Robotica (2025), **43**, pp. 1640–1659
doi:10.1017/S0263574725000554

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Motion planning for agile fixed-wing UAVs in complex low-altitude environments

Fei Zou, Jie Li and Yifeng Niu

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China
Corresponding author: Yifeng Niu; Email: niuyifeng@nudt.edu.cn

Received: 20 September 2024; **Revised:** 2 March 2025; **Accepted:** 24 March 2025; **First published online:** 19 May 2025

Keywords: motion planning; agile fixed-wing UAVs; complex low-altitude environments; path planning; trajectory planning

Abstract

The autonomous safe flight of fixed-wing unmanned aerial vehicles (UAVs in complex low-altitude environments presents significant challenges and holds practical application value. This paper proposes a motion planning method for agile fixed-wing UAVs to address safety issues in navigating narrow corridors within such environments. In the path planning phase, we introduce the Improved Batch Informed Trees (IBIT*) to enhance both the solving speed and quality of BIT*. The IBIT* incorporates strategies such as using Rapidly Exploring Random Tree (RRT)-Connect for initial pathfinding, informed sparse sampling, and re-selecting parent nodes. During the trajectory planning phase, we first decouple the roll angle of the UAV from its three-dimensional position based on the agility of fixed-wing UAVs; subsequently, we address constraints related to smoothness and mission time by leveraging the characteristics of the Minimum Control Effort; finally, we design a differentiable penalty function to satisfy the dynamic performance constraints of the UAV. The effectiveness and superiority of the proposed motion planning method are demonstrated through numerical simulations and physical flight experiments.

1. Introduction

With advancements in control science, aeronautical engineering, and other disciplines, unmanned aerial vehicles (UAVs) have rapidly developed and are widely used across various fields, including film and television aerial photography, agricultural and forestry applications, geological exploration, and logistics and transportation. In recent years, the operation of UAVs in complex low-altitude environments, such as urban areas and mountainous regions, has garnered significant attention [1, 2]. These environments pose unique challenges due to the low flight altitudes, which limit the ability to obtain clear mapping information from satellite remote sensing. Additionally, the irregular distribution of obstacles, the varying heights of these obstacles, and the fluctuating widths of flight corridor pose significant challenges for UAV safety.

Agile fixed-wing UAVs are a type of UAV with aerobatic capabilities, characterized by a high thrust-to-weight ratio, strong rudder effect, and low aspect ratio [3]. Compared to rotary-wing UAVs, agile fixed-wing UAVs offer faster flight speeds and superior endurance. In contrast to traditional fixed-wing UAVs, agile fixed-wing UAVs have enhanced maneuvering capabilities, including large-angle hovering, very low-speed flight, and significant lateral maneuvers [4–6]. Hence, it is feasible to consider leveraging the superior lateral flight capabilities of agile fixed-wing UAVs to enhance the safety of UAV missions performed in complex low-altitude environments.

Motion planning is a key technology for enabling autonomous flight in UAVs. It involves determining an optimal or near-optimal path or trajectory that considers the constraints of the UAV, the mission, and the environment to achieve a specific objective [7]. Depending on the planning space, motion planning can be classified into path planning, which operates in discrete state space; trajectory planning,

© College of Intelligence Science and Technology, National University of Defense Technology, China, 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

which operates in continuous curve parameter space; and hybrid planning, which involves planning in the space of trajectories or action primitives [8–10].

Path planning refers to the process of planning a path in a space that contains only spatial information [11]. Algorithms of path planning include search-based, sampling-based, artificial potential field-based, intelligent optimization-based, and reinforcement learning-based algorithms. Search-based algorithms can guarantee the optimality and completeness of the solution but may encounter the curse of dimensionality. Artificial potential field-based algorithms are prone to getting trapped in local optima, while intelligent optimization-based algorithms often exhibit slow convergence rates. Most reinforcement learning-based algorithms are still in the simulation phase. Sampling-based path planning algorithms perform path searches by randomly selecting sample points in the state space and connecting these points according to specific rules to form a graph or tree structure. One of the most representative algorithms is the Rapidly exploring Random Tree (RRT), which is probabilistically complete but not necessarily optimal [12]. Many algorithms have been developed based on RRT. RRT* incorporates the process of re-selecting parent nodes and rewiring the tree, making it asymptotically optimal [13, 14]. The Informed RRT* restricts the sampling points to an ellipse centered around the start and end points, thereby enhancing sampling efficiency [15]. The Batch Informed Trees (BIT*) combines the advantages of the sampling strategies from the Fast Marching Trees and the Informed RRT* by batching points within the ellipse and introduces heuristic approaches to evaluate path costs, significantly improving efficiency [16].

Trajectory planning encompasses not only time information but also takes into account the kinematic and dynamic constraints of the UAV [17, 18]. It typically utilizes initial points provided by path planning to enhance the success rate and speed of solutions. Trajectory representations include segmented polynomial, Dubins, B-spline, and Minimum Control Effort (MINCO) [19–22]. Obstacles are usually modeled as inequality constraints (hard constraints) or cost functions (soft constraints) in trajectory planning. Hard constraint methods can ensure safety but are often difficult and slow to solve, whereas soft constraint methods are typically numerically stable and easier to resolve [23]. Tal derived the global differential flatness properties of a VTOL fixed-wing UAV, generating polynomial trajectories in flat space to achieve agile maneuvering flights, including lateral uncoordinated flight and inverted flight [24]. Bry proposed a Dubins-Polynomial trajectory based on differential flatness properties to enable high-speed flight of fixed-wing UAVs in underground garages [23]. Based on B-spline, Gao proposed an EGO planner for real-time planning of autonomous flights in environments with unknown obstacles [25]. Wang proposed the minimum control effort trajectory that can simultaneously optimize time and space, achieving swarm flights of rotary-wing UAVs in dense bamboo forest environments [22, 26].

The core concept of hybrid planning is to combine trajectories or motion segments that satisfy dynamic characteristics to form a longer continuous trajectory or motion sequence [27]. For autonomous flight missions in complex low-altitude environments, there are numerous constraints involved, and the aircraft may need to perform maneuvering flight actions. The solutions for trajectory libraries or motion primitives of agile fixed-wing UAVs differ significantly from those of conventional fixed-wing UAVs. Moreover, hybrid planning need to solve additional optimization problems, which often results in poor real-time performance. Thus, the hybrid planning may not fulfill our requirements.

In practice, the motion planning framework that combines front-end path planning and back-end trajectory planning has been widely applied. The challenge in front-end path planning is to quickly obtain high-quality solutions. Despite the advanced sampling-based path planning algorithm BIT* demonstrating good performance, issues such as its slow initial solution computation speed still limit its application in complex low-altitude environments. The challenge in back-end trajectory planning is to derive a flyable trajectory for the UAV in real time, considering mission time, UAV dynamic performance, and flight safety. Currently, most fixed-wing UAV trajectory planning validated through actual flight experiments primarily focuses on the UAV's position without considering its attitude, which limits the adaptability of the UAV in narrow corridors within complex low-altitude environments. Additionally, some trajectory planning methods are based on a coordinated turning model of fixed-wing UAVs to derive differential flatness [28]. In summary, motion planning for fixed-wing UAVs in complex low-altitude environments remains challenging.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

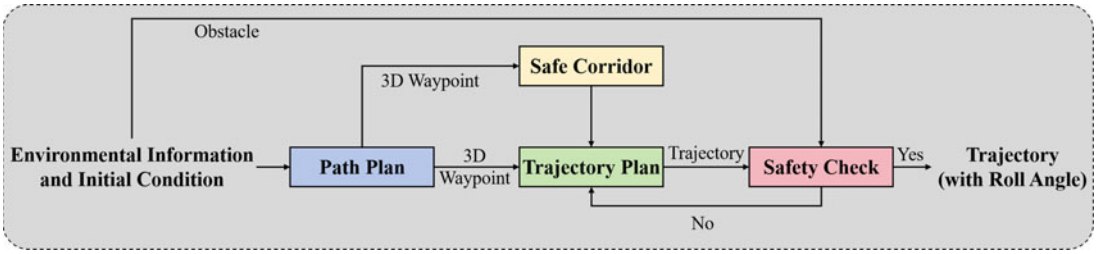


Figure 1. *Motion planning framework.*

This paper proposes a motion planning method for small agile fixed-wing UAVs operating in complex low-altitude environments. At the path planning layer, to address the issues of slow solution speed and low solution quality of the BIT algorithm that fail to meet environmental requirements, we introduce an Improved Batch Informed Trees (IBIT*) algorithm, which enhances computational speed while improving the quality of the planned paths. In the trajectory planning layer, we first decouple the roll angle of the UAV from its 3-D position based on the lateral maneuverability of agile fixed-wing UAVs, thereby enhancing the UAV's ability to navigate through narrow corridors. Subsequently, we utilize the MINCO approach to represent the trajectory and design a differentiable penalty function to account for the constraints imposed by the UAV's dynamic performance. Experimental results demonstrate that the proposed motion planning method can rapidly generate reasonable flight trajectories with roll angles.

The remainder of this paper is organized as follows. Section 2 introduces the designed motion planning framework. Section 3 presents the IBIT* algorithm. Section 4 discusses the trajectory planning problem for agile fixed-wing UAVs. Section 5 describes the flight experiment. Finally, Section 6 provides a summary of the entire paper.

2. Agile fixed-wing UAV motion planning framework

The complex low-altitude environment is subject to numerous constraints, including time, safety, smoothness, and the dynamic performance of the UAV. Among these, safety constraints differ significantly from those in ordinary environments. In certain areas of the complex low-altitude environment, UAVs may need to fly laterally at a specific roll angle to pass through safely. Given the robust lateral flight capability of agile fixed-wing UAVs, we can treat the roll angle as a degree of freedom specifically dedicated to obstacle avoidance. This implies that the outcome of the motion planning process will be a trajectory with an associated roll angle [29].

The agile fixed-wing UAV motion planning framework designed for complex low-altitude environments is illustrated in Figure 1. This framework comprises four steps: path planning, Safety Flight Corridor (SFC) generation, trajectory planning, and safety detection, with a primary focus on path planning and trajectory planning in this paper. The SFC is generated based on the method proposed in [35]. The motion planning framework first conducts path planning to compute a set of discrete 3-D waypoints; then, to enhance the real-time performance of trajectory planning, the SFC is constructed based on the results of path planning to narrow down the planning space. Subsequently, in trajectory planning, constraints such as safety, smoothness, and dynamic performance are considered to generate a trajectory online that meets these constraints. Finally, a safety loop closure check is conducted on the trajectory to further ensure its safety.

3. Improved Batch Informed Trees based path planning

3.1. The drawbacks of the BIT in complex low-altitude environments*

In complex low-altitude environments, autonomous UAVs require real-time path planning based on local environmental data, demanding planners to quickly generate high-quality paths. Optimal solutions

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

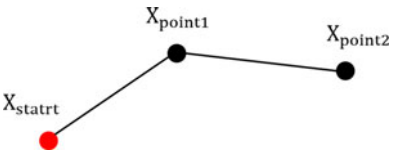


Figure 2. *Polyline segment.*

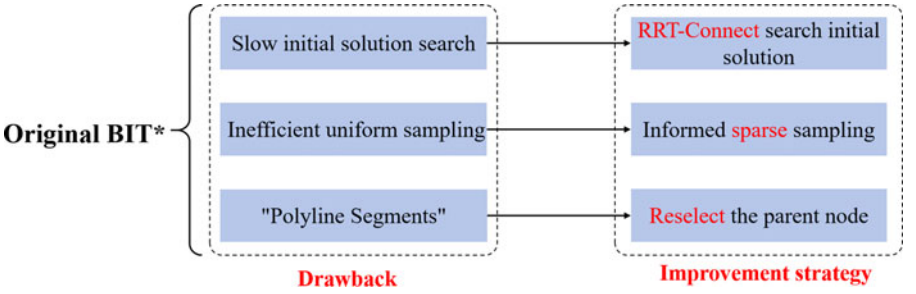


Figure 3. *Three improvement strategies for the BIT*.*

aren't essential for two reasons: (1) Local planning inherently lacks global environmental awareness, making true optimization unattainable and (2) sampling-based planning methods typically converge to near-optimal solutions within reasonable iterations, rendering excessive time spent on perfect paths unjustified.

BIT* algorithm integrates FMT* and Informed RRT* strengths with heuristics for efficiency, yet faces three key limitations in complex low-altitude environments:

1) **Inefficient uniform sampling**: BIT* relies on uniform sampling before and after finding initial solutions (state-space-wide then ellipsoid-restricted). This approach generates redundant clustered points that increase computation without improving solutions.

2) **Slow initial solution discovery**: By retaining only quality-enhancing samples, BIT* experiences delayed initial path finding. This prolongs uniform sampling cycles, severely degrading performance.

3) **Suboptimal pruning mechanism**: While pruning non-contributing vertices improves solution quality, BIT* delays pruning until both vertex/edge queues empty. The absence of parent node reselection creates suboptimal "Polyline Segments" (Figure 2), further compromising solution quality.

3.2. Improved Batch Informed Trees

To address the three issues present in BIT*, the Improved Batch Informed Trees (IBIT*) is proposed, with its improvement strategies illustrated in Figure 3.

To resolve slow initial solution searches, we use RRT-Connect for rapid solution discovery [30]. This single-query algorithm builds two trees from the start and goal, alternately extending toward each other to achieve accelerated exploration. While RRT-Connect does not ensure optimality, we leverage its speed advantage exclusively. Being a classic algorithm, its implementation details are omitted here.

To improve uniform sampling inefficiency, we implement an informed sparse sampling strategy (detailed in Algorithm 1). Its core step (SampleSpare(x_{rand}, r_{sp}, nf)) inputs sparse radius r_{sp}, current points x_{rand}, and allowable failures nf, outputting a new point. SampleSpare(x_{rand}, r_{sp}, nf) first samples within the informed ellipsoid, then checks distances to x_{rand}. If all exceed r_{sp}, it outputs the point; otherwise, it resamples. After nf consecutive failures, it returns the last valid sample. Parameter nf prevents excessive iterations to save resources. Note: Line 9 in Algorithm 1 handles uniform sampling, while line 11 performs collision detection.

To tackle inadequate pruning, we propose a parent node reselection method, detailed in Algorithm 2. This process is triggered whenever a new node and edge are added to the search tree. The decision to

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

Algorithm 1 imSample (m,x_{start},X_{goal},c_i) Informed sparse sampling

```
is_sp = True
if r_sp < 0.1 then
    is_sp = False
end if
repeat
    if i ≤ m/2 and m ≥ 20 and is_sp then
        x_temp = SampleSpare(x_rand, r_sp, nf)
    else
        x_temp = SampleEllipsoid(c_i)
    end if
    collision_detection(x_rand, obstacle)
    x_rand = x_rand_update(x_temp)
until length(x_rand) = m
```

Algorithm 2 rc_parent($\mathbf{v}_{min}$,$\mathbf{x}_{min}$) Reselect the parent node

```
if ! (if_collision(v_min_parent, x_min)) then
    v_temp ← ∅
    v_temp ←+ {v_min}, v_temp ←+ {v_min_parent}
    distance _ list = distance(v_temp, x_min)
    mindis _ index = min _ distance(distance _ list)
    new _ par = v_temp(mindis _ index)
    c_i ← distance _ list(mindis _ index)
    E ←+ {(new _ par, x_min)}
    Parent(x_min) ← new _ par
end if
```

reassign the parent of node $\mathbf{v}_{\min}$ as the parent of node $\mathbf{x}_{\min}$ hinges on safety (evaluating obstacles between $\mathbf{v}_{\min}$ and $\mathbf{x}_{\min}$) and the potential to reduce cost from $\mathbf{x}_{\min}$ back to the origin.

The pseudocode for IBIT* is shown in Algorithm 3. The vertex expansion step ExpandNextVertex ($\mathcal{Q}_V$, $\mathcal{Q}_E$, c_i) and graph pruning step Prune ($\mathcal{T}, X_{\text{unconn}}, c_i$) are the same as in BIT*, so they won't be discussed here. The core steps of IBIT* include:

1) **Initialization**: Initialize start state x_{start}, goal state x_{goal}, tree $T := (V, E)$, disconnected state set X_{unconn}, vertex expansion queue Q_V, and edge evaluation queue Q_E. Also define goal region vertices $V_{sol'n}$ and unexpanded vertex variables $V_{unexpnd}$.

2) **Batch Sampling**: During new batch initialization, the queue resets. Samples and trees are pruned to remove states incapable of enhancing solutions. New samples are then collected from subproblems with superior solutions (Sample (m, $\mathbf{x}_{\text{start}}$, X_{goal}, c_i)).

3) **Edge Selection**: The tree grows (ExpandNextVertex ($\mathcal{Q}_V$, $\mathcal{Q}_E$, c_i)) by processing edges sorted by ascending estimated costs. When multiple edges share identical estimates, priority is given to those with minimal current cost to the source vertex (PopBest InQueue ($\mathcal{Q}_E$)).

4) **Edge Processing**: Heuristics accelerate edge processing and delay actual cost computations. Functions $\widehat{g}(x)$ and $\widehat{h}(x)$ estimate forward costs (start to $x \in X$) and backward costs (to goal), respectively. Parent node rc_parent($\mathrm{v}_{\min}$,$\mathrm{x}_{\min}$) reselection is embedded within edge-processing function EdgeProcess().

5) **Vertex Expansion**: Remove a vertex from the vertex queue and add the edges originating from that vertex to the edge queue.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

Algorithm 3 Improved Batch Informed Trees IBIT*

$V, E, c_i \leftarrow RRT - Connect(X_{\text{start}}, X_{\text{goal}})$
$\mathcal{T} = (V, E)$; $X_{\text{unconn}} \leftarrow X_{\text{goal}}$; $\mathcal{Q}_V \leftarrow V$; $\mathcal{Q}_E \leftarrow \emptyset$
$V_{\text{sol'n}} \leftarrow V \cap X_{\text{goal}}$; $V_{\text{unexpnd}} \leftarrow V$; $X_{\text{new}} \leftarrow X_{\text{unconn}}$
repeat
  if $\mathcal{Q}_E \equiv \emptyset$ and $\mathcal{Q}_V \equiv \emptyset$ **then**
    $X_{\text{reuse}} \leftarrow \text{Prune}\ (\mathcal{T}, X_{\text{unconn}}, c_i)$
    $X_{\text{sampling}} \leftarrow \text{imSample}\ \left(m, \mathbf{x}_{\text{start}}, X_{\text{goal}}, c_i\right)$
    $X_{\text{new}} \leftarrow X_{\text{reuse}} \cup X_{\text{sampling}}$; $X_{\text{unconn}} \leftarrow X_{\text{new}}$; $\mathcal{Q}_V \leftarrow V$
  end if
  while BestQueueValue($\mathcal{Q}_V$) $\leq$ BestQueueValue($\mathcal{Q}_E$) **do**
    ExpandNextVertex($\mathcal{Q}_V$, $\mathcal{Q}_E$, c_i)
  end while
  $(\mathbf{v}_{\min}, \mathbf{x}_{\min}) \leftarrow \text{PopBestInQueue}(\mathcal{Q}_E)$
  EdgeProcess()
until STOP
return $\mathcal{T}$

6) **Graph Pruning**: Remove states that cannot provide a better solution than the given cost.

3.3. Performance analysis of IBIT algorithm*

Completeness and optimality are crucial for path planning algorithms. The proposed IBIT* algorithm's completeness and optimality are analyzed below.

Theorem 1. (Probabilistic completeness of IBIT*). *When the number of samples approaches infinity, if a feasible path exists, the probability that IBIT* finds it is 1,*

$$\lim_{q\to\infty} \inf P\left(\sigma_{q,\ \text{IBIT}^*} \in \Sigma, \sigma_{q,\ \text{IBIT}^*}(0) = \mathbf{x}_{\text{start}}, \sigma_{q,\text{IBIT}^*}(1) \in X_{\text{goal}}\right) = 1 \tag{1}$$

where q is the number of samples,$\boldsymbol{\sigma}_{q,\text{IBIT}^*}$ is the path found by IBIT* from q samples, and Σ is the set ofall feasible, collision-free paths.

Proof. Proof of Theorem 1 follows from the proof of asymptotic optimality (Theorem 2). □

Theorem 2. (Asymptotic optimality of IBIT*). If an optimal solution exists for a given path planning problem, the probability that IBIT* asymptotically converges to the optimal solution is 1 when given infinite samples,

$$P\left(\lim_{q\to\infty} \sup\ c\left(\sigma_{q,\text{BIT}^*}\right) = c(\sigma^*)\right) = 1 \tag{2}$$

where q is the number of samples,$\boldsymbol{\sigma}_{q,\text{IBIT}^*}$ is the path found by IBIT* from q samples, and σ^* is optimal solution to the planning problem.

Proof. Theorem 2 is proven by showing that IBIT* considers at least the same edges as RRT* for a sequence of states, $X_{\text{samples}} = \left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\right)$, and connection limit, $r_{\text{IBIT}^*} \geqslant r_{\text{RRT}^*}$. □

RRT* incrementally builds a tree from a sequence of samples. For each state in the sequence, $\mathbf{x}_k \in X_{\text{samples}}$, it considers the neighborhood of earlier states that are within the connection limit. It selects the connection from this neighborhood that minimizes the cost-to-come of the state and then evaluates the

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

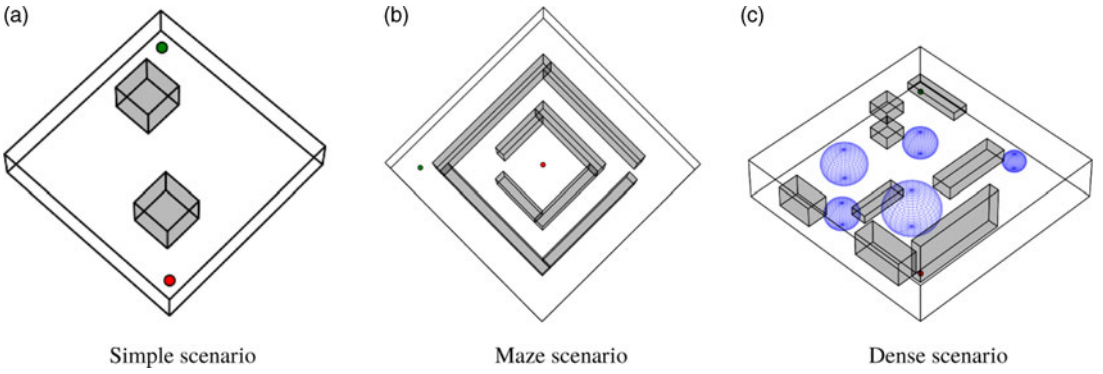


Figure 4. *Three typical 3D scenarios.*

ability of connections from this state to reduce the cost-to-come of the other states in the neighborhood.

$$X_{\text{near},k} := \left\{ \mathbf{x}_j \in X_{\text{samples}} \mid j<k, \left\| \mathbf{x}_k - \mathbf{x}_j \right\|_2 \leq r_{\text{RRT}^*} \right\} \tag{3}$$

IBIT* organizes the input state sequences into batches ($X_{\text{samples}} = (Y_1, Y_2, \ldots, Y_\ell)$), each containing $m < q$ samples (e.g., $Y_1 := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$). It progressively constructs a tree by processing these batches sequentially. For every state $\mathbf{y} \in Y_k$ in the sequence, the algorithm evaluates neighboring states from the same or prior batches within the predefined connection limit.

$$X_{\text{near},k} := \left\{ \mathbf{x} \in Y_j \mid jk, \|\mathbf{y} - \mathbf{x}\|_2 r_{\text{BIT}^*} \right\} \tag{4}$$

It adds the edge in this neighborhood to the tree that minimizes the cost-to-come of the state and considers all outgoing edges to connect its neighbors. Given $m \geqslant 1$, this set of edges includes all those considered by RRT* for an equivalent connection limit $r_{\text{BIT}^*} \geqslant r_{\text{RRT}^*}$.

Since the connection radius used for a batch is the same as that used by RRT* for the first sample in the batch, and both connection radius are monotonically decreasing, this indicates that IBIT* considers at least the same edges as RRT* [16]. From these edges, IBIT* selects those that can improve the cost-to-come of the target state and currently provide a better solution, as stated in [13]. Therefore, as stated in Theorem 2, it is almost surely asymptotically optimal.

3.4. Comparison and verification of the IBIT algorithm*

To evaluate the performance of the IBIT* algorithm, three typical 3-D scenarios were designed, as illustrated in Figure 4.

The green and red points represent the start and end points, respectively, while the gray rectangular prisms, and blue spheres represent the obstacles. All scenarios share the dimensions of $20 \times 20 \times 5$. In the simple scenario, there are only two primary obstacles, with the start and end points located at (1.5, 1.5, 1) and (18.5, 18.5, 4), respectively. The maze scenario replicates a classic labyrinth, with the start and end points positioned at (2, 2, 1) and (10, 10, 4) [31]. The dense scenario is filled with numerous closely packed obstacles and maintains the same start and end points as the simple scenario.

The first comparison validates the search time for the initial solution. Tests were conducted on a computer with an Intel(R) Xeon(R) Gold 6128 CPU @ 3.40 GHz and an NVIDIA Quadro P4000 GPU to measure the time taken for the BIT* and IBIT* algorithms to find initial solutions in typical scenarios. The initial solution search times for both algorithms are presented in Table I, with the data indicating that the IBIT* algorithm has a faster search speed for the initial solution compared to the BIT* algorithm, demonstrating a significant improvement.

The IBIT* was compared with Improved Bi-P-RRT* [32], Informed-RRT*, and BIT*. These algorithms were tested repeatedly in three scenarios, with key parameter settings listed in Table II.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

Table I. *Comparison of initial solution search time for paths in typical scenarios.*

Algorithm\Scenario	Simple scenario	Maze scenario	Dense scenario
BIT*	49.3*ms*	193*ms*	191*ms*
IBIT*	9.66*ms*	73.5*ms*	4.20*ms*

Table II. *Important parameters of the four sampling path planning algorithms.*

Algorithm	Parameter
Improved Bi-P-RRT*	$r = 1, k_r = 0.8, k_a = 2.4$
Informed-RRT*	stepsize = 1, rgoal = 1, eta = 1
BIT*	eta = 5, m = 20
IBIT*	$\text{stepsize}_{rrt-connect}$ = 1, m = 20, $nf = 5$
	eta = 5, $r_{sp} = \sqrt[3]{\frac{V_{ellipsoid}}{8m}} \cdot 0.9^{sample_times}$

Figure 5 illustrates the average path lengths and execution times of the algorithms. The results demonstrate that the IBIT* algorithm significantly outperforms the other three algorithms in both initial solution search speed and solution convergence speed. In all three scenarios shown in Figure 5, IBIT* consistently achieves the earliest initial solution (indicated by the green curve starting farthest to the left on the time axis), validating the effectiveness of our proposed strategy for accelerating initial solution search. This strategy employs RRT-Connect, which exhibits rapid exploration capabilities.

Furthermore, Figure 5 shows that IBIT* achieves faster convergence (steep green curve decline) and shorter paths (lowest green curve). These improvements originate from: 1) Informed sparse sampling: Eliminating redundant sampling to enhance efficiency. 2) Parent node reselection: Pruning suboptimal branches, smoothing paths, and optimizing length.

In non-maze scenarios, IBIT* finds high-quality solutions within 20*ms*. Even in maze scenarios, it achieves solutions within 0.5*s*. Since maze-like scenarios rarely occur in practice, IBIT* proves ideal for online path planning in complex low-altitude environments.

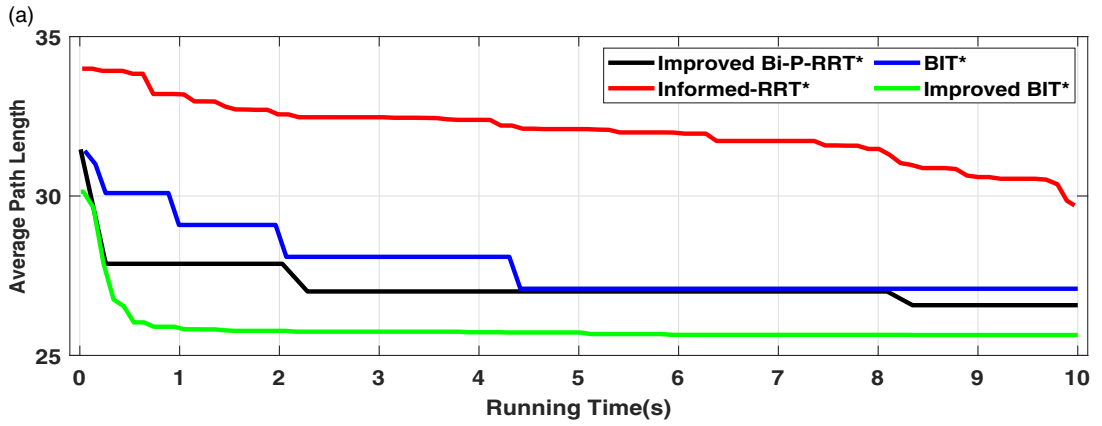
4. Agile fixed-wing UAVs trajectory planning

4.1. Trajectory planning for multiple constraint conditions

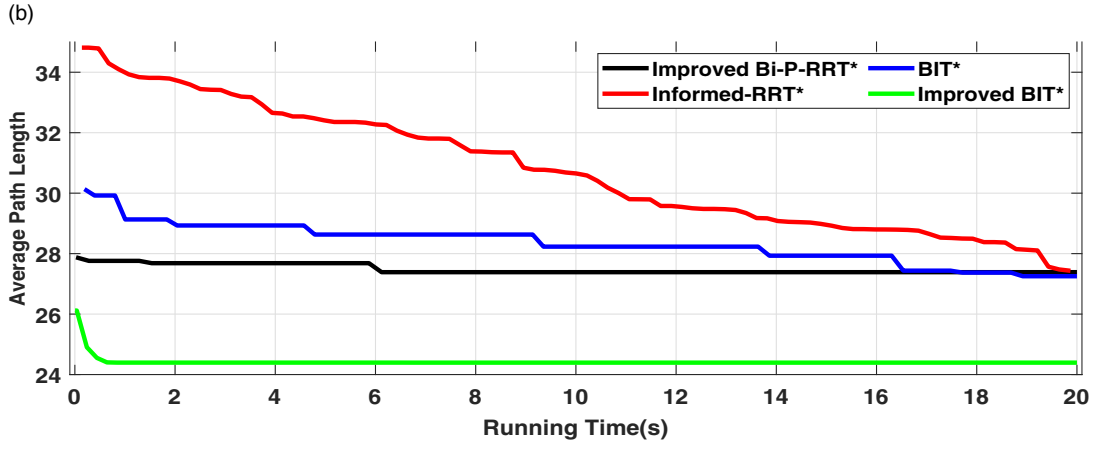
In complex low-altitude environments, trajectory planning for agile fixed-wing UAVs faces numerous constraints. First, the presence of constricted flight corridors requires that aircraft adopt specific roll attitudes to navigate through them. Second, mission execution is subject to temporal constraints. Additionally, the kinematic properties of fixed-wing UAVs necessitate smooth trajectory generation. Lastly, the dynamic performance limitations of agile fixed-wing UAVs must also be considered. In summary, trajectory planning in such settings must account for constraints related to time, safety, UAV dynamic performance, and trajectory smoothness.

Conventional fixed-wing UAVs control lateral-directional movements through the combined use of ailerons and rudders to minimize sideslip. Due to the strong maneuverability and large control surfaces of agile fixed-wing UAVs, they can achieve lateral movement primarily using the rudder, with minimal influence from sideslip [33]. Consequently, the ailerons of agile fixed-wing UAVs can be decoupled from controlling lateral-directional movement, treating the roll angle as an additional degree of freedom to enhance the survivability of agile fixed-wing UAVs in confined passageways. Based on this principle, the trajectory planning presented in this paper aims to optimize and solve for a state variable $(p_x, p_y, p_z, \phi)^T$, which is composed of 3-D position and roll angle.

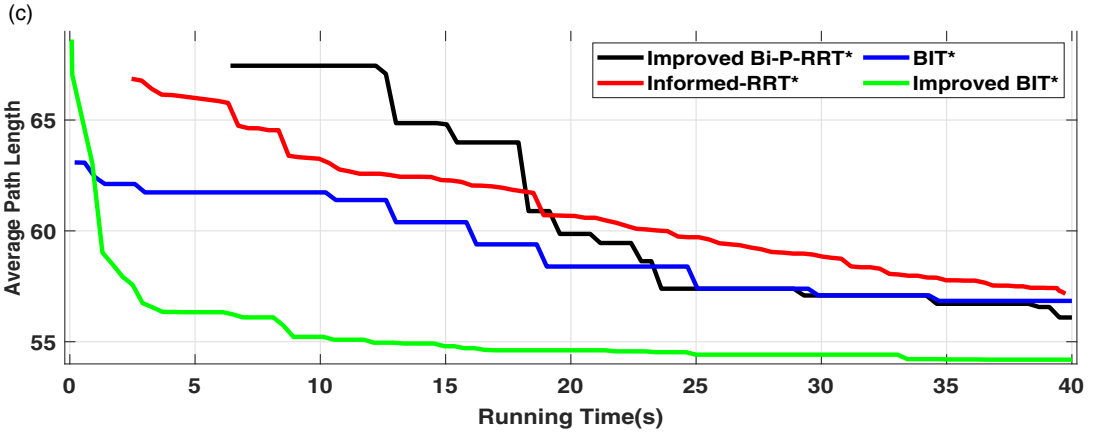
https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press



Simple scenario



Dense scenario



Maze scenario

Figure 5. *Performance comparison curves of the four algorithms in typical scenarios.*

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

Since the IBIT* algorithm solely provides 3-D path points without incorporating roll angles, and the trajectory to be planned also includes roll angles, it is necessary to determine the initial roll angles corresponding to each 3-D path point generated by the IBIT* algorithm. In accordance with the lateral-directional dynamics of the UAV, the initial value of the roll angle can be written as

$$\phi = \arctan \frac{\dot{\psi}\, V_a}{g}, \tag{5}$$

where V_a, ψ, and g represent the airspeed, the yaw angle, and the gravitational acceleration, respectively. In trajectory planning, suboptimal initial roll angles at path points may result in insufficient roll angles in narrow corridors, risking collisions. However, given the typically short length of narrow corridors in complex low-altitude environments, unsafe roll angle sequences are limited. Thus, this issue is resolved during safety detection: if collision risk is detected, the roll angle is incrementally adjusted (by 5° per step) until collision-free.

We have chosen MINCO for trajectory representation due to its capability to simultaneously optimize both time and space. MINCO inherently meets the condition of unconstrained minimum control efforts, which aligns with the requirement for trajectory smoothness. Moreover, MINCO can address the temporal constraints of mission time and the spatial constraints of safe flight corridors by leveraging its characteristics of spacetime deformation [22].

UAV dynamic performance constraints primarily include speed and attitude angular velocity limitations. By representing trajectories with MINCO, smoothness is guaranteed, avoiding abrupt lateral/altitude position changes. Since pitch and yaw angles control longitudinal/lateral motion, their magnitudes and rates are indirectly constrained. Thus, only velocity and roll angle constraints are necessary for UAV dynamic performance.

We formulate the constraints in the form of soft constraints, which are incorporated into the objective function in the form of penalty function for optimization [23]. To facilitate program implementation, the i-th segment trajectory is discretized with a resolution of κ_i. Therefore, **the objective function** for trajectory planning of an agile fixed-wing UAV in a complex low-altitude environment is

$$\min_{\mathbf{c(q,T)}} J = \lambda_V J_V + \lambda_\phi J_\phi + \lambda_{\omega_\phi} J_{\omega_\phi} + \lambda_{\phi_{la}} J_{\phi_{la}}$$

$$s.t. \begin{cases} J_V = \sum_{i=0}^{M_N} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} K\left(v^2\left(t_i, \frac{j}{\kappa_i}\right) - v_{\max}^2\right) + K\left(-v^2\left(t_i, \frac{j}{\kappa_i}\right) + v_{\min}^2\right) \\ J_\phi = \sum_{i=0}^{M_N} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} K\left(\phi^2\left(t_i, \frac{j}{\kappa_i}\right) - \phi_{\max}^2\right) \\ J_{\omega_\phi} = \sum_{i=0}^{M_N} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} K\left(\omega_\phi^2\left(t_i, \frac{j}{\kappa_i}\right) - \omega_{\phi_{\max}}^2\right) \\ J_{\phi_{la}} = \sum_{i=0}^{M_N} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} K\left(-\phi\left(t_i, \frac{j}{\kappa_i}\right) \psi\left(\dot{t}_i, \frac{j}{\kappa_i}\right)\right) \end{cases}, \tag{6}$$

where M_N is the number of trajectory points; $v_{\max}$, $v_{\min}$, $\phi_{\max}$, and $\omega_{\phi_{\max}}$ are the lower and upper bounds of the UAV velocity, the maximum roll angle, and the maximum roll rate, respectively; λ_V, λ_ϕ, λ_{ω_ϕ}, $\lambda_{\phi_{la}}$ is the weight corresponding to the penalty term; and T_i stands for the time of the i-th trajectory.

Maximum speed penalty J_V: Fixed-wing UAV have a minimum flight speed v_{min} and a maximum flight speed v_{max}, and thus J_V addresses the constraints from both aspects.

Maximum roll angle penalty J_ϕ: The UAV's roll angle has a range of $[0\ \pm 180^\circ]$.

Maximum roll rate penalty J_{ω_ϕ}: The UAV's roll rate is limited to a maximum roll rate of $\omega_{\phi_{\max}}$.

Roll angle direction penalty $J_{\phi_{la}}$: This describes the constraints on the roll angle direction of the UAV. The lateral forces acting on a fixed-wing UAV are generated by the deflection of both the rudder

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

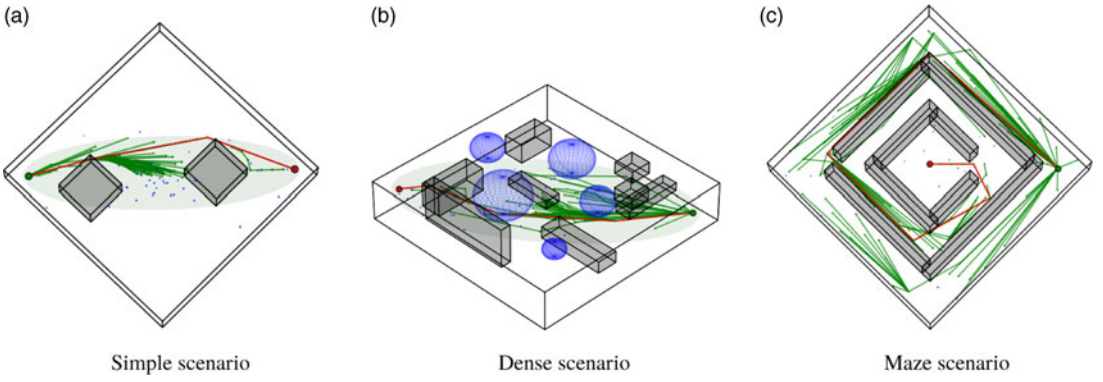


Figure 6. *Path planning of the IBIT* in typical scenarios.*

and the ailerons [34]. To improve flight stability, it is typically expected that the yaw rate resulting from rudder deflection aligns with the roll angle produced by aileron deflection.

The penalty function for the roll angle direction constraint can be designed as follows:

$$\tau_{la} = K(-\phi\dot{\psi})\ \dot{\tau_{la}} = -\dot{K}(-\phi\dot{\psi}) \cdot (\dot{\phi}\dot{\psi} + \phi\ddot{\psi}) \cdot \tag{7}$$

The expected yaw angle on the trajectory and its first and second derivatives are as follows:

$$\begin{aligned} \psi &= -\arctan\left(\frac{v_y}{v_x}\right),\ \dot{\psi} = \frac{a_x v_y - a_y v_x}{{v_x}^2 + {v_y}^2} \\ \ddot{\psi} &= \frac{\left(jerk_x v_y - jerk_y v_x\right)\left({v_x}^2 + {v_y}^2\right) - 2\left(a_x v_x + a_y v_y\right)\left(a_x v_y - a_y v_x\right)}{\left({v_x}^2 + {v_y}^2\right)^2} \cdot \end{aligned} \tag{8}$$

The state variables of the UAV, such as $v, v_x, a_x, jerk_x, \dot{\phi}$, can be represented using four independent state variables $z = (p_x, p_y, p_z, \phi)^T$ and their respective derivatives. When the directions of the roll angle and yaw rate are the same, there is $-\phi\dot{\psi} < 0, F(-\phi\dot{\psi}) < 0$; when the signs of the roll angle and yaw rate differ, there is $-\phi\dot{\psi} > 0, F(-\phi\dot{\psi}) > 0$, which can constrain the direction of the roll angle. The expressions for function $K(\cdot)$ and its derivatives are as follows:

$$K(x) = \begin{cases} 0 & x < 0 \\ \dfrac{2\mu_x x^3 - x^4}{2{\mu_x}^3} & 0 \le x \le \mu_x \\ x - \dfrac{1}{2}\mu_x & x > \mu_x \end{cases} \quad \frac{dK(x)}{dx} = \begin{cases} 0 & x < 0 \\ \dfrac{3\mu_x x^2 - 2x^3}{{\mu_x}^3} & 0 \le x \le \mu_x \\ 1 & x > \mu_x \end{cases}, \tag{9}$$

where μ_x is a range related to dynamic performance x, enabling approach to state limits. Parameters can vary with performance needs. The piecewise function $K(\cdot)$ is continuous and differentiable at segment points. It effectively constrains dynamic performance and acts as a penalty function.

By combining the gradient of the objective function with respect to the control variable and the time, along with the gradients of the penalties in Equation (6), we can obtain the complete gradient expression [22]. Finally, numerical optimization can be carried out using the gradient descent method.

4.2. Trajectory planning experiments for agile fixed-wing UAV in complex low-altitude environment

Figure 7 shows a complex low-altitude environment with a narrow corridor, whose width can be adjusted to test trajectory planning in narrow spaces. Experimental parameters are shown in Table III.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

Table III. *Trajectory planning constraints and experimental parameter settings.*

Parameter	Symbol	Value	Unit
Flight speed range	$[v_{\min}, v_{\max}]$	[10, 20]	m/s
Roll angle range	$[\phi_{\min}, \phi_{\max}]$	$[-180, 180]$	deg
Roll angle rate range	$[\omega_{\phi\,\min}, \omega_{\phi\,\max}]$	$[-4, 4]$	rad/s
Control derivative order	s	3	
Trajectory resolution	κ	5	
Mission time	T_Σ	9	s
Initial and final acceleration	$[a_0, a_M]$	[0, 0]	m/s^2
Constraint weight	$\lambda_V, \lambda_\phi, \lambda_{\omega_\phi}, \lambda_{\phi_{la}}$	[1, 1, 0.5, 0.25]	
Constraint tolerance	$\mu_v, \mu_\phi, \mu_{\omega_\phi}, \mu_{\phi_{la}}$	[0.01, 0.01, 0.01, 0.01]	

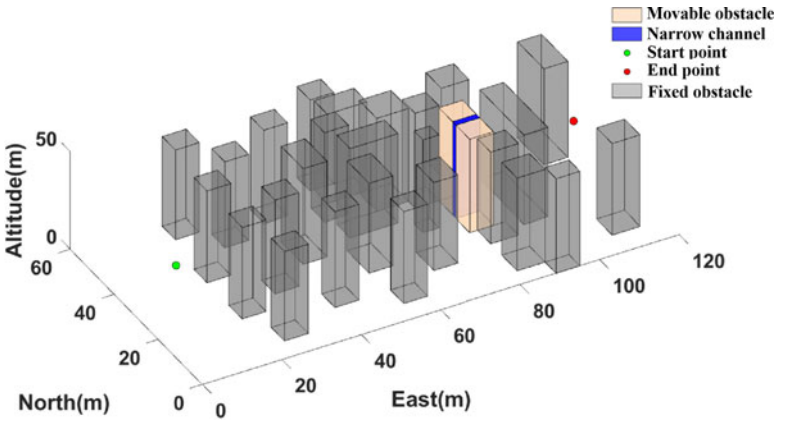


Figure 7. *Typical complex low-altitude environment.*

Considering the ability of the small agile fixed-wing UAV, we use to fly safely in complex low-altitude environments, it is important to recognize that the UAV is a physical object with specific dimensions and should not be treated as a mass. In order to navigate safely through narrow corridors, it is necessary to model a safe width for a small agile fixed-wing UAV. A frontal view and safe width schematic of this UAV is shown in Figure 8, where the height of the vehicle is $h_{uav} = 43.5cm$ and the width is $w_{uav} = 122cm$ when all three-axis attitude angles are set to zero.

To ensure sufficient flight safety, a lateral safety distance $d_{safe} = 50cm$ is specified, and the relationship between the UAV's safety width w_{ef} and the aircraft's roll angle ϕ is given by Equation (10).

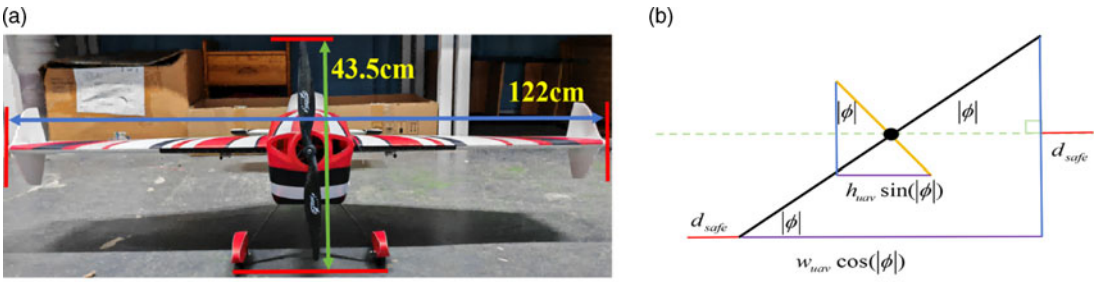
$$w_{ef} = \begin{cases} h_{uav} \sin(|\phi|) + 2d_{safe} & |\phi| \in [70.376^\circ \quad 109.624^\circ] \\ w_{uav} \cos(|\phi|) + 2d_{safe} & otherwise \end{cases}, \tag{10}$$

The UAV requires a maximum safe width of $w_{ef_\max} = 2.22m$ when $|\phi| = k\pi$ $(k = 0, 1)$, and a minimum width of $w_{ef_\min} = 1.4m$ when $|\phi| = 70.376^\circ$ or $|\phi| = 109.624^\circ$. These widths correspond to 1.82 and 1.15 times the wingspan of the UAV, respectively.

4.2.1 Experiment with a narrow corridor width of 1.82 times the wingspan

The narrow corridor is $2.22m$ wide, with initial/final velocities of $15m/s$ and roll angles of 0°. Trajectory planning under these conditions generates a constraint-satisfying trajectory (Figure 10) with $71.8ms$ computation time. The trajectory in the narrow corridor, along with the UAV's lateral safety radius and the minimum safe distance to obstacles, is shown in Figure 9. In this segment of the trajectory, the UAV's lateral safety radius always remains smaller than the minimum safe distance from the obstacles.

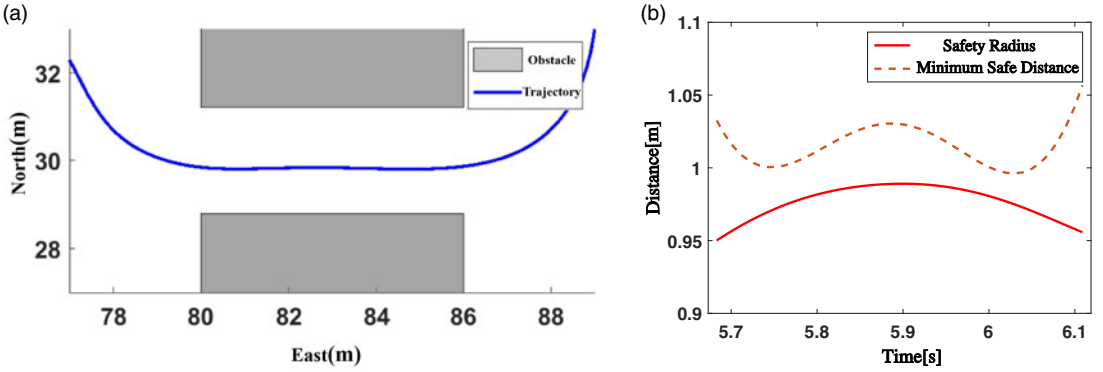
https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press



Frontal view of the UAV and its dimensions

The lateral safety width of the UAV

Figure 8. *Frontal view and safety width diagram of small agile fixed-wing UAV.*



Safe trajectory in the narrow corridor

UAV's lateral safety radius

Figure 9. *Trajectory planning in the narrow corridor.*

4.2.2 Experiment with a narrow corridor width of 1.15 *times the wingspan*

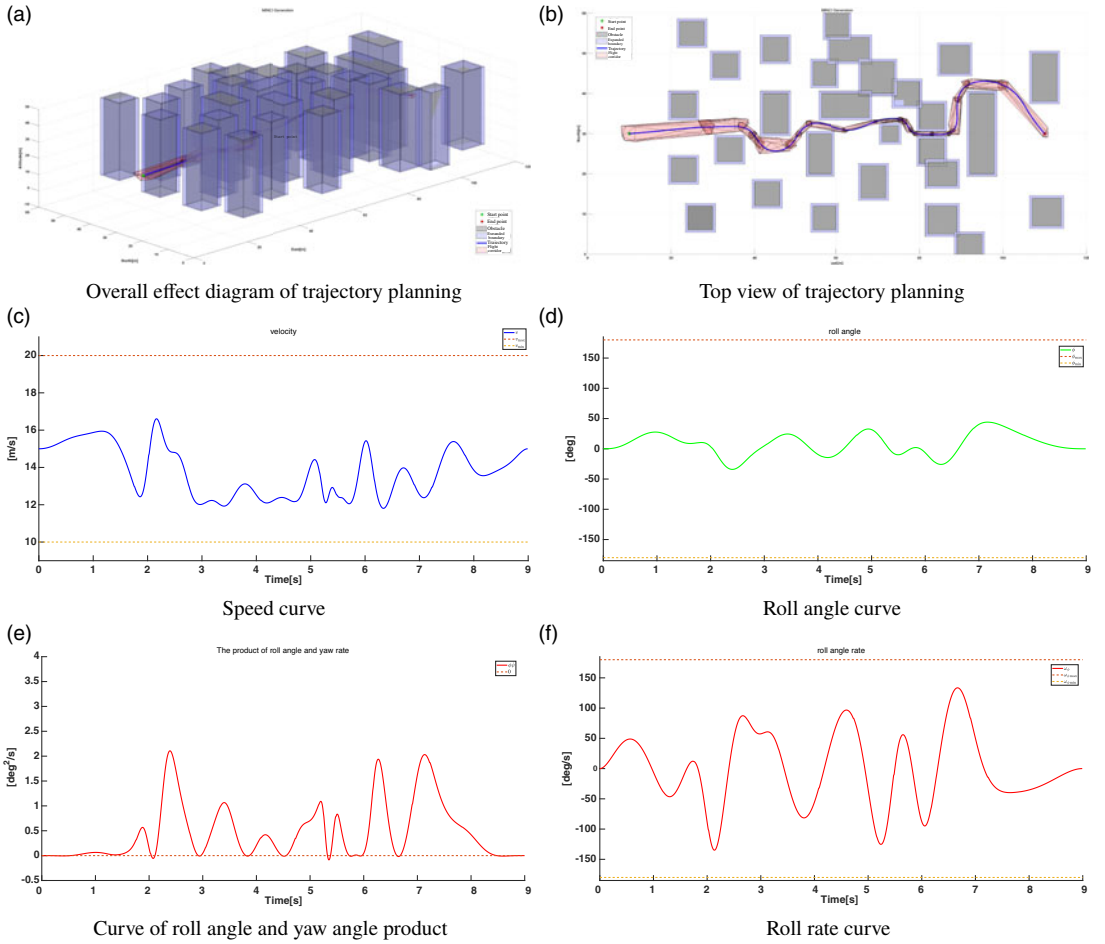
The width of the narrow corridor is set to 1.4m, and the initial conditions are the same as those in the previous experiment. Under these initial conditions, the trajectory planning results are shown in Figure 12. The algorithm took only 67.6ms, and the planned trajectory is smooth, remaining entirely within the SFC. The velocity, roll angle, roll rate, and other parameters of the trajectory all satisfy the constraints.

Figure 11 shows the UAV trajectory through the 1.15 times wingspan narrow corridor and the minimum safe distance curve. The smooth trajectory maintains safe lateral distances from obstacles. Throughout the passage, the lateral safety radius remains below the minimum distance, confirming safe navigation with proper roll angles.

In typical scenarios, the algorithm for trajectory planning can be controlled to within 100ms. Considering that motion planning involves steps such as path planning and the construction of a SFC, our single motion planning can still be maintained around 200ms, which meets the requirements for online planning.

Furthermore, it is important to note that the trajectory planning method we proposed is the first to introduce MINCO to agile fixed-wing UAV, enabling planning with roll angles. We have also designed a penalty function for soft constraint numerical optimization. Our trajectory planning aims to achieve higher solution efficiency while satisfying constraints. Given that numerical optimization methods already demonstrate higher efficiency compared to other optimization approaches, and that MINCO exhibit greater efficiency than other trajectory representations, such as B-spline, it can be concluded that our proposed trajectory planning method offers distinct advantages.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press



Overall effect diagram of trajectory planning

Top view of trajectory planning

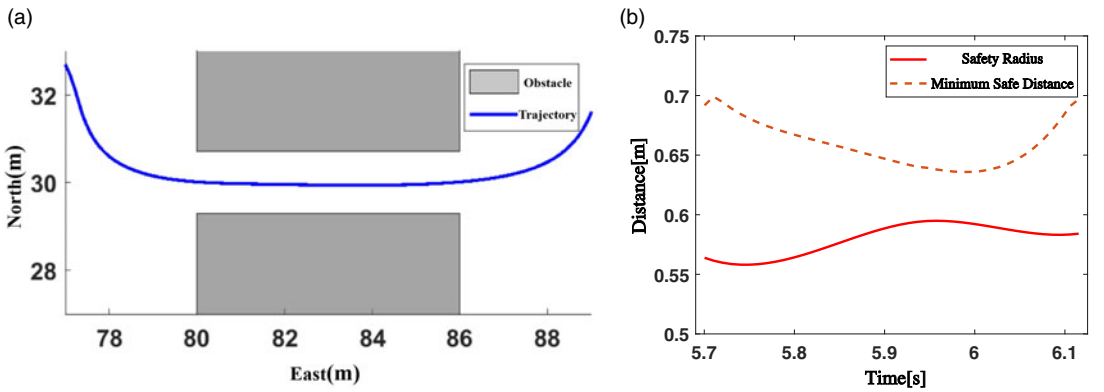
Speed curve

Roll angle curve

Curve of roll angle and yaw angle product

Roll rate curve

Figure 10. *Trajectory planning for a narrow corridor width of* 1.82 *times the wingspan.*



Safe trajectory in the narrow corridor

UAV's lateral safety radius

Figure 11. *Trajectory planning in the narrow corridor.*

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

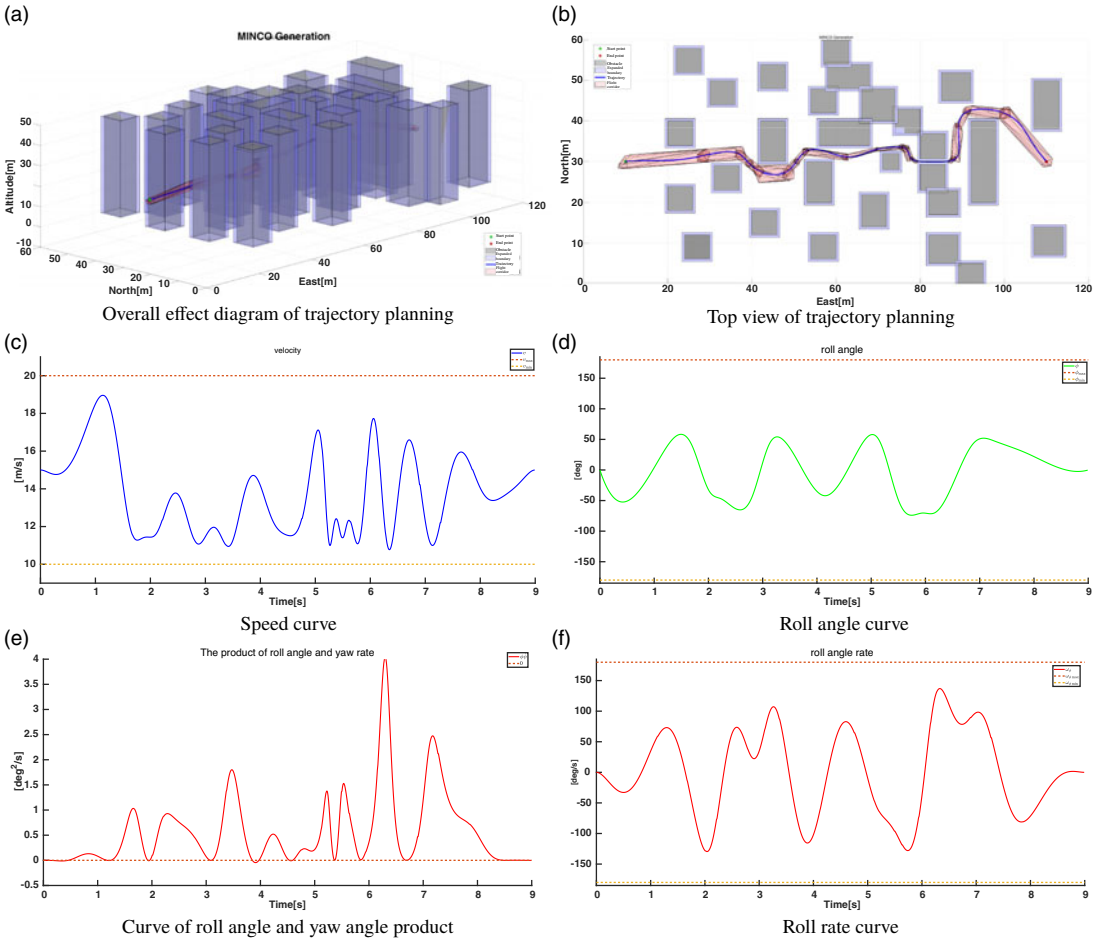


Figure 12. *Trajectory planning for a narrow corridor width of* 1.15 *times the wingspan.*

5. Physical flight test in a virtual complex low-altitude environment

We control the agile fixed-wing UAV using ROS and PX4, with MAVROS enabling data transmission and acquisition. Figure 13 illustrates our designed autonomous flight software framework. The left side of the figure, implemented on the upper computer, adopts the trajectory tracking strategy from [29]. While real low-altitude environments require perception sensors for local data, we employ virtual environmental perception due to construction challenges and our non-focus on perception/mapping. For instance, we simulate a $20m$-radius detection range using a virtual lidar, assuming the UAV acquires environmental data within this scope.

The hardware system (Figure 14) features an EDGE-540T-30E-based UAV measuring $1.21m$ length, $1.22m$ wingspan, $30dm^2$ wing area, and $1.52Kg$ total mass. Constructed from lightweight yet strong PP material, its three control surfaces achieve 50° maximum deflection for exceptional roll/lateral performance. The platform integrates a Pixhawk-6X flight controller, Jetson Xavier NX computer, enhanced-torque TMOTOR AM480-KV900 motors (2:1 thrust-to-weight ratio), and GDW DS296X digital servos for rapid deflection response.

To validate the effectiveness of the motion planning algorithm for agile fixed-wing UAVs in complex low-altitude environments, two virtual scenarios are created. Scenario one tests whether the UAV can safely traverse a $5m$-wide narrow corridor, while scenario two assesses obstacle avoidance capabilities in environments with both vertical and horizontal obstacles. Figure 15 illustrates these scenarios.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

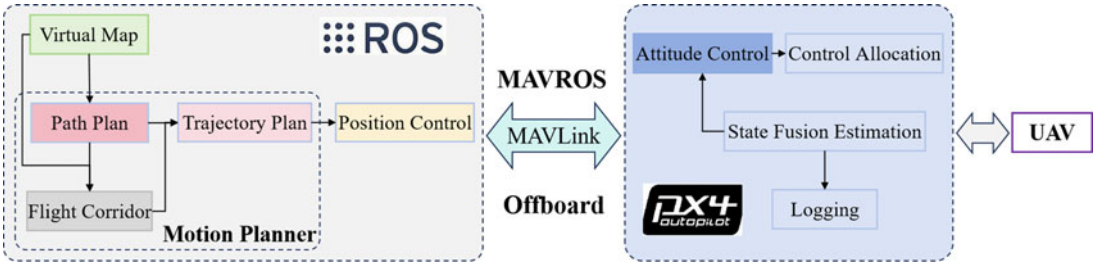


Figure 13. *Flight system software framework.*

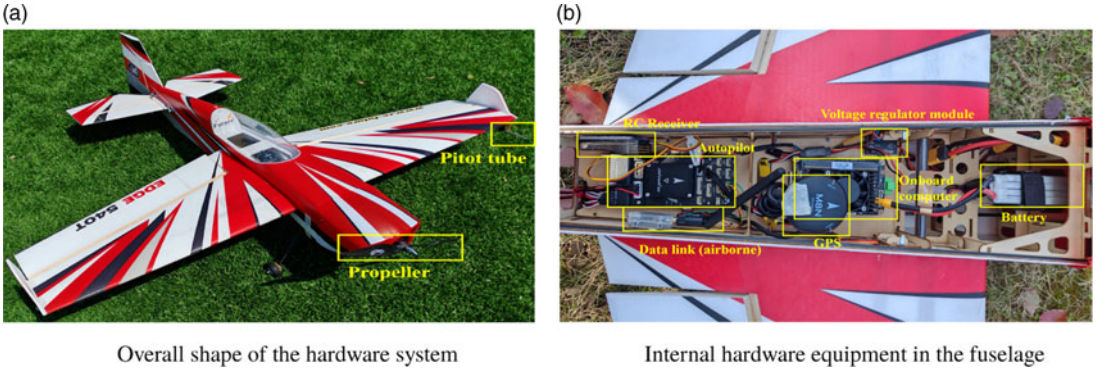


Figure 14. *Small agile fixed-wing UAV test platform.*

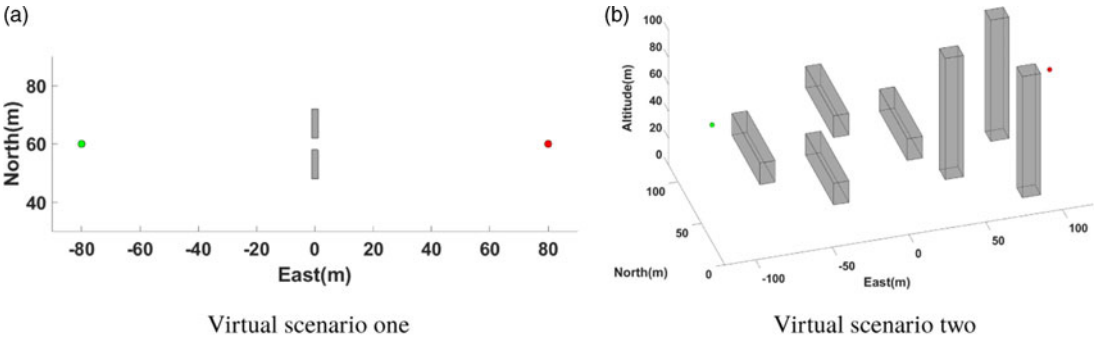


Figure 15. *Virtual scenario for flight experiments.*

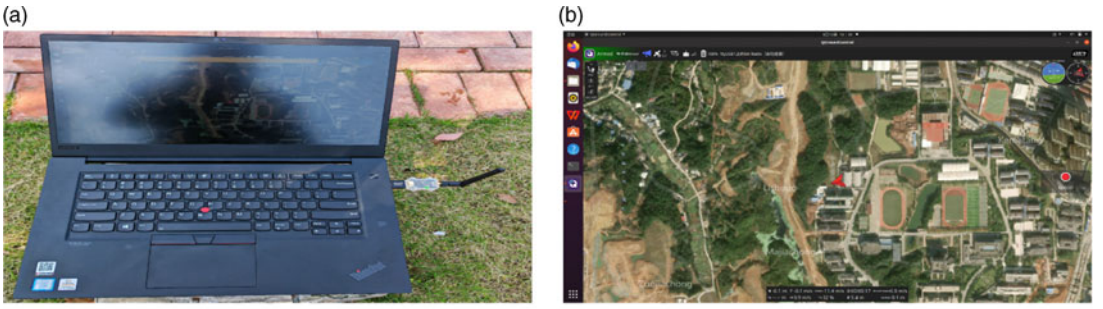
Figure 16 shows the experimental setup with ground control computer and QGroundControl (QGC) interface. The computer connects to the ground station for real-time UAV status monitoring via QGC. Pre-flight procedures include sensor calibration (accelerometer, magnetometer, level, and airspeed) and configuring the remote controller's mode/safety switches until QGC confirms green arm status.

The process of conducting flight experiments in virtual scenario one is illustrated in Figure 17, where the agile fixed-wing UAV maneuvers through a narrow corridor with a significant roll angle.

Figure 18 presents the experimental results in virtual scenario one, where the closest distance between the UAV and the obstacle while passing through the narrow corridor is always greater than the safety distance, indicating that the UAV safely traverses the narrow corridor formed by the obstacles.

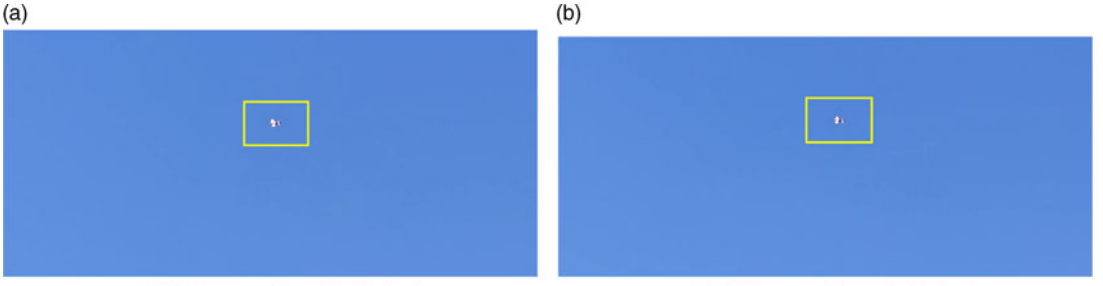
Flight experiments in Virtual Scenario 2 are shown in Figure 19, illustrating the ascent and lateral turning flights of the agile fixed-wing UAV to avoid obstacles.

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press



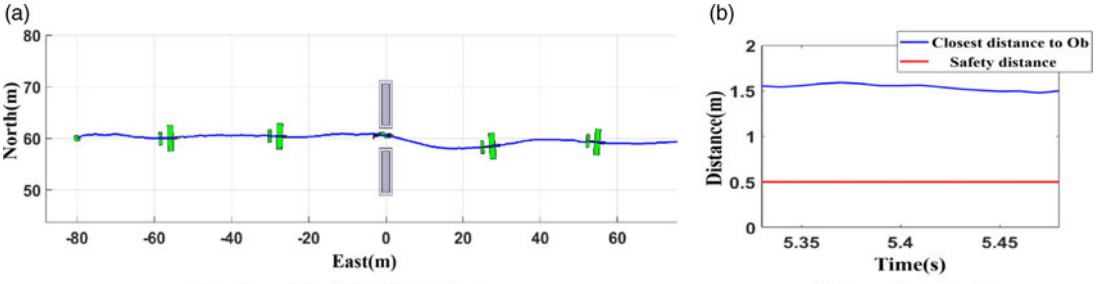
Ground station control computer QGC display in the flight experiment scenario

Figure 16. *Ground station computer and QGC display screen.*



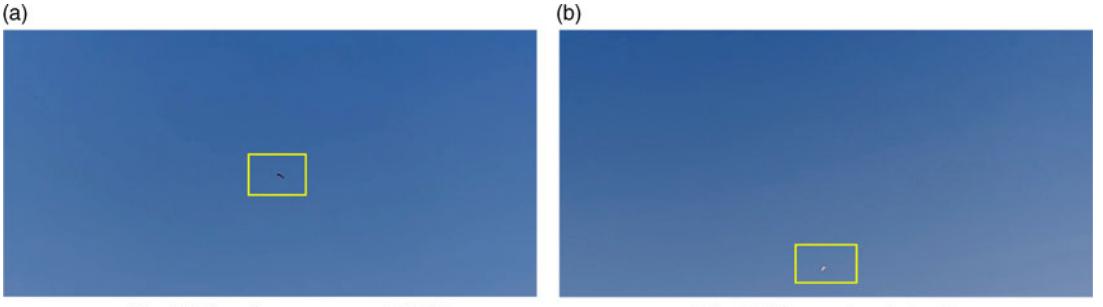
UAV large roll angle flight 1 UAV large roll angle flight 2

Figure 17. *Images from the flight experiment process in virtual scenario 1.*



Top view of the UAV flight trajectory Distance from obstacles

Figure 18. *Experimental results in virtual scenario 1.*



The UAV performs an ascent flight The UAV executes a lateral turn

Figure 19. *Images from the flight experiment process in virtual scenario 2.*

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

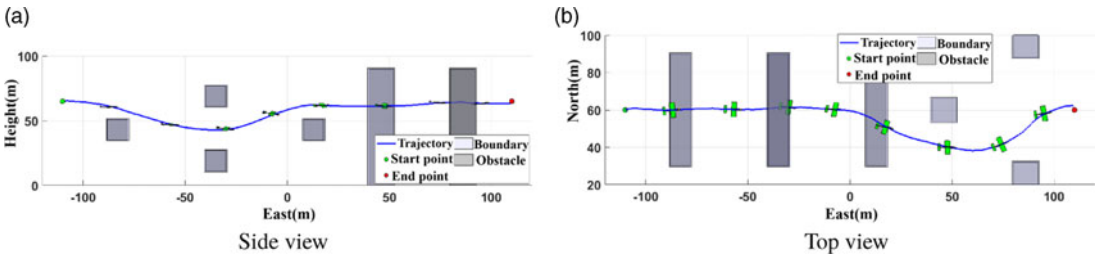


Figure 20. *Experimental results in virtual scenario 2.*

In Figure 20, the UAV's flight in Virtual Scenario 2 shows that the distance between the UAV and the obstacles is significantly greater than the safety distance, allowing the UAV to safely traverse the narrow corridor created by the obstacles.

6. Conclusion and future works

This paper investigates the motion planning problem for agile fixed-wing UAV in complex low-altitude environment. In the path planning layer, we propose the IBIT*, which has been verified in typical scenarios to improve both the quality of the planned path and the computation speed. In the trajectory planning layer, we decouple the UAV's roll angle from its 3-D position, eliminate the spatial and temporal constraints of the flight corridor, and design a penalty function based on the dynamic performance constraints of the UAV. Experiments conducted in typical scenarios demonstrate the effectiveness of the trajectory planning. Finally, we built a physical flight system for a small agile fixed-wing UAV, which safely navigated in a complex low-altitude environment during practical experiments, validating the effectiveness of our proposed methods.

Although we have achieved safe flight for the agile fixed-wing UAV in complex low-altitude environments, there are still some issues that need to be addressed in the future. We can explore environmental perception methods suitable for fixed-wing UAVs in complex low-altitude environments, particularly by equipping small visual or laser sensors to enable real-time state estimation on board, helping the UAV accurately determine its pose information. Additionally, we can continue to develop the maneuverability of agile fixed-wing UAVs, which have the potential for stall flight at high angles of attack.

Author contributions. F. Zou conceived, designed, and completed the research and writing of the paper; J. Li participated in the design of the research ideas; Y. Niu provided the overall writing strategy for the paper and support for the flight experiments.

Financial support. This work was supported by the National Natural Science Foundation of China under Grant 61876187.

Conflicts of interest. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] Y. Zhang, G. Xu and G. Shan, "Sensor cooperative scheduling method for low-altitude maneuvering target tracking in complex environment," *Sens. Rev.* **42**(1), 133–144 (2022).

[2] M. H. Garibeh, A. M. Alshorman, M. A. Jaradat, A. B. Younes and M. Khaleel, "Motion planning of unmanned aerial vehicles in dynamic 3D space: A potential force approach," *Robotica* **40**(10), 3604–3630 (2022).

[3] W. Khan and M. Nahon, "Modeling dynamics of agile fixed-wing UAVs for real-time applications," **In:** *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, Arlington, VA, 2016) pp. 1303–1312.

[4] Z.-H. Cheng and H.-L. Pei, "Transition analysis and practical flight control for ducted fan fixed-wing aerial robot: Level path flight mode transition," *IEEE Rob. Autom. Lett.* **7**(2), 3106–3113 (2022).

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

[5] M. Basescu, A. Polevoy, B. Yeh, L. Scheuer, E. Sutton and J. Moore, "Agile fixed-wing UAVs for urban swarm operations," *IEEE Trans. Field Robot* **3**(1), 725–765 (2023).

[6] K. K. S. Kumar, H. Arya and A. Joshi, "Performance of Variable Pitch Propeller for Longitudinal Control in an Agile Fixed-Wing UAV," **In:** *2019 Aerospace Conference* (IEEE, Big Sky, MT, 2019) pp. 1–11.

[7] Y. Wu, "A survey on population-based meta-heuristic algorithms for motion planning of aircraft," *Swarm Evol. Comput*. **62**, 100844 (2021).

[8] L. Dong, Z. He, C. Song and C. Sun, "A review of mobile robot motion planning methods: From classical motion planning workflows to reinforcement learning-based architectures," *J. Syst. Eng. Electron.* **34**(2), 439–459 (2023).

[9] L. Quan, Z. Han, B. Zhou, S. Shen and F. Gao, "Survey of UAV motion planning," *IET Cyber-Syst. Robot* **2**(1), 14–21 (2020).

[10] A. Israr, Z. A. Ali, E. H. Alkhammash and J. J. Jussila, "Optimization methods applied to motion planning of unmanned aerial vehicles: A review," *Drones* **6**(5), 126 (2020).

[11] M. Jones, S. Djahel and K. Welsh, "Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey," *ACM Comput. Surv*. **55**(11), 1–39 (2023).

[12] C. Zammit and E. J. v. Kampen, "Comparison between A* and RRT algorithms for 3D UAV path planning," *Unmanned Systems* **10**(2), 129–146 (2022).

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.* **30**(7), 846–894 (2011).

[14] G. Gao, D. Li, K. Liu, Y. Ge and C. Song, "A study on path-planning algorithm for a multi-section continuum robot in confined multi-obstacle environments," *Robotica* **42**(10), 3324–3347 (2024).

[15] J. Gammell, S. Srinivasa and T. B. S. Srinivasa, "Informed RRT*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic," **In:** *2014 International Conference on Intelligent Robots and Systems (lROS)* (IEEE/RSJ, Chicago, IL, 2014) pp. 2997–3004.

[16] J. Gammell, T. Barfoot and S. Srinivasa, "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search," *Int. J. Rob. Res.* **39**(5), 543–567 (2020).

[17] Y. Guo, Z. Guo, Y. Wang, D. Yao, B. Li and L. L., "A survey of trajectory planning methods for autonomous driving – part I: Unstructured scenarios," *IEEE Trans. Intell. Veh.* **10**(6), 1–29 (2023).

[18] B. Zhu, J. He and J. Sun, "Kinematic modeling and hybrid motion planning for wheeled-legged rovers to traverse challenging terrains," *Robotica* **42**(1), 3324–3347 (2024).

[19] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control For Quadrotors," **In:** *2011 International Conference on Robotics and Automation (lCRA)* (IEEE, Shanghai, China, 2011) pp. 2520–2525.

[20] H. Huang, Y. Shen, H. Zhang, G. Huang and R. Yang, "Improved UCAV Attack Trajectory Planning Algorithm Based on GWO and Dubins Curve," **In:** *2022 International Conference on Autonomous Unmanned Systems (lCAUS)* (Springer, Xi'an, China, 2022) pp. 1995–2004.

[21] V. Usenko, L. von Stumberg, A. Pangercic and D. Cremers, "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer," **In:** *2017 International Conference on Autonomous Unmanned Systems (lROS)* (IEEE/RSJ, Vancouver, BC, Canada, 2017) pp. 215–222.

[22] Z. Wang, X. Zhou, C. Xu and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.* **38**(5), 3259–3278 (2022).

[23] A. Bry, C. Richter, A. Bachrach and N. Roy, "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments," *Int. J. Rob. Res.* **34**(7), 969–1002 (2015).

[24] E. Tal, G. Ryou and S. Karaman, "Aerobatic trajectory generation for a VTOL fixed-wing aircraft using differential flatness," *IEEE Trans. Robot.* **39**(6), 4805–4819 (2023).

[25] X. Zhou, Z. Wang, H. Ye, C. Xu and F. Gao, "EGO-Planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Rob. Autom. Lett.* **6**(2), 478–485 (2021).

[26] X. Zhou, X. Wen, Z. Wang and F. Gao, "Swarm of micro flying robots in the wild," *Sci. Robot.* **7**(66), 5954 (2022).

[27] J. Zhang, "Kinodynamic motion planning for robotics: a review," **In:** *2021 International Conference on Robotics and Automation Sciences (lCRAS)* (IEEE, Wuhan, China, 2021) pp. 75–83.

[28] T. Liu, J. Li and F. Zou, "Differential Flatness-Based Trajectory Planning and Tracking for Fixed-Wing Aircraft in Clustered Environments," **In:** *2023 Chinese Control Conference (CCC)* (IEEE, Tianjin, China, 2023) pp. 3627–3631.

[29] J. C. H. Ramírez and M. Nahon, "Nonlinear Vector-Projection Control for Agile Fixed-Wing Unmanned Aerial Vehicles," **In:** *2020 International Conference on Robotics and Automation (lCRA)* (IEEE, Paris, France, 2020) pp. 5314–5320.

[30] J. Chang, N. Dong and D. Li, "Skeleton extraction and greedy-algorithm-based path planning and its application in UAV trajectory tracking," *IEEE Trans. Aerosp. Electron. Syst*. **58**(6), 4953–4964 (2022).

[31] X. Cui, C. Wang, Y. Xiong, L. Mei and S. Wu, "More quickly-RRT*: Improved quick rapidly-exploring random tree star algorithm based on optimized sampling point with better initial solution and convergence rate," *Eng. Appl. Artif. Intell.* **133**(3), 108246 (2024).

[32] J. Fan, X. Chen and X. Liang, "UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased," *Expert Syst. Appl*. **213**(C), 119137 (2023).

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press

[33] J. C. H. Ramírez and M. Nahon, "Aerobatics on three-dimensional paths for agile fixed-wing unmanned aerial vehicles," **In:** *2022 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, Dubrovnik, Croatia, 2022) pp. 1547–1554.
[34] X. Lu, Z. Li and J. Xu, "Design and control of a hand-launched fixed-wing unmanned aerial vehicle," *IEEE Trans. Ind. Inform*. **19**(3), 3006–3016 (2023).
[35] Z. Ma, Z. Wang, A. Ma, Y. Liu and Y. Niu, "A low-altitude obstacle avoidance method for UAVs based on polyhedral flight corridor," *Drones* **7**(9), 588 (2023).

Cite this article: F. Zou, J. Li and Y. Niu (2025). "Motion planning for agile fixed-wing UAVs in complex low-altitude environments", Robotica **43**, 1640–1659. https://doi.org/10.1017/S0263574725000554

https://doi.org/10.1017/S0263574725000554 Published online by Cambridge University Press