th UAV satisfies

$$d_{xy,ij} \le R_\rho,\ |z_i - z_j| \le H_\rho \tag{26}$$

then it is regarded as a local neighbor of the i-th UAV.

$$d_{xy,ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{27}$$

where R_ρ and H_ρ denote the horizontal radius and vertical height threshold for local density statistics, respectively. The number of neighbors of the i-th UAV is given by

$$n_i = \sum_{j=1, j\neq i}^{N} I(d_{xy,ij} \le R_\rho,\ |z_i - z_j| \le H_\rho) \tag{28}$$

where $I(\cdot)$ is the indicator function. To facilitate comparison across scenarios with different scales, the number of local neighbors is normalized as

$$\rho_i = \frac{n_i}{N-1} \tag{29}$$

where $\rho_i \in [0,1]$ indicates that the i-th UAV is located in a denser local airspace, and its predicted trajectory is more likely to form candidate conflicts with surrounding targets. By jointly considering velocity and local density, the adaptive time step is defined as

$$\Delta t_i = \mathrm{clip}\left(\frac{\eta D_s}{v_i + \kappa \rho_i v_{\max} + \varepsilon}, \Delta t_{\min}, \Delta t_{\max}\right) \tag{30}$$

where $v_{\max}$ denotes the maximum UAV speed, and κ denotes the local density weighting coefficient. In this expression, $\kappa \rho_i v_{\max}$ is used to transform the local density into an equivalent velocity penalty, so that a smaller time step can be assigned to high-density regions. Compared with the fixed time step method, the proposed method can improve trajectory segmentation accuracy in high-risk regions while reducing redundant trajectory segments in low-risk regions.

Adaptive Trajectory Segment Generation Method

According to the adaptive time step Δt_i, the prediction time nodes of the i-th UAV can be recursively generated as

$$t_{i,0} = t_0,\ i = 1, 2, \ldots, N \tag{31}$$

$$t_{i,m+1} = \min(t_{i,m} + \Delta t_i,\ t_0 + T_h) \tag{32}$$

The corresponding number of trajectory segments is

$$M_i = \left\lceil \frac{T_h}{\Delta t_i} \right\rceil \tag{33}$$

The four-dimensional trajectory segment of the i-th UAV can be expressed as

$$S_i^m = (\mathbf{q}_i(t_{i,m}), \mathbf{q}_i(t_{i,m+1})),\ m = 0, 1, \ldots, M_i - 1 \tag{34}$$

Therefore, after adaptive time segmentation is adopted, the total number of trajectory segments for all UAVs becomes

$$N_s^A = \sum_{i=1}^{N} M_i \tag{35}$$

Compared with the fixed segmentation method, adaptive time segmentation no longer requires all UAVs to have the same number of trajectory segments. Instead, prediction trajectory segments with different lengths are generated according to their respective velocities and local densities. Since the time intervals of all trajectory segments still continuously cover the prediction time window, no prediction time gap is introduced, and the completeness of the DTPM conflict search is not affected.

The improved trajectory-segment set is given by

$$\mathcal{S}^A = \bigcup_{i=1}^{N} \{S_i^m \mid m = 0, 1, \ldots, M_i - 1\} \tag{36}$$

Subsequently, $\mathcal{S}^A$ is input into the decision tree for bounding-box construction and pruning search.

Figure 3a illustrates the time-segmentation strategy of the proposed ATS-DTPM method. In low-risk regions, where the number of UAVs is small and their velocities are low, a larger time step is adopted to reduce the number of trajectory segments and computational redundancy. In high-risk regions, where the number of UAVs is large and their velocities are high, a smaller time step is adopted to improve trajectory segmentation accuracy and avoid generating a large number of invalid candidates due to overly conservative bounding boxes. Figure 3b shows the fixed time step strategy of the conventional DTPM method, in which the same time step is used regardless of variations in UAV velocity and local density.

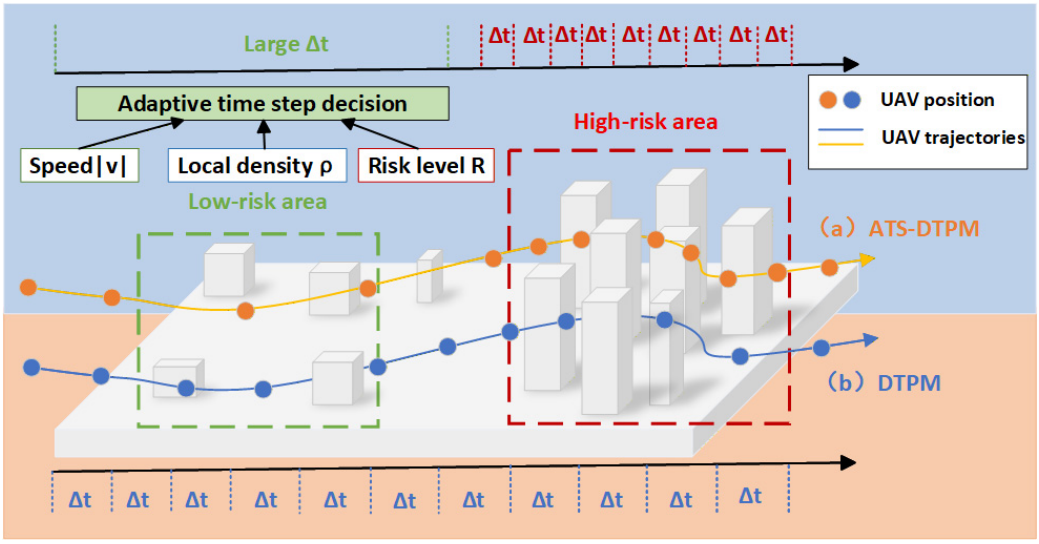


Figure 3. Adaptive time segmentation.

4.2. Cylindrical Warning Zone Lower-Bound Pruning Mechanism

The conventional DTPM method mainly performs pruning according to the interval overlap relationship of axis-aligned four-dimensional bounding boxes. Although this strategy can eliminate some trajectory segment pairs that are separated in time or space, it may still retain many pseudo-candidate node pairs whose bounding boxes intersect but whose trajectories cannot enter the cylindrical warning zone. In dense low-altitude multi-UAV scenarios, such pseudo-candidate pairs increase the number of recursive node searches and subsequent CPA confirmations.

To reduce this redundancy while preserving warning-level candidate relations, this paper introduces a cylindrical warning zone lower-bound pruning mechanism at the trajectory tree node level. The lower-bound pruning criterion is defined with respect to the outer warning zone, because the warning zone is used as the conservative candidate preservation boundary. A node pair is pruned only when it is guaranteed to remain outside the warning cylinder in at least one safety dimension. Specifically, the lower bounds of the horizontal and vertical distances between two tree nodes are calculated separately. If the horizontal lower bound is greater than the warning horizontal threshold D_{h}^{warn}, or the vertical lower bound is greater than the warning vertical threshold D_{z}^{warn}, then no trajectory segment combination contained in the two nodes can form either a warning conflict or a core conflict. Therefore, the node pair can be safely pruned before leaf node search and continuous CPA confirmation.

The core-zone thresholds are not used as hard pruning thresholds in the tree search stage. Instead, they are used in the final continuous CPA confirmation stage to distinguish confirmed core conflict pairs from warning-only pairs. This design ensures that warning-only candidates are not prematurely removed during lower-bound pruning.

4.2.1. Calculation of the Horizontal Distance Lower Bound

Let A and B be two trajectory tree nodes. Their spatial projection intervals in the x- and y-directions are defined as

$$\begin{aligned} I_x(A) &= [x_A^-, x_A^+],\ I_y(A) = [y_A^-, y_A^+], \\ I_x(B) &= [x_B^-, x_B^+],\ I_y(B) = [y_B^-, y_B^+]. \end{aligned} \tag{37}$$

The minimum interval separation distance between the two nodes in the x-direction is defined as

$$\Delta_x(A,B) = \max\left(0,\ x_A^- - x_B^+,\ x_B^- - x_A^+\right) \tag{38}$$

Similarly, the minimum interval separation distance between the two nodes in the y-direction is given by

$$\Delta_y(A,B) = \max\left(0,\ y_A^- - y_B^+,\ y_B^- - y_A^+\right) \tag{39}$$

When the projection intervals overlap in a certain direction, the corresponding interval separation is zero. When the two intervals are completely separated, the interval separation is equal to the shortest distance between their endpoints. Therefore, the lower bound of the horizontal distance between nodes A and B can be obtained as

$$d_h^{LB}(A,B) = \sqrt{\Delta_x^2(A,B) + \Delta_y^2(A,B)} \tag{40}$$

For any trajectory point p contained in node A and any trajectory point q contained in node B, their horizontal distance satisfies

$$\|p_{xy} - q_{xy}\|_2 \geq d_h^{LB}(A,B) \tag{41}$$

Therefore, if

$$d_h^{LB}(A,B) > D_h^{warn} \tag{42}$$

then all trajectory segment combinations within the two nodes are guaranteed to remain outside the horizontal boundary of the outer warning cylinder. Since the core cylinder is contained within the warning cylinder, the node pair cannot generate either a warning conflict or a core conflict and can be directly pruned.

4.2.2. Calculation of the Vertical Distance Lower Bound

The cylindrical protection zone also requires vertical separation evaluation. Let the height intervals of nodes A and B be defined as

$$I_z(A) = [z_A^-, z_A^+],\ I_z(B) = [z_B^-, z_B^+] \tag{43}$$

The minimum vertical separation distance between the two nodes is given by

$$d_z^{LB}(A,B) = \max\left(0,\ z_A^- - z_B^+,\ z_B^- - z_A^+\right) \tag{44}$$

If the height intervals of the two nodes overlap, the vertical lower bound is zero. If the height intervals are completely separated, the vertical lower bound is the shortest distance between the two height intervals. For any trajectory point p in node A and any trajectory point q in node B, the vertical separation satisfies

$$\left|p_z - q_z\right| \geq d_z^{LB}(A,B). \tag{45}$$

Therefore, if

$$d_z^{LB}(A,B) > D_z^{warn} \tag{46}$$

then all trajectory segment combinations within nodes A and B are guaranteed to remain outside the vertical boundary of the outer warning cylinder. In this case, even if their horizontal projections are close, they cannot simultaneously satisfy the horizontal and vertical intrusion conditions of the warning zone. Since the core zone is contained within the warning zone, the node pair also cannot form a confirmed core conflict and can be directly excluded.

4.2.3. Cylindrical Warning Zone Lower-Bound Pruning Criterion

A retained candidate relation can occur only when the two UAV trajectories overlap in time and may enter the outer warning cylinder. Therefore, time overlap should first be checked before spatial lower-bound pruning is performed. Let the time intervals of nodes A and B be

$$I_t(A) = [t_A^-, t_A^+],\ I_t(B) = [t_B^-, t_B^+] \tag{47}$$

If the two time intervals do not overlap, namely

$$I_t(A) \cap I_t(B) = \varnothing \tag{48}$$

then the two nodes have no simultaneous occupancy relationship within the prediction window and can be directly pruned.

If their time intervals overlap, the horizontal and vertical lower bounds are further evaluated. The node pair (A, B) can be pruned when either of the following two conditions is satisfied:

$$d_h^{LB}(A, B) > D_h^{warn} \text{ or } d_z^{LB}(A, B) > D_z^{warn} \tag{49}$$

The reason is that a retained warning candidate requires both the horizontal distance and the vertical distance to be no greater than their corresponding warning thresholds. If the minimum possible horizontal distance has already exceeded D_h^{warn}, or the minimum possible vertical distance has already exceeded D_z^{warn}, then no trajectory segment pair contained in the two nodes can form a warning zone intrusion. Because the core zone is contained within the warning zone, such a node pair also cannot generate a core zone conflict.

For unified expression, a warning zone lower-bound pruning indicator is defined as

$$\mathcal{P}(A, B) = \begin{cases} 1, & I_t(A) \cap I_t(B) = \varnothing, \\ 1, & d_h^{LB}(A, B) > D_h^{warn}, \\ 1, & d_z^{LB}(A, B) > D_z^{warn}, \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

where $\mathcal{P}(A, B) = 1$ indicates that the node pair can be safely pruned, and $\mathcal{P}(A, B) = 0$ indicates that the node pair still requires further recursive search or leaf-level confirmation.

Based on this criterion, the node search process of AC-DTPM can be expressed as

$$\text{Search}(A, B) = \begin{cases} \text{Prune}(A, B), & \mathcal{P}(A, B) = 1, \\ \text{Search child nodes}, & \mathcal{P}(A, B) = 0. \end{cases} \tag{51}$$

This pruning mechanism is conservative. A node pair is removed only when all trajectory segments inside the two nodes are guaranteed to remain outside the outer warning cylinder in at least one safety dimension, or when they have no time overlap. Therefore, candidate pairs that may form warning zone or core zone intrusions will not be removed by the lower-bound pruning process. For node pairs that cannot be pruned, the algorithm continues recursive search until leaf nodes are reached and then performs continuous cylindrical CPA confirmation on the retained candidate trajectory segments. In the final CPA confirmation stage, the core thresholds are used to identify confirmed core conflict pairs, while the warning thresholds are used to identify warning-only pairs. Figure 4 shows the basic idea of cylindrical warning zone lower-bound pruning.

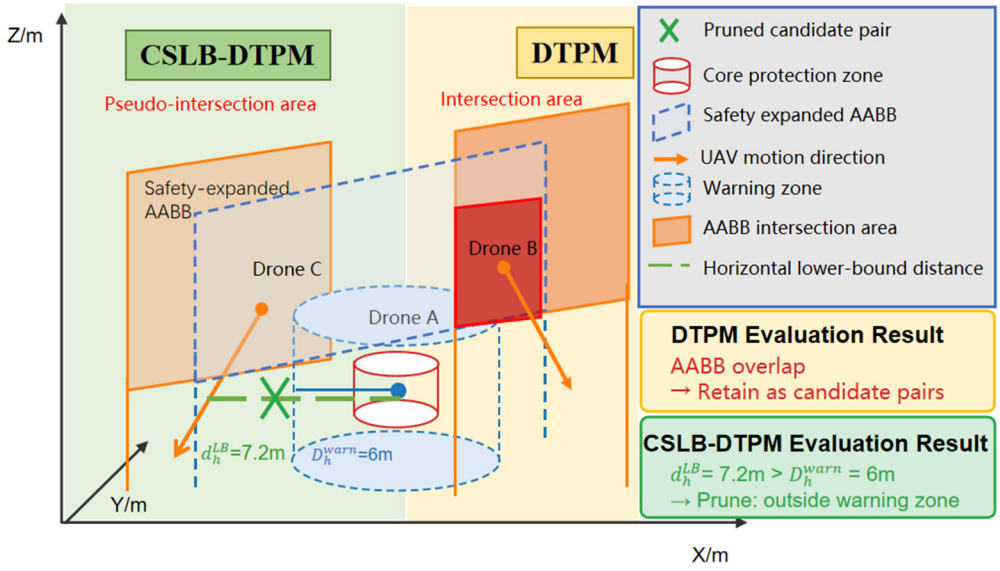


Figure 4. Cylindrical lower-bound pruning.

The lower-bound distance d_h^{LB} is computed from the original unexpanded trajectory node bounds. Although the safety-expanded AABBs overlap in conventional DTPM, the node pair is pruned by CSLB-DTPM because $d_h^{LB} > D_h^{warn}$, indicating that the node pair cannot enter the outer warning cylinder.

4.3. Continuous Cylindrical CPA Confirmation

After the cylindrical lower-bound pruning process, the remaining leaf node trajectory segment pairs are regarded as candidate pairs rather than final conflicts. To avoid missed detections caused by discrete sampling, a continuous closest-point-of-approach (CPA) confirmation is further performed within the overlapping time interval of each candidate segment pair. For two candidate trajectory segments of UAV i and UAV j, let their common time interval be

$$[t_s, t_e] = [t_{i,m}^{\min}, t_{i,m}^{\max}] \cap [t_{j,n}^{\min}, t_{j,n}^{\max}] \tag{52}$$

If $[t_s, t_e] = \varnothing$, the segment pair is directly discarded. Otherwise, the relative position and relative velocity are defined as

$$\begin{aligned} \mathbf{r}_{ij}(t_s) &= \mathbf{p}_i(t_s) - \mathbf{p}_j(t_s) \\ \mathbf{v}_{ij} &= \mathbf{v}_i - \mathbf{v}_j \end{aligned} \tag{53}$$

Under the piecewise linear representation of the dynamically simulated trajectory segments, the CPA time within the overlapping interval can be obtained as

$$t_{\mathrm{CPA}} = \mathrm{clip}\left(t_s - \frac{\mathbf{r}_{ij}(t_s)^T \mathbf{v}_{ij}}{\|\mathbf{v}_{ij}\|^2 + \varepsilon},\ t_s,\ t_e \right) \tag{54}$$

where ε is a small positive value used to avoid numerical instability when the relative velocity is close to zero, and $\mathbf{clip}(\cdot)$ restricts the CPA time to the valid overlapping interval. The horizontal and vertical distances at $\boldsymbol{t_{\mathrm{CPA}}}$ are then calculated as

$$\begin{aligned} d_{xy}^{\mathrm{CPA}} &= \|\mathbf{p}_{i,xy}(t_{\mathrm{CPA}}) - \mathbf{p}_{j,xy}(t_{\mathrm{CPA}})\|_2 \\ d_z^{\mathrm{CPA}} &= \left| z_i(t_{\mathrm{CPA}}) - z_j(t_{\mathrm{CPA}}) \right| \end{aligned} \tag{55}$$

Accordingly, a cylindrical protection zone conflict is confirmed only when the horizontal and vertical separation conditions are simultaneously violated [24]:

$$d_{xy}^{\mathrm{CPA}} \le D_{xy},\ d_z^{\mathrm{CPA}} \le D_z \tag{56}$$

For the double-layer protection zone model, the above criterion is applied with different threshold pairs. If $(D_{xy}, D_z) = (D_{xy}^{core}, D_z^{core})$, the pair is identified as a confirmed core conflict. If $(D_{xy}, D_z) = (D_{xy}^{warn}, D_z^{warn})$, the pair is identified as a warning conflict candidate. Therefore, the proposed AC-DTPM first uses adaptive segmentation and cylindrical lower-bound pruning to reduce redundant candidates and then applies continuous cylindrical CPA confirmation to ensure that the final conflict set is determined by a continuous-time geometric criterion rather than by discrete trajectory sampling alone.

Algorithm 1 summarizes the overall implementation procedure of AC-DTPM. The proposed method first generates adaptive four-dimensional trajectory segments according to UAV speed and local density and then constructs a decision tree structure for hierarchical candidate search. During tree traversal, conventional bounding-box overlap pruning is combined with cylindrical safety lower-bound pruning. The lower-bound distances are computed from the original node bounds before safety expansion, and the results are compared with the cylindrical safety thresholds. Finally, only the retained candidate UAV pairs are verified by continuous cylindrical CPA confirmation, which produces the final core conflict and warning conflict sets.

Algorithm 1. Overall procedure of the proposed AC-DTPM

Initialize trajectory segment set SA, candidate pair set Cp, core conflict set Ccore, and warning conflict set Cwarn.
for each UAV i = 1, 2, . . ., N do
    Compute the number of local neighboring UAVs within R_ρ and H_ρ.
    Normalize the local density ρ_i.
    Compute the adaptive time step Δt_i according to UAV speed and local density.
    Generate adaptive prediction time nodes within $[t_0, t_0 + T_h]$.
    Construct four-dimensional trajectory segments for UAV i.
    Build the original segment bounds and safety-expanded bounding boxes.
    Add the generated trajectory segments into $\mathcal{S}^A$.
end for
Construct the decision tree structure using the trajectory segment set $\mathcal{S}^A$.
for each pair of decision tree nodes A and B do
    if the time intervals of A and B do not overlap then
        Prune the node pair directly.
    else if the safety-expanded bounding boxes do not overlap then
        Prune the node pair directly.
    else
        Compute the horizontal and vertical lower-bound distances using the original node bounds before safety expansion.
        if $d_{xy}^{LB}(A,B) > D_{xy}^{\mathrm{warn}}$ or $d_z^{LB}(A,B) > D_z^{\mathrm{warn}}$ then
            Prune the node pair by cylindrical safety lower-bound pruning.
        else if A and B are leaf nodes then
            Add the corresponding UAV pair into the candidate-pair set C_p.
        else

Algorithm 1. *Cont.*

Continue recursive search on the child-node pairs.
end if
end if
end for
Remove duplicate UAV pairs from C_p.
for each candidate UAV pair (i, j) in C_p do
Perform continuous cylindrical CPA confirmation within the overlapping prediction interval.
if $d_{xy}^{\mathrm{CPA}} \le D_{xy}^{\mathrm{core}}$ and $d_z^{\mathrm{CPA}} \le D_z^{\mathrm{core}}$ then
Add (i, j) into C_{core}.
else if $d_{xy}^{\mathrm{CPA}} \le D_{xy}^{\mathrm{warn}}$ and $d_z^{\mathrm{CPA}} \le D_z^{\mathrm{warn}}$ then
Add (i, j) into C_{warn}.
end if
end for
Return C_{core} and C_{warn}.

4.4. Computational Complexity and Memory Cost Analysis

The computational cost of multi-UAV conflict detection mainly depends on the number of UAVs, the number of generated trajectory segments, the number of visited tree node pairs, and the number of final candidate pairs requiring CPA confirmation. Let N denote the number of UAVs, and let M_i denote the number of trajectory segments generated for the i-th UAV. The total number of trajectory segments is given by

$$S = \sum_{i=1}^{N} M_i \tag{57}$$

where S denote the total number of trajectory segments. For exhaustive pairwise CPA detection, if each UAV pair is checked once over the prediction window, the computational complexity is $O(N^2)$. If segment-level exhaustive comparison is adopted, the computational cost further depends on the number of trajectory segments [25] and can be expressed as

$$O\left(\sum_{i<j} M_i M_j\right) \tag{58}$$

When all UAVs have approximately the same number of trajectory segments, i.e., $M_i \approx M$, the above complexity becomes $O(N^2M^2)$ [26]. Therefore, exhaustive detection becomes inefficient in dense low-altitude multi-UAV scenarios, especially when the prediction window is divided into many trajectory segments.

For the basic DTPM method, the main computational steps include trajectory segment generation, decision tree construction, tree node traversal, and CPA confirmation for retained candidate pairs. Assuming that a balanced tree structure is constructed, the tree construction cost is approximately $O(S \log S)$. Let V_D denote the number of visited node pairs during tree traversal, and let C_D denote the number of retained candidate pairs requiring final CPA confirmation. The computational complexity of basic DTPM can then be written as

$$O(S \log S + V_D + C_D) \tag{59}$$

Compared with exhaustive pairwise detection, DTPM reduces the computational burden by replacing full pairwise checking with hierarchical pruning. However, because conventional DTPM mainly relies on axis-aligned bounding-box overlap, many pseudo-

intersection cases may still be retained in dense airspace, resulting in relatively large V_D and C_D.

For the proposed AC-DTPM, the adaptive time-segmentation mechanism changes the number of trajectory segments according to UAV speed and local density. Let M_i^A denote the adaptively generated number of trajectory segments for the i-th UAV. The total number of adaptive trajectory segments is

$$S_A = \sum_{i=1}^{N} M_i^A \tag{60}$$

The local density calculation introduces an additional preprocessing cost. If direct pairwise neighborhood checking is used, its upper-bound complexity is $O(N^2)$. When a grid index or spatial search structure is used, the practical cost can be reduced to approximately $O(N + N\bar{k})$, where $\bar{k}$ is the average number of nearby UAVs queried for each aircraft. Let this preprocessing cost be denoted as T_ρ.

$$O(T_\rho + S_A \log S_A + V_{AC} + C_{AC}) \tag{61}$$

In the worst case, AC-DTPM has the same asymptotic order as the DTPM-based search framework, because all node pairs may still need to be visited under extremely dense or highly overlapping conditions. However, in practical low-altitude traffic scenarios, adaptive segmentation and cylindrical lower-bound pruning reduce the empirical values of V_{AC} and C_{AC}. Therefore, AC-DTPM improves practical detection efficiency mainly by reducing invalid tree node visits and redundant CPA confirmations, rather than by changing the worst-case theoretical complexity order.

The memory cost of AC-DTPM mainly consists of UAV state storage, trajectory segment storage, decision tree node storage, and retained candidate pair storage. The UAV state storage requires $O(N)$ memory. The trajectory segment set, and the corresponding decision tree nodes require $O(SA)$ memory, because each trajectory segment corresponds to a bounded spatiotemporal element in the tree. The retained candidate pair list requires $O(C_{AC})$ memory. Therefore, the total memory cost of AC-DTPM can be written as

$$O(N + S_A + C_{AC}) \tag{62}$$

If a grid index is used for local density query, an additional grid-storage term can be included. Let G denote the number of occupied or maintained grid cells. The memory cost then becomes

$$O(G + N + S_A + C_{AC}) \tag{63}$$

In most large-scale UAV scenarios, the trajectory segment and candidate pair storage dominate the UAV state storage, so the memory cost can be approximately regarded as $O(S_A + C_{AC})$. Unlike dense pairwise relation methods, AC-DTPM does not need to store a complete $N \times N$ conflict matrix during the whole detection process. Instead, only the trajectory tree structure and the candidate pairs retained after pruning are stored for final CPA confirmation. This makes the proposed method more memory-efficient when the candidate compression ratio remains low.

5. Simulation Experiments and Result Analysis

5.1. Simulation Setup

All simulation experiments in this paper were implemented in the MATLAB R2023a environment. To ensure the using of comparisons among different algorithms, all algorithms were evaluated under the same airspace, UAV number, initial states, and simulation parameters. The main simulation parameters are listed in Table 1.

Table 1. Main simulation parameter settings.

Parameter	Value	Description
Airspace size (m)	$1200 \times 1200 \times 80$	Three-dimensional cuboid research airspace
Flight altitude range (m)	30–70	Valid altitude range for UAV operations
Number of UAVs	100–800	Large-scale scenario testing
Horizontal velocity range (m/s)	8–15	UAV cruising speed range
Vertical velocity range (m/s)	$[-1.5, 1.5]$	Climb/descent velocity constraint
Prediction time window (s)	6	Look-ahead window for dynamic conflict detection
Minimum time step (s)	0.25	Lower bound for adaptive time segmentation
Maximum time step (s)	0.75	Upper bound for adaptive time segmentation
Core safety threshold (m)	$D_h^{core} = 3.0 \quad D_z^{core} = 2.0$	Used for confirmed core conflict classification in CPA confirmation
Warning safety threshold (m)	$D_h^{warn} = 6.0 \quad D_z^{warn} = 4.0$	Used for lower-bound pruning and warning-level classification
Number of repeated experiments	10	Statistical averaging of results

To simulate local congestion in low-altitude multi-UAV operations, the experimental scenario does not adopt a completely uniform random distribution. Instead, local dense regions and crossing-route areas are introduced on the basis of globally randomized UAV initial positions, so that some UAVs exhibit spatial clustering. This setting enables a more comprehensive evaluation of the candidate screening capability and real-time detection performance of the proposed algorithm in locally high-density airspace. The resulting XY-plane occupancy density distribution of the mixed multi-UAV scenario is shown in Figure 5.

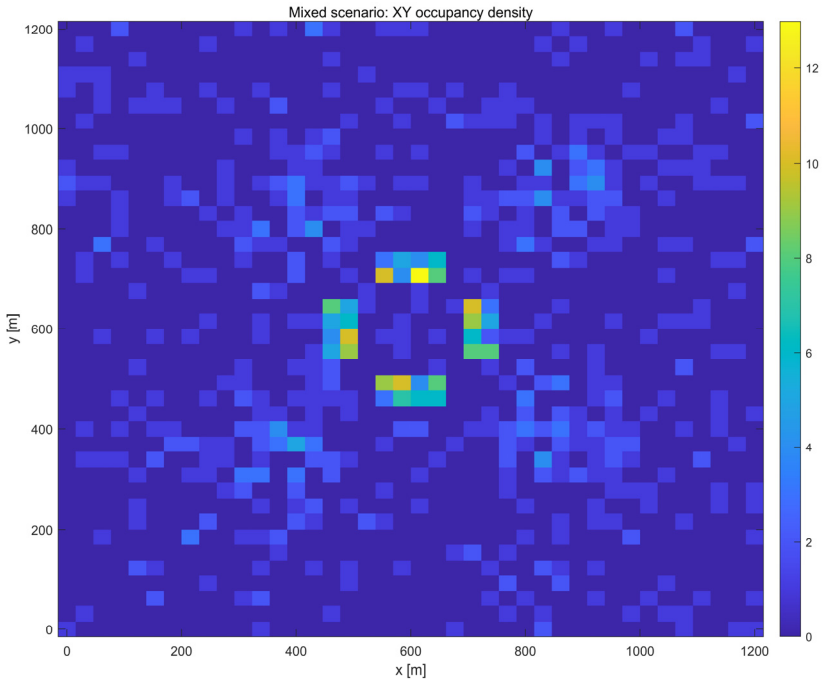


Figure 5. Airspace occupancy density distribution in the mixed low-altitude UAV scenario.

5.2. Experiment 1: Realistic UAV Motion Model Verification

As illustrated in Figure 6, the trajectory-tracking results show that the realistic UAV generally follows the spatial trend of the reference trajectory, while visible deviations occur around curved segments. These deviations are caused by the finite acceleration and yaw-response capabilities of the UAV, rather than numerical instability. As summarized in Table 2, dynamic response profiles show that the maximum ground speed reaches 15.30 m/s, with an average value of 12.52 m/s. The horizontal acceleration reaches its imposed upper bound of 5.00 m/s^2 during high-curvature and speed-transition segments, indicating that the UAV enters acceleration saturation conditions in these regions. The climb and descent rates remain within 1.05 m/s and 0.98 m/s, respectively, which are below the imposed vertical motion limits. The maximum yaw rate is 22.48°/s, also below the prescribed 35.00°/s limit. These results indicate that the UAV model can execute the reference trajectory within the defined physical constraints, but its motion cannot be approximated as an ideal point-mass response with instantaneous velocity and heading changes.

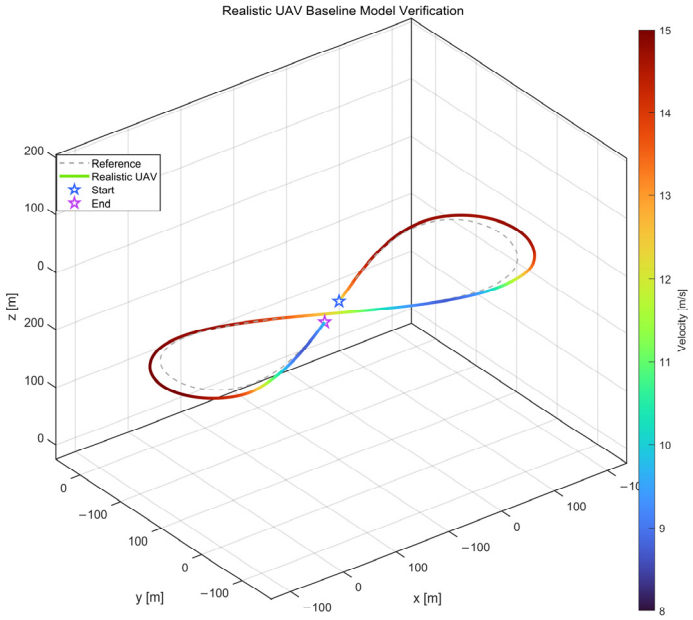


Figure 6. Baseline verification of the realistic UAV motion model.

Combined with the horizontal acceleration and vertical-rate constraint verification in Figure 7, the control response and error statistics further show that short-term prediction and safety assessment should account for dynamic lag and sensing uncertainty. The average speed command lag is 1.25 m/s, and the maximum lag reaches 4.04 m/s, mainly occurring during rapid speed transitions and curved flight segments. As shown in Figure 8, the RMS tracking error is 5.06 m, while the recorded average tracking error is 7.33 m, reflecting the cumulative effect of acceleration saturation, response delay, and heading adjustment. The average and maximum positioning errors are 0.53 m and 1.59 m, respectively, indicating that position uncertainty is non-negligible for meter-level separation assessment. Therefore, the subsequent experiments use the verified UAV operating envelope, including the 8–15 m/s speed range, bounded vertical motion, response delay, and positioning error characteristics, as the basis for safety threshold calibration and large-scale AC-DTPM conflict detection evaluation. Detailed physical, dynamic, simulation, and reference-trajectory parameters are provided in Appendix A.

Table 2. Verification results of the realistic UAV motion model.

Indicator	Value	Constraint
Maximum Ground Speed	15.30 m/s	≤18.00 m/s
Average Ground Speed	12.52 m/s	Recorded Value
Maximum Horizontal Acceleration	5.00 m/s^2	≤5.00 m/s^2
Maximum Climb Rate	1.05 m/s	≤3.00 m/s
Maximum Descent Rate	0.98 m/s	≤2.50 m/s
Maximum Yaw Rate	22.48°/s	≤35.00°/s
Average Speed-Command Lag	1.25 m/s	Recorded Value
Maximum Speed Command Delay	4.04 m/s	Recorded Value
Root Mean Square (RMS) Tracking Error	5.06 m	Recorded Value
Average Tracking Error	7.33 m	Recorded Value
Average Positioning Error	0.53 m	Recorded Value
Maximum Positioning Error	1.59 m	Recorded Value

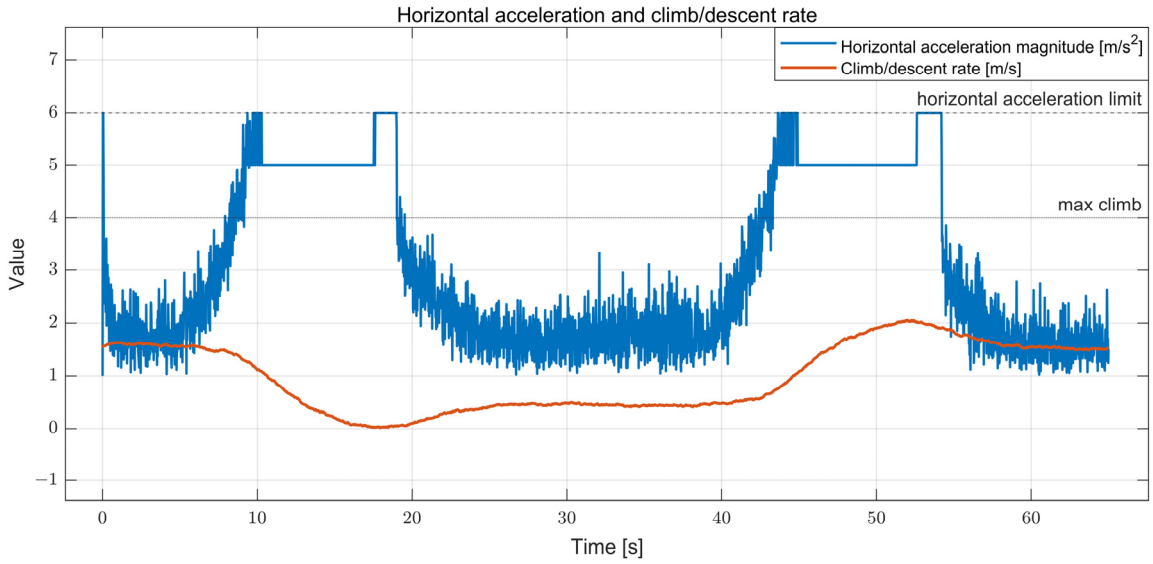


Figure 7. Verification of horizontal acceleration and vertical-rate constraints.

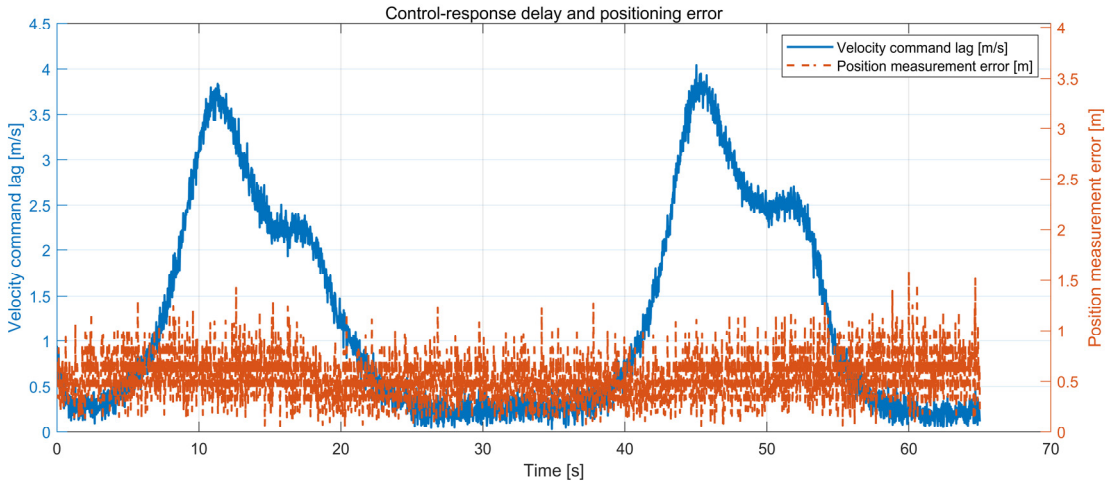


Figure 8. Velocity command lag and positioning error of the realistic UAV model.

5.3. Experiment 2: Safety Threshold Calibration and Protection Zone Shape Verification

To justify the safety model used in the proposed AC-DTPM, this experiment was designed with two connected objectives: first, to calibrate the horizontal and vertical safety thresholds under realistic low-altitude UAV motion constraints; and second, to verify whether the double-layer cylindrical protection zone geometry is more suitable than other representative protection zone shapes. Both parts were conducted using the same realistic UAV motion envelope, positioning error setting, prediction horizon, and typical encounter scenarios.

Figure 9 shows the seven low-altitude UAV encounter scenarios used in this experiment, including head-on, crossing, overtaking, vertical convergence, cluster, wind-disturbed, and delay-disturbed encounters. These scenarios were used both for safety threshold calibration and for protection zone shape verification, so that the calibrated threshold values and the selected protection zone geometry were evaluated under consistent conflict conditions.

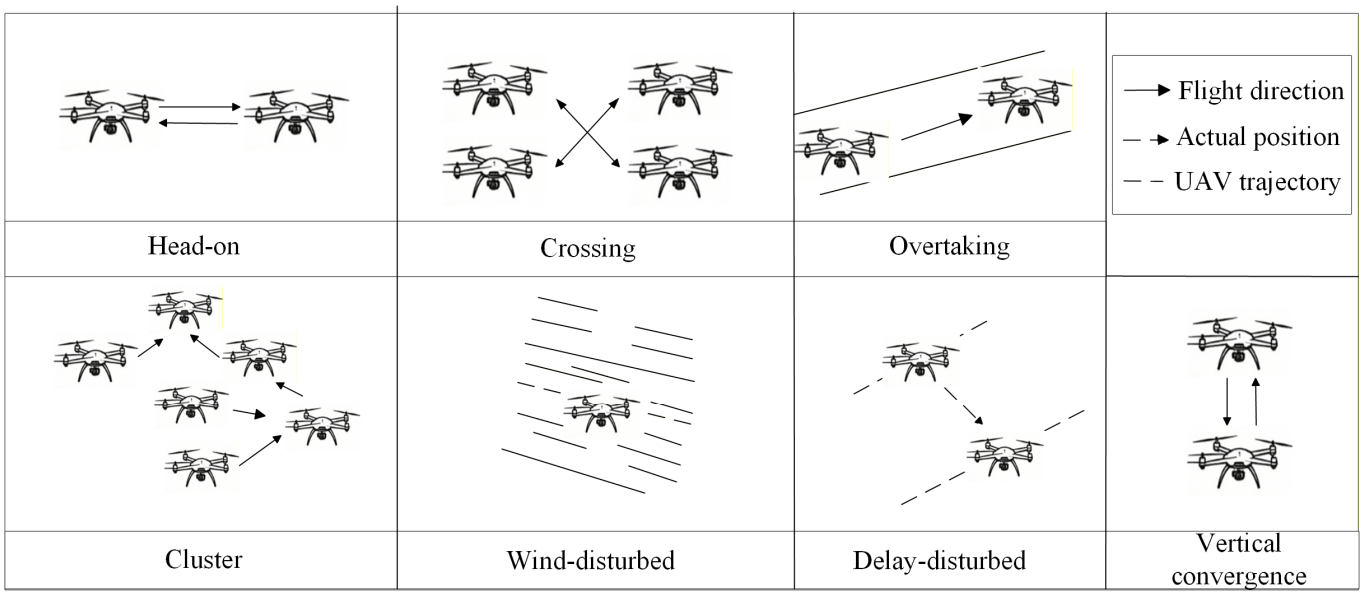


Figure 9. Typical low-altitude UAV encounter scenarios used for protection zone verification.

For safety threshold calibration, 1440 candidate threshold combinations were evaluated, and each combination was tested over 14 repeated Monte Carlo encounter simulations, resulting in 20,160 total runs. The threshold selection was based on multiple criteria rather than manual assignment: the core zone should be as compact as possible while satisfying collision safety, near-miss safety, detection completeness, and response feasibility, whereas the warning zone should provide sufficient lead time without causing excessive operational cost.

As shown in Table 3, the selected threshold combination achieves zero collision, zero near-miss after response, zero missed dangerous encounters, and a response feasibility of 1.000. The fifth percentile warning lead time reaches 5.57 s, which is higher than the 1 s minimum requirement, while the operational cost remains at 7.18%. Therefore, the final double-layer safety thresholds are set as $D_h^{core} = 3.0$ m, $D_z^{core} = 2.0$ m, $D_h^{warn} = 6.0$ m, and $D_z^{warn} = 4.0$ m. In this setting, the inner layer is used for strict conflict confirmation, while the outer layer provides a conservative candidate-preservation boundary and preserves early-warning information for downstream prioritization.

Table 3. Safety threshold calibration results.

Indicator Category	Result	Interpretation
Core compactness	18	Smallest eligible core zone with the adopted vertical safety floor
Warning compactness	144	Moderate outer warning zone: (6, 4) m
Collision safety	0.000	No physical collision observed
Near-miss safety	0.000	No independent near-miss event after response
Completeness	0.000	No dangerous encounter was missed
Response feasibility	1.000	All detected dangerous cases retained feasible response
Warning timeliness	5.57 s	Conservative lead-time margin for downstream resolution
Operational cost	7.18%	Acceptable deviation for a compact core zone setting

After safety threshold calibration, four protection zone geometries were compared under the same calibrated safety range and encounter scenarios: spherical, ellipsoidal, single-layer cylindrical, and double-layer cylindrical zones. The evaluation considered missed dangerous encounter rate, confirmed false alarm rate, fifth percentile lead time, mean candidate pair count, and candidate redundancy rate, aiming to identify the geometry most compatible with AC-DTPM screening.

As shown in Table 4, all models achieved zero missed dangerous encounters, but their operational performance differed. The spherical zone produced the highest false alarm rate and candidate count due to isotropic over-expansion, while the ellipsoidal zone was less compatible with separated horizontal–vertical lower-bound pruning. The single-layer cylinder produced the fewest candidates but lacked an independent warning layer. In contrast, the double-layer cylinder maintained zero missed dangerous encounters, provided the longest fifth percentile lead time, and kept a moderate candidate count. Therefore, it was selected as the protection zone model for AC-DTPM, because it supports both strict core conflict confirmation and conservative warning-level candidate preservation.

Table 4. Protection zone shape comparison.

Safety Zone Model	Missed Dangerous Encounter Rate	Confirmed False Alarm Rate	5% Lead Time	Mean Candidate Pair Count	Candidate Redundancy Rate
Double-layer cylinder	0	0.145	5.475 s	2.50	0.050
Ellipsoidal zone	0	0.137	4.800 s	2.44	0.057
Single-layer cylinder	0	0.152	5.075 s	1.61	0
Spherical zone	0	0.17	5.275 s	2.89	0.050

5.4. Experiment 3: Main Performance Evaluation of AC-DTPM

To evaluate the detection efficiency and completeness of the proposed AC-DTPM, large-scale multi-UAV conflict detection experiments were conducted, with the number of UAVs increasing from 100 to 800. All methods were tested using the same airspace, random seeds, UAV speed range, prediction horizon, and calibrated double-layer cylindrical safety thresholds. Exhaustive continuous CPA was used as the ground-truth reference for missed-detection evaluation, while the DTPM-family methods were compared in terms

of candidate compression ratio, total detection time, visited node pairs, and confirmed conflict completeness.

To verify the independent contributions and synergistic effects of the two improvement mechanisms in the proposed AC-DTPM method, four groups of comparative experiments were designed in this paper:

- DTPM: the basic decision-tree pruning method, which adopts a fixed time step and axis-aligned bounding-box pruning.
- ATS-DTPM: an improved method that introduces only the adaptive time segmentation mechanism.
- CSLB-DTPM: an improved method that introduces only the cylindrical safety lower-bound pruning mechanism.
- AC-DTPM: the complete method that incorporates both improvement mechanisms.

As shown in Figure 10, AC-DTPM consistently produces the lowest candidate compression ratio among the DTPM-family methods as the UAV number increases. This reduction is not caused by relaxing the safety criterion, because all methods retain the same confirmed conflict pairs, and no missed detection is observed. Instead, the improvement comes from two mechanisms. First, adaptive time segmentation reduces redundant trajectory segments in low-risk or low-density regions, thereby decreasing unnecessary node comparisons. Second, cylindrical lower-bound pruning removes node pairs that are guaranteed to remain outside the outer warning zone in either the horizontal or vertical safety dimension before expensive CPA confirmation is performed. This explains why ATS-DTPM improves over basic DTPM by reducing segmentation redundancy, why CSLB-DTPM further reduces invalid node pair traversal through geometric pruning, and why AC-DTPM achieves the strongest compression by combining both mechanisms. Detailed candidate screening and CPA confirmation metrics at the 800-UAV scale are summarized in Table 5. At 800 UAVs, exhaustive CPA evaluates 319,600 pairs, whereas AC-DTPM only evaluates 178.8 CPA candidate pairs on average, while preserving all 66 true conflict pairs. Its false candidate ratio is reduced to 0.632, lower than those of basic DTPM, ATS-DTPM, and CSLB-DTPM.

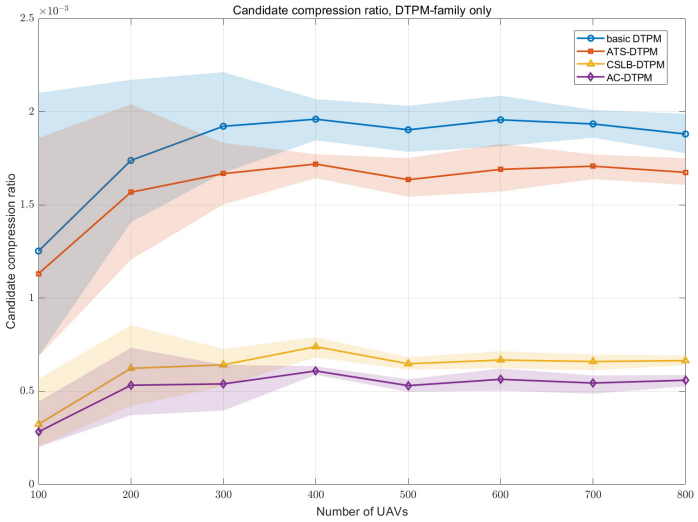


Figure 10. Comparison of candidate compression ratios at different UAV scales.

The detection time and visited node results further confirm the efficiency advantage of AC-DTPM. As the number of UAVs increases, the total detection time of all DTPM-family methods increases, but AC-DTPM remains the fastest among them. At 800 UAVs, exhaustive CPA requires 280.959 s, while AC-DTPM reduces the CPA confirmation time to 0.063 s. The visited node- pair curve shows the same mechanism-level trend: basic DTPM visits the largest number of node pairs, because it mainly depends on conservative

bounding-box overlap; ATS-DTPM reduces part of the redundancy through adaptive segmentation; CSLB-DTPM removes more invalid node pairs through cylindrical lower-bound pruning; and AC-DTPM achieves the lowest visited node count by combining both mechanisms. Therefore, the main advantage of AC-DTPM is not merely fewer final CPA checks, but earlier elimination of impossible conflicts at the decision tree search stage. These results demonstrate that AC-DTPM provides an effective incremental improvement over the basic DTPM pipeline by reducing candidate redundancy and detection time while maintaining complete conflict detection.

Table 5. Candidate screening and CPA confirmation performance at the 800-UAV scale.

Method	CPA-Evaluated Pairs	True Conflict Pairs	Confirmed Pairs	False Candidate Ratio	CPA Time (s)
Exhaustive CPA	319,600.0	66	66	—	280.959
Basic DTPM	601.0	66	66	0.891	0.248
ATS-DTPM	535.2	66	66	0.877	0.213
CSLB-DTPM	212.4	66	66	0.691	0.079
AC-DTPM	178.8	66	66	0.632	0.063

Figure 11 compares the total detection time of different DTPM variants. As the number of UAVs increases, all methods require more detection time. However, AC-DTPM consistently maintains the lowest runtime, because adaptive time segmentation reduces redundant trajectory representation, and cylindrical lower-bound pruning removes invalid candidate pairs before CPA confirmation. The advantage becomes more obvious in large-scale scenarios, indicating better scalability of the proposed method.

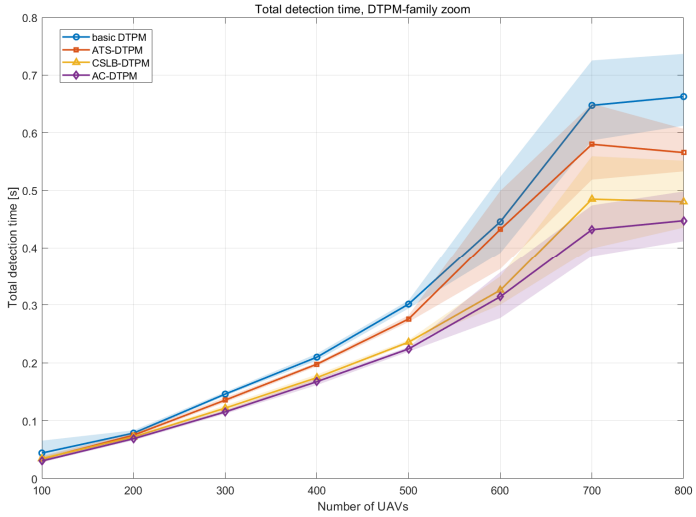


Figure 11. Total detection time of DTPM-family methods.

The local conflict cluster visualization further illustrates the practical meaning of the proposed screening process. The identified high-risk cluster contains both confirmed core conflict pairs and warning-only pairs, indicating that the double-layer cylindrical model does not treat all candidate pairs as identical. Core conflict pairs are retained for strict conflict confirmation, while warning-only pairs are preserved as early-risk information for downstream prioritization or resolution. The risk-based hierarchical ranking result further shows how the detected conflict information can be organized after AC-DTPM screening. As shown in Figure 12, the top-ranked conflict clusters exhibit high maximum risk values, while their mean risk values differ across clusters. This indicates that a cluster with a severe

pairwise conflict is not necessarily identical to a cluster with consistently high overall risk. For example, some clusters have a high maximum risk but a relatively moderate mean risk, suggesting that the risk is dominated by one or two critical UAV pairs. In contrast, clusters with both high maximum and high mean risk contain more uniformly distributed conflict pressure and should receive higher priority in downstream conflict resolution scheduling. Therefore, the ranking module provides an additional decision layer beyond binary conflict detection: AC-DTPM first identifies candidate and confirmed conflict relations efficiently, and the hierarchical ranking then prioritizes local conflict clusters according to their risk intensity and spatial concentration. As illustrated in Figure 13, this makes the proposed framework suitable not only for fast conflict screening, but also for structured risk-aware conflict management in dense low-altitude UAV operations.

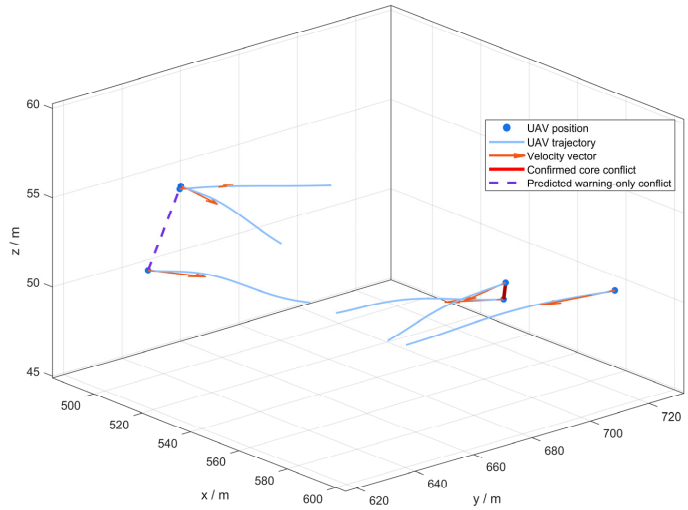


Figure 12. Visualization of confirmed core conflicts and warning-only conflicts in a high-risk local cluster.

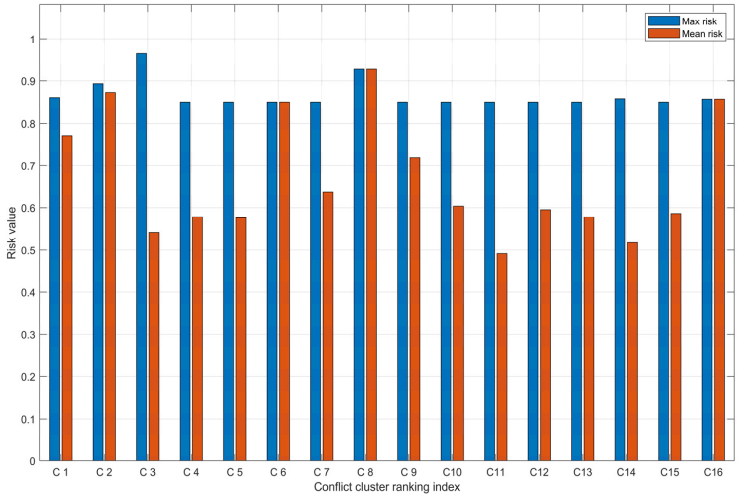


Figure 13. Risk stratification ranking results of conflict clusters.

5.5. Experiment 4: Complexity and Memory Cost Evaluation

To further evaluate the scalability of the proposed method, the number of visited node pairs and the estimated memory cost are analyzed at different UAV scales. The visited node pairs reflect the actual search burden of the decision tree pruning process, while the memory cost is estimated from the main retained data structures, including UAV states, trajectory segments, tree nodes, and candidate pair lists.

As shown in Figure 14, the number of visited node pairs increases with the number of UAVs for all methods. Basic DTPM has the largest search burden, because it mainly relies on conventional bounding-box overlap and retains many pseudo-intersection cases. ATS-DTPM and CSLB-DTPM reduce the search burden from adaptive segmentation and cylindrical lower-bound pruning, respectively. AC-DTPM achieves the lowest number of visited node pairs by combining both mechanisms, and its advantage becomes more evident at larger UAV scales.

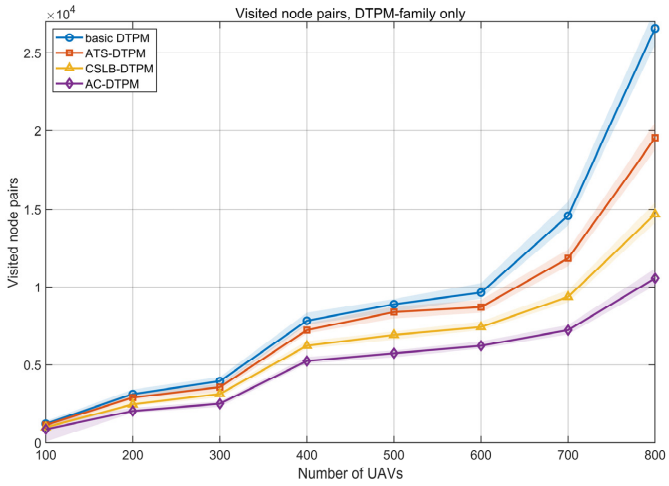


Figure 14. Comparison of number of visited node pairs at different UAV scales.

Figure 15 shows that the memory cost also increases with UAV scale. ATS-DTPM may consume slightly more memory at large scales, because finer segmentation is used in dense or high-speed regions. In contrast, CSLB-DTPM and AC-DTPM reduce redundant candidate storage through lower-bound pruning. Overall, AC-DTPM maintains lower memory cost while preserving complete conflict detection, indicating that the proposed method improves both search efficiency and storage efficiency.

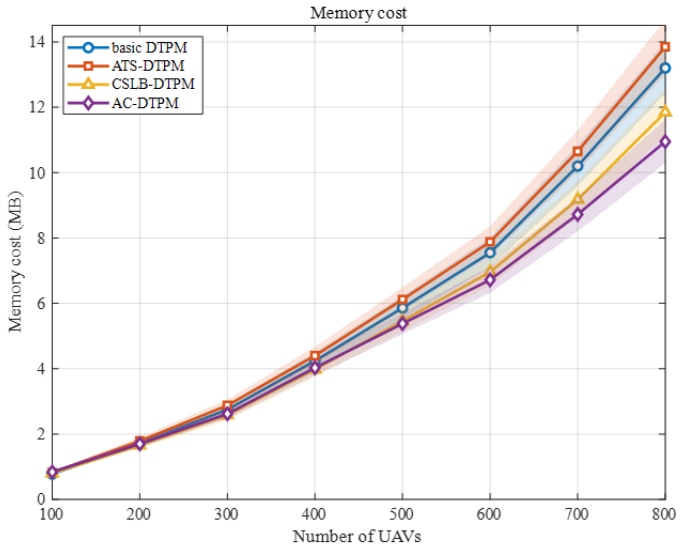


Figure 15. Memory cost growth at different UAV scales.

5.6. Experiment 5: Comparison with Representative Baseline Variants

To further evaluate the competitiveness of AC-DTPM, two recent representative baselines were selected: PU-STG and MG-RL. PU-STG represents an uncertainty-aware spatiotemporal replanning direction inspired by Chen et al., where UAV position uncertainty

is considered in conflict-free trajectory replanning. MG-RL represents a graph-enhanced reinforcement-learning direction inspired by Li et al., where multiscale graph representation and learned policies are used for conflict resolution in dense UAV networks. These two baselines cover two relevant state-of-the-art research directions: uncertainty-aware optimization-based replanning and learning-based multi-UAV conflict resolution [27,28].

To ensure a fair and meaningful comparison, PU-STG and MG-RL are re-implemented as detection-oriented baselines under a unified evaluation framework. Although originally designed for trajectory replanning and multi-agent decision-making, their core principles—uncertainty-aware prediction in PU-STG and graph-based interaction modeling in MG-RL—are preserved, while their outputs are adapted to conflict candidate screening. All methods share identical UAV trajectories, motion constraints, safety thresholds, and prediction windows, ensuring consistency in input conditions. Since PU-STG and MG-RL do not explicitly define candidate compression or memory usage, unified abstraction is adopted to derive comparable metrics based on pairwise interaction evaluation and runtime state storage. This ensures that performance differences reflect algorithmic characteristics rather than implementation inconsistencies. The comparison was conducted at N = 600, and the results are summarized in Table 6.

Table 6. Comprehensive comparison of multiple conflict detection algorithms.

Method	Missed Detection Rate	Candidate Redundancy Rate	Candidate Compression Ratio	Avg Detection Time (s)	Memory Cost (MB)
PU-STG	0.000	0.985	0.014080	1.82	12.52
MG-RL	0.010	0.873	0.002816	0.255	21.92
AC-DTPM	0.000	0.632	0.000559	0.447	6.26

As shown in Table 6, PU-STG achieves zero missed detection in this experiment, which can be attributed to its conservative uncertainty-aware candidate retention strategy. However, this conservativeness also leads to the highest candidate redundancy rate and the largest candidate compression ratio, indicating that many retained candidates do not finally correspond to true conflicts. MG-RL obtains the shortest average detection time, which is consistent with the fast online inference property of a trained reinforcement-learning policy. Nevertheless, it shows a small missed detection rate and the highest memory cost, mainly because graph features and policy network parameters need to be maintained during inference. In contrast, AC-DTPM is not the fastest method in absolute runtime, but it achieves zero missed detection, the lowest candidate redundancy rate, the smallest candidate compression ratio, and the lowest memory cost. These results suggest that AC-DTPM provides a more balanced trade-off for safety-critical low-altitude conflict detection, where detection completeness, candidate-screening compactness, and controlled memory usage are more important than inference speed alone.

5.7. Experiment 6: Parameter Sensitivity Analysis

In the experiments, the number of UAVs was fixed at 600, while the prediction time window length Th, density weighting coefficient, and minimum time step $\Delta t_{\min}$ were varied separately. The average number of candidate UAV pairs, candidate compression ratio, number of leaf nodes, and average total detection time were then recorded. The experimental results are shown in Table 7.

1. Prediction time window Th: The prediction time window has a direct influence on the retained candidate scale and the total detection time. When T_h increases from 4 s to 8 s, the average number of candidate UAV pairs increases from 72.0 to 148.0,

and the average total detection time increases from 0.238 s to 0.461 s. This is because a longer prediction horizon covers a larger future trajectory range and retains more trajectory segments in the decision tree, thereby increasing the probability of bounding-box overlap and candidate pair generation. Although a longer prediction window can provide earlier conflict awareness, it also increases the computational burden. Therefore, $T_h = 6$ s is selected as a balanced setting between look-ahead capability and detection efficiency.

2. Density weighting coefficient λ_ρ: The density weighting coefficient controls the influence of local traffic density on adaptive time segmentation. When λ_ρ increases from 0 to 0.8, the average number of candidate UAV pairs decreases from 126.0 to 94.8, and the candidate compression ratio decreases from 0.701×10^{-3} to 0.528×10^{-3}. This indicates that stronger density-aware segmentation can reduce candidate redundancy in locally crowded regions. However, the number of leaf nodes increases from 448 to 640, and the average total detection time increases from 0.303 s to 0.366 s, because finer segmentation introduces additional tree nodes and traversal overhead. The default value $\lambda_\rho = 0.4$ achieves a compromise between candidate reduction and computational cost.
3. Minimum time step $\Delta t_{\min}$: The minimum time step determines the lower bound of trajectory segment length in adaptive segmentation. When $\Delta t_{\min}$ is reduced from 0.40 s to 0.20 s, the number of leaf nodes increases from 384 to 768, and the average total detection time increases from 0.277 s to 0.412 s. Meanwhile, the average number of candidate UAV pairs decreases from 122.0 to 88.0, and the candidate compression ratio decreases from 0.679×10^{-3} to 0.490×10^{-3}. This shows that a smaller minimum time step improves the temporal resolution of predicted trajectories and makes the trajectory bounding boxes tighter, thereby reducing candidate redundancy. However, excessive segmentation increases tree size and computational overhead. Therefore, $\Delta t_{\min} = 0.30$ is adopted as a moderate setting.

Table 7. Parameter sensitivity analysis of AC-DTPM.

Parameter Setting	Average Number of Candidate UAV Pairs	Candidate Compression Ratio/$\times 10^{-3}$	Number of Leaf Nodes	Average Total Detection Time (s)
Th= 4 s λ_ρ= 0.4 $\Delta t_{\min}$= 0.3 s	72	0.401	384	0.238
Th= 6 s λ_ρ= 0.4 $\Delta t_{\min}$= 0.3 s	101.4	0.564	512	0.314
Th= 8 s λ_ρ= 0.4 $\Delta t_{\min}$= 0.3 s	148.0	0.824	704	0.461
Th= 6 s λ_ρ= 0 $\Delta t_{\min}$= 0.3 s	126.0	0.701	448	0.303
Th= 6 s λ_ρ= 0.8 $\Delta t_{\min}$= 0.3 s	94.8	0.528	640	0.366
Th= 6 s λ_ρ= 0.4 $\Delta t_{\min}$= 0.2 s	88.0	0.490	768	0.412
Th= 6 s λ_ρ= 0.4 $\Delta t_{\min}$= 0.4 s	122.0	0.679	384	0.277

6. Conclusions

This study investigated fast and reliable conflict detection for dense low-altitude multi-UAV operations. Using realistic UAV motion constraints, positioning errors, response delays, and locally congested airspace conditions, conventional pairwise detection and basic DTPM-based methods may retain many redundant candidate pairs, which increases computational burden and reduces real-time applicability. To address this problem, this paper proposed an Adaptive Cylindrical-Pruning Decision Tree Method, namely AC-DTPM, which improves the basic DTPM framework through adaptive time segmentation and cylindrical warning-zone lower-bound pruning based on a double-layer cylindrical protection zone model.

The experimental results show that AC-DTPM improves candidate-screening compactness while preserving detection completeness in the tested scenarios. At the scale of 800 UAVs, AC-DTPM reduced the number of evaluated candidate pairs from 601.0 in basic DTPM to 178.8, corresponding to a 70.3% reduction and a candidate compression ratio of 5.59×10^{-4}. The total detection time was reduced by 32.5%, and the memory cost was also reduced compared with basic DTPM. With the exhaustive continuous cylindrical CPA reference used in this study, AC-DTPM maintained a zero missed-detection rate in all tested scenarios, indicating that the proposed pruning strategy improves practical detection efficiency without sacrificing detection completeness.

The main contribution of this study lies in improving the DTPM-based conflict detection pipeline for dense low-altitude UAV scenarios. The double-layer cylindrical protection zone model distinguishes horizontal and vertical separation constraints, the adaptive time segmentation mechanism adjusts trajectory representation according to UAV speed and local traffic density, and the cylindrical warning zone lower-bound pruning mechanism reduces invalid tree node searches by exploiting independent horizontal and vertical safety bounds. These mechanisms provide an interpretable and computationally efficient front-end conflict detection method for large-scale low-altitude UAV traffic management.

The comparison with representative uncertainty-aware and learning-based baseline variants further shows the trade-off characteristics of AC-DTPM. Although the learning-based baseline variant achieved shorter average online inference time, AC-DTPM achieved zero missed detection, lower candidate redundancy, a smaller candidate compression ratio, and lower memory cost in the tested scenario. Therefore, AC-DTPM is not necessarily the fastest method in absolute runtime, but it is useful when deterministic screening, explainability, detection completeness, and controlled memory usage are prioritized over inference speed alone.

This study is still limited by the use of simulated low-altitude traffic scenarios and by its focus on conflict detection rather than integrated conflict resolution. Future work will validate AC-DTPM using real low-altitude UAV traffic data, combine it with downstream conflict resolution methods, and explore distributed online implementation for practical UAV traffic management.

Author Contributions: Conceptualization, Z.C. and X.L.; methodology, Z.L. and Z.C.; software, Z.L.; validation, Z.L. and D.L.; formal analysis, Z.L.; investigation, Z.L.; data curation, D.L.; writing—original draft preparation, Z.L.; writing—review and editing, Z.L. and X.L.; visualization, Z.L.; supervision, X.L.; project administration, X.L.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Physical and dynamic parameters of the realistic multirotor UAV model.

Parameter Category	Symbol	Parameter Name	Current Value	Unit	Description
Mass	M	UAV Mass	1.0	kg	UAV mass in the model class
Gravity	g	Gravitational Acceleration	9.8	m/s^2	Used for thrust and dynamics calculations

Table A1. *Cont.*

Parameter Category	Symbol	Parameter Name	Current Value	Unit	Description
Arm Length	L	Equivalent Arm Length	0.2	m	Quadrotor geometric parameter
Maximum Thrust-to-Weight Ratio	TWR_{max}	Maximum TWR	3.5	—	Determines maximum total thrust
Maximum Total Thrust	T_{max}	Maximum Thrust	34.3	N	T_max = MgTWR_max
Inertia Matrix	**J**	Body Inertia	diag([2.32 × 10^{-3}, 2.32 × 10^{-3}, 4.00 × 10^{-3}])	kg·m^2	Attitude dynamics parameter
Collision Radius	R_{body}	Body Collision Radius	0.1	m	Used for physical collision envelope
Collision Height	H_{body}	Body Collision Height	0.1	m	Used for physical collision envelope
Linear Drag Coefficient	$c\{D, L\}$	Linear Aerodynamic Drag	[−0.2864, −0.2929, −0.3797]	—	Drag related to body frame velocity
Nonlinear Drag Coefficient	$c\{D, H\}$	Nonlinear Aerodynamic Drag	0.00075	—	Drag term related to horizontal velocity squared
Thrust Delay	τ_T	Thrust First-Order Delay	0.05	s	Thrust command response lag
Angular Velocity Delay	τ_ω	Angular Velocity First-Order Delay	0.05	s	Attitude angular velocity response lag
Maximum Horizontal Speed	$v_{xy,max}$	Maximum Horizontal Speed	25	m/s	Speed limit in the model class
Maximum Vertical Speed	$v_{z,max}$	Maximum Vertical Speed	10	m/s	Vertical speed limit in the model class
Max Roll/Pitch Angular Velocity	$\omega_{xy,max}$	Max Roll/Pitch Angular Velocity	10.0	rad/s	Angular velocity limit in the model class
Max Yaw Angular Velocity	$\omega_{z,max}$	Max Yaw Angular Velocity	0.3	rad/s	Approx. 17.19 deg/s
Velocity Response Time Constant	τ_{vel}	Outer-Loop Velocity Response Time	0.60	s	Velocity response parameter in ORCA velocity interface
Attitude Control Gain	k_{att}	Attitude Angular Velocity Proportional Gain	8.0	—	Generates angular velocity command from desired attitude
Explicit Horizontal Acceleration Limit	$a_{xy,max}$	ORCA Velocity Interface Horizontal Acceleration Limit	10.0	m/s^2	Prevents velocity command jumps
Explicit Upward Acceleration Limit	$a_{z,up,max}$	ORCA Velocity Interface Ascent Acceleration Limit	5.0	m/s^2	Vertical acceleration limit in velocity interface
Explicit Downward Acceleration Limit	$a_{z,down,max}$	ORCA Velocity Interface Descent Acceleration Limit	5.0	m/s^2	Vertical acceleration limit in velocity interface

Table A2. Simulation and reference trajectory parameters for realistic UAV baseline validation.

Parameter Category	Symbol	Parameter Name	Current Value	Unit	Description
Model Type	—	UAV Model Type	constrained multirotor UAV model	—	Low-altitude multirotor model constrained by velocity, acceleration, climb rate, yaw rate, control lag, positioning error, and wind disturbance
Simulation Experiment	—	Experiment Name	realistic UAV baseline verification	—	This experiment verifies whether the realistic UAV model satisfies dynamic constraints; it is not the final algorithm comparison experiment

Table A2. *Cont.*

Parameter Category	Symbol	Parameter Name	Current Value	Unit	Description
Trajectory Type	—	Reference Trajectory	figure-eight/pentagram	—	Default reference trajectory is figure-eight
Simulation Step	Δt	Integration Step	0.02	s	Used for realistic trajectory integration
Simulation Duration	T_{end}	Total Simulation Time	65	s	Duration of current single-UAV baseline verification
Random Seed	—	Random Seed	7	—	Used to ensure experiment reproducibility
Reference Speed Lower Bound	$v_{\min}$	Minimum Reference Speed	8	m/s	Lower bound of baseline speed for the realistic model
Reference Speed Upper Bound	$v_{\max}$	Maximum Reference Speed	15	m/s	Upper bound of baseline speed for the realistic model
Cruise Speed	v_{cruise}	Average Reference Cruise Speed	11.5	m/s	Derived as (v_min + v_max)/2
Speed Fluctuation Amplitude	v_{amp}	Reference Speed Sinusoidal Amplitude	3.15	m/s	Derived as 0.45(v_max − v_min)
Maximum Horizontal Speed	$v_{xy,\max}$	Maximum Horizontal Airspeed	16	m/s	Limits horizontal motion speed
Maximum 3D Speed	$v_{\max,\text{total}}$	Maximum 3D Airspeed	18	m/s	Limits overall flight speed
Maximum Horizontal Acceleration	$a_{xy,\max}$	Maximum Horizontal Acceleration	4.0	m/s^2	Limits horizontal acceleration capability
Maximum Horizontal Deceleration	$a_{xy,\text{dec},\max}$	Maximum Horizontal Deceleration	5.0	m/s^2	Limits horizontal deceleration capability
Maximum Climb Rate	$v_{z,\text{up},\max}$	Maximum Ascent Speed	3.0	m/s	Limits climb capability
Maximum Descent Rate	$v_{z,\text{down},\max}$	Maximum Descent Speed	2.5	m/s	Limits descent capability
Maximum Upward Vertical Acceleration	$a_{z,\text{up},\max}$	Maximum Upward Acceleration	2.0	m/s^2	Limits upward vertical velocity change
Maximum Downward Vertical Acceleration	$a_{z,\text{down},\max}$	Maximum Downward Acceleration	2.5	m/s^2	Limits downward vertical velocity change
Maximum Yaw Rate	$\omega_{\psi,\max}$	Maximum Yaw Angular Speed	35	deg/s	Limits UAV heading change rate
Velocity Command Lag	τ_v	Velocity Command First-Order Lag Time Constant	0.38	s	Represents lag between velocity command and actual response
Yaw Response Time Constant	τ_ψ	Yaw Response Time Constant	0.30	s	Represents heading response lag
Position Feedback Gain	$\mathbf{K}_p$	Position Feedback Control Gain	diag([0.34, 0.34, 0.45])	—	Used to generate desired velocity correction from position error
Horizontal Positioning Error Std. Dev.	σ_{xy}	Horizontal Position Measurement Noise	0.30	m	Simulates positioning error
Vertical Positioning Error Std. Dev.	σ_z	Vertical Position Measurement Noise	0.20	m	Simulates altitude measurement error
Horizontal Velocity Measurement Error Std. Dev.	$\sigma_{v,xy}$	Horizontal Velocity Measurement Noise	0.06	m/s	Records velocity measurement error
Vertical Velocity Measurement Error Std. Dev.	$\sigma_{v,z}$	Vertical Velocity Measurement Noise	0.04	m/s	Records vertical velocity measurement error
Positioning Bias Time Constant	τ_b	Gauss-Markov Position Bias Time Constant	18.0	s	Used to generate smooth positioning bias

Table A2. *Cont.*

Parameter Category	Symbol	Parameter Name	Current Value	Unit	Description
Positioning Bias Std. Dev.	σ_b	Position Bias Standard Deviation	[0.15, 0.15, 0.10]	m	Corresponds to x, y, and z directions, respectively
Steady Wind Field	$\mathbf{w}_{\text{steady}}$	Steady Wind Speed	[0.45, −0.25, 0.03]	m/s	Simulates stable low-altitude wind disturbance
Gust Amplitude	$\mathbf{w}_{\text{amp}}$	Sinusoidal Gust Amplitude	[0.25, 0.20, 0.05]	m/s	Simulates low-frequency smooth gusts
Gust Frequency	$\mathbf{f}_{\text{gust}}$	Sinusoidal Gust Frequency	[0.06, 0.09, 0.12]	Hz	Corresponds to x, y, and z directions, respectively
Random Gust Time Constant	τ_{gust}	Gauss–Markov Gust Time Constant	6.0	s	Used to generate smooth random gusts
Random Gust Std. Dev.	σ_{gust}	Random Gust Standard Deviation	[0.10, 0.10, 0.03]	m/s	Corresponds to x, y, and z directions, respectively
Figure-Eight Horizontal Scale (x)	A	Amplitude in x Direction	155	m	Current figure-eight trajectory parameter
Figure-Eight Horizontal Scale (y)	B	Amplitude in y Direction	95	m	Current figure-eight trajectory parameter
Figure-Eight Vertical Amplitude	C	Height Fluctuation Amplitude in z Direction	10	m	Height variation amplitude of current figure-eight trajectory
Figure-Eight Center Altitude	z_0	Reference Trajectory Center Altitude	70	m	Primary flight altitude of current figure-eight trajectory
True Output Airspeed	v_{air}	Airspeed	Recorded by simulation	m/s	Used to compute true acceleration and verify motion constraints
Ground Speed	v_{ground}	Ground Speed	v_air + w	m/s	Includes wind disturbance effect
Trajectory Tracking Error	e_{track}	Reference vs. True Trajectory Error	Recorded by simulation	m	Used to verify whether the realistic model can track stably
Position Measurement Error	e_{meas}	Measured vs. True Position Error	Recorded by simulation	m	Used for subsequent safety threshold calibration

References

1. Lyu, Z.; Gao, Y.; Chen, J.; Du, H.; Xu, J.; Huang, K.; Kim, D.I. Empowering Intelligent Low-Altitude Economy with Large AI Model Deployment. *IEEE Wirel. Commun.* **2026**, *33*, 64–72. [CrossRef]
2. Hamissi, A.; Dhraief, A.; Sliman, L. A Comprehensive Survey on Conflict Detection and Resolution in Unmanned Aircraft System Traffic Management. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 1395–1418. [CrossRef]
3. Chen, Y.; Xu, Y.; Yang, L.; Hu, M. In-Flight Fast Conflict-Free Trajectory Re-Planning Considering UAV Position Uncertainty and Energy Consumption. *Transp. Res. Part C Emerg. Technol.* **2025**, *171*, 104988. [CrossRef]
4. Monteiro, L.B.; Ribeiro, V.F.; Garcia, C.P.; Rocha Filho, G.P.; Weigang, L. 4D Trajectory Conflict Detection and Resolution Using Decision Tree Pruning Method. *IEEE Lat. Am. Trans.* **2023**, *21*, 277–287. [CrossRef]
5. Li, Y.; Li, J.; Wang, J.; Zhang, X.; Ding, H.; Du, W. Multiscale-Graph-Enhanced Reinforcement Learning for Conflict Resolution in Dense UAV Networks. *IEEE Internet Things J.* **2025**, *12*, 44290–44303. [CrossRef]
6. Isufaj, R.; Omeri, M.; Piera, M.A. Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 610. [CrossRef]
7. Zhao, B.; Huo, M.; Li, Z.; Yu, Z.; Qi, N. Graph-Based Multi-Agent Reinforcement Learning for Large-Scale UAVs Swarm System Control. *Aerosp. Sci. Technol.* **2024**, *150*, 109166. [CrossRef]
8. Prandini, M.; Hu, J.; Lygeros, J.; Sastry, S. A Probabilistic Approach to Aircraft Conflict Detection. *IEEE Trans. Intell. Transp. Syst.* **2000**, *1*, 199–220. [CrossRef]
9. Lindqvist, B.; Mansouri, S.S.; Agha-mohammadi, A.A.; Nikolakopoulos, G. Nonlinear MPC for Collision Avoidance and Control of UAVs with Dynamic Obstacles. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6001–6008. [CrossRef]
10. Acevedo, J.J.; Castaño, Á.R.; Andrade-Pineda, J.L.; Ollero, A. A 4D Grid-Based Approach for Efficient Conflict Detection in Large-Scale Multi-UAV Scenarios. In Proceedings of the 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS), Cranfield, UK, 25–27 November 2019; pp. 18–23.
11. Wubben, J.; Calafate, C.T.; Cano, J.C.; Manzoni, P. FFP: A Force Field Protocol for the Tactical Management of UAV Conflicts. *Ad Hoc Netw.* **2023**, *140*, 103078. [CrossRef]
12. Zhou, Z.; Zhao, G.; Jiang, Y.; Wu, Y.; Yang, J.; Meng, L. Research on UAV Conflict Detection and Resolution Based on Tensor Operation and Improved Differential Evolution Algorithm. *Aerospace* **2024**, *11*, 1008. [CrossRef]

13. Zhang, Y.H.; Yi, P.; Hong, Y.G. Cooperative Safe Trajectory Planning for Quadrotor Swarms. *Sensors* **2024**, *24*, 707. [CrossRef] [PubMed]
14. Guo, Y.; Liu, X.; Jiang, W.; Zhang, W. Collision-Free 4D Dynamic Path Planning for Multiple UAVs Based on Dynamic Priority RRT* and Artificial Potential Field. *Drones* **2023**, *7*, 180. [CrossRef]
15. Candan, F.; Beke, A.; Mahfouf, M.; Mihaylova, L. A Real-Time Fuzzy Interacting Multiple-Model Velocity Obstacle Avoidance Approach for Unmanned Aerial Vehicles. *J. Intell. Robot. Syst.* **2024**, *110*, 61. [CrossRef]
16. Xian, B.; Jiang, P.; Hu, J.; Wang, Y.; Li, Q. A Three-Dimensional Collision Avoidance Strategy for Multiple Unmanned Aerial Vehicles Using Velocity Obstacle Method. *Aerosp. Sci. Technol.* **2026**, *168*, 110793. [CrossRef]
17. Li, Y.; Li, J.; Yu, C.; Du, W. A Hierarchical Conflict Resolution Framework with Graph Transformer-Based Reinforcement Learning for Heterogeneous UAV Networks. *IEEE Internet Things J.* **2026**, *13*, 5640–5654. [CrossRef]
18. Zhong, Q.; Yu, Y.; Zhong, L.; Zhang, Y.; Liu, S.; Pan, W.; Yan, X. Optimizing Safety Buffer Spacing and 3D Dynamic Obstacle Avoidance Strategies for Urban Low-Altitude UAV Swarm Operations. *Chin. J. Aeronaut.* **2025**, 103901. [CrossRef]
19. Qi, Y.; Jiang, H.; Huang, G.; Yang, L.; Wang, F.; Xu, Y. Multi-UAV Path Planning Considering Multiple Energy Consumptions via an Improved Bee Foraging Learning Particle Swarm Optimization Algorithm. *Sci. Rep.* **2025**, *15*, 14755. [CrossRef] [PubMed]
20. Vila Carbó, J.A.; Balbastre Tejedor, J.V.; Morcillo-Pallarés, P.; Yuste Pérez, P. Risk-Based Method for Determining Separation Minima in Unmanned Aircraft Systems. *J. Air Transp.* **2023**, *31*, 57–67. [CrossRef]
21. Xie, H.; Su, F.; Yin, J.; Han, S.; Zhang, X. Network Modeling and Refined Management of UAV Flight Conflicts in Complex Low-Altitude Airspace. *Acta Aeronaut. Astronaut. Sin.* **2023**, *44*, 328226.
22. Liu, Z.; Nan, Y.; Xie, R. Airspace Conflict Detection Method Based on Divided Grid. *Nav. Sci. Technol.* **2022**, *42*, 157–162.
23. Qu, K.; Zhao, G.; Zhang, Y.; Wu, Y.; Wei, X.; Ouyang, W. Airspace Conflict Detection Method Based on Matrix Hadamard Product Operation. *J. Air Force Eng. Univ.* **2023**, *24*, 55–63.
24. Bi, K.; Wu, M.; Wen, X.; Zhang, W.; Yang, W. Modeling and Analysis of Flight Conflict Network Based on Velocity Barrier Method. *Syst. Eng. Electron. Technol.* **2021**, *43*, 2163–2173.
25. Xu, Y. Hadamard Product of Tensors and Its Applications. Bachelor's Thesis, Yunnan University, Kunming, China, 2016.
26. Rosen, K.H. *Elementary Number Theory and Its Applications*; Xia, H., Translator; China Machine Press: Beijing, China, 2015.
27. Yuan, Y.; Dong, Z. Differential Evolution Algorithm Based on Adaptive Multi-mutation and Its Application. *Sci. Technol. Innov.* **2024**, *12*, 104–107.
28. Li, Y. Research on Optimization Method of Differential Evolution and Its Application in Flight Conflict Resolution. Ph.D. Thesis, Sichuan University, Chengdu, China, 2023.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.