

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

AUTOMATED LANDING OF QUADROTORS
ON AN UNMANNED AERIAL VEHICLE
CARRIER IN UNKNOWN ENVIRONMENTS:
TRAJECTORY PLANNING AND
NONLINEAR MODEL PREDICTIVE
CONTROL

ZHANG CHENGCHEN

MPhil

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Aeronautical and Aviation Engineering

**Automated Landing of Quadrotors on
an Unmanned Aerial Vehicle Carrier
in Unknown Environments:
Trajectory Planning and Nonlinear
Model Predictive Control**

ZHANG Chengchen

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Philosophy

August 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Zhang Chengchen

(Name of student)

Abstract

Unmanned Aerial Vehicles (UAVs) have found widespread application across diverse fields, yet their operational endurance remains severely constrained by on-board energy storage. While researchers have explored technologies such as high-energy-density batteries, in-flight wireless power transfer, and ground-based charging stations to extend operational duration, these approaches offer only incremental improvements and fail to meet the demand for a self-contained, flexible platform capable of transporting, deploying, and autonomously retrieving a fleet of UAVs in remote areas without human intervention. Addressing this gap, this thesis focuses on the automated landing of quadrotors on an Unmanned Aerial Vehicle Carrier (UAVC), a "mother ship" that serves as a base for take-off, landing, and recovery, with the aim of enhancing autonomy and operational sustainability in unknown environments.

This task presents significant challenges, with two core aspects being real-time collision-free path planning and the design of robust control systems to mitigate disturbances caused by downwash winds. Accordingly, the research objective is to develop effective, scenario-specific trajectory planning and control strategies to address this automated landing task. Two trajectory planners are proposed: a hard-constrained planner that ensures strict compliance with collision and dynamic constraints via global pathfinding and convex decomposition, and a soft-constrained planner that balances efficiency and safety through integrated global planning and local refinement. A Nonlinear Model Predictive Control (NMPC) framework with a dynamic downwash wind model is also designed to improve landing stability.

Key results validate our proposed methods. The hard-constrained planner achieves full success in simulations, ensuring strict compliance with collision and dynamic constraints, a feature critical for high-risk scenarios. Meanwhile, the soft-constrained planner maintains a 95% success rate while achieving significantly higher speeds and computation times reduced by half, making it well-suited for common real-world environments. The NMPC framework with the downwash model significantly reduces vertical position errors during landing, and real-world tests confirm the soft-constrained planner's robustness with low tracking errors.

This work holds significant value in addressing a critical gap in UAV operations: enabling reliable autonomous landing on moving aerial carriers, thereby overcoming the endurance limitations of battery-powered UAVs and facilitating sustained, distributed missions in remote or complex environments. By focusing on unknown environments, it fills a void in existing trajectory planning and control methods that often rely on prior environmental knowledge or struggle with dynamic disturbances like downwash winds. Specifically, the proposed trajectory planning frameworks can be applied in different scenarios by balancing the trade-off between strict constraint compliance and real-time performance. The hard-constrained approach ensures collision avoidance and dynamic feasibility in high-risk scenarios, while the soft-constrained method demonstrates robustness and efficiency in common real-world scenarios. The NMPC framework with the downwash model further enhances stability, reducing vertical oscillations of the UAVC by 68% in real-world tests. Comprehensive experimental validation including simulated tests in Gazebo and real-world trials confirms the practical applicability of these methods, bridging the gap between theoretical design and real-world implementation.

Keywords: UAV Application; Trajectory Planning; Model Predictive Control; Unmanned Aerial Vehicle Carrier.

Publications Arising from the Thesis

C. Zhang, Y. L. Lam, C. M. B. Ip, and H. Huang, "Automated Landing of Quadrotors on an Unmanned Aerial Vehicle Carrier via Real-Time Trajectory Planning and Nonlinear Model Predictive Control," in *2025 IEEE 19th International Conference on Control & Automation (ICCA)*, Tallinn, Estonia, 2025, pp. 411–416, doi: 10.1109/ICCA65672.2025.11129795.

C. Zhang, Y. L. Lam, C. M. B. Ip, and H. Huang, "Dynamic Path Optimization and Nonlinear Model Predictive Control for Autonomous UAV Landings on Mobile Aerial Platforms," *Unmanned Systems*, accepted.

Contents

1	Introduction	1
2	Literature Review	5
2.1	UAV Docking Station Typology and Capabilities	6
2.2	Global Path Planning Methods for UAVs	11
2.3	Local Trajectory Planning: Hard-constrained and Soft-constrained Optimization	13
2.3.1	Hard-constrained Optimization	14
2.3.2	Soft-constrained Optimization	17
2.4	UAV Control Strategies: From Traditional Controllers to Advanced MPC	20
2.5	Summary	24
2.6	Notation	26
3	Hard-constrained Optimization with Safety Flight Corridor	27
3.1	Global Planner via Jump Point Search	27
3.1.1	Pruning Rules	28
3.1.2	Jump Point Algorithm	30
3.2	Convex Decomposition	32
3.2.1	Find Ellipsoids	32
3.2.2	Find Polyhedron	34
3.2.3	Bounding Box	35
3.3	Local Planner via Mixed-Integer Quadratic Programming	36

4	Soft-constrained Optimization via Kinodynamic Path Search and B-spline Refinement	41
4.1	Global Planner via Kinodynamic Path Search	41
4.2	Local Planner via B-Spline Refinement with Cost Functions	44
5	Nonlinear Model Predictive Control with Downwash Wind Model	52
5.1	Dynamics Model of Quadrotors	52
5.2	Generation of NMPC	55
6	Experimental Verification and Result Analysis	59
6.1	System Architecture and Implementation Environment	60
6.2	Simulated Experiment Setup	63
6.3	Experiment Result in Simulated Environment	66
6.3.1	The Result of Simulated Experiment about Hard-constrained Planner	67
6.3.2	The Result of Simulated Experiment about Soft-constrained Planner	70
6.3.3	Comparison between Different Planners	72
6.3.4	Stability Test of the Proposed Controller	75
6.4	Real-World Experiment Setup	79
6.5	Experiment Result in Real-world Experiment	84
6.5.1	Determination of the Downwash Wind Model	84
6.5.2	Experiment Result	86
7	Conclusion	96
7.1	Summary of Contributions	96
7.2	Future Research	97

List of Figures

1	System architecture of the UAV, with the modules highlighted in orange marking the primary focus of our work.	5
2	An example of the fixed-docking system	7
3	Classification of UAV docking systems	7
4	An example of the UGV–UAV system	8
5	An example of the USV–UAV system	9
6	Transportation of working UAV	10
7	2-dimensional pruning case	29
8	Ellipsoid generation around a path segment	33
9	Process of constructing the convex polyhedron	35
10	Process of formulating the bounding boxes	36
11	Visualization of the local-planner trajectory parameterization	37
12	Collision-avoidance via the B-spline convex hull	47
13	The UAV Model in the Body and World Frame	53
14	Overview of the system	61
15	System Structure for the UAV and the UAVC, with the modules highlighted in orange marking the primary focus of our work.	63
16	Simulated UAV and UAVC with basic parameters.	64
17	Scenario setting in the simulation experiment. Green cylinders represent obstacles.	64
18	Dynamic reconfiguration	65
19	The overall working process in simulation	67

20	An example of applying the hard-constrained planner in the simulation	68
21	Generated trajectory and the following result by applying the proposed hard-constrained planner	69
22	Tracking error during flight.	69
23	Generated trajectory and the following result by applying soft-constrained planner.	71
24	Tracking error during flight.	72
25	Trajectories generated by different planning methods	74
26	Thrust percentage variation during the landing process.	76
27	Variations in the UAVC's position along the X, Y, and Z axes during the landing process.	78
28	The Intel RealSense D435 used for obstacle detection	80
29	The flight controller used in the UAV	81
30	The flight controller used in the UAVC	81
31	The UAV and UAVC used in the experiment.	82
32	Environment in real-world experiment.	83
33	Anemograph used in the real-world experiment	85
34	Experimental wind speed data (markers) and the fitted exponential decay curve (line)	86
35	Snapshots of a real-world experiment.	87
36	Generated trajectory and following result in real-world experiment .	88
37	Tracking error of the UAV in the real-world experiment	89
38	Thrust variation across the three landing phases.	90
39	Variations in the UAVC's position along the X-, Y- and Z-axes during the landing process when the dynamic model for the downwash flow is not considered.	92

40	Variations in the UAVC's position along the X-, Y- and Z-axes during the landing process when the dynamic model for the downwash flow is considered.	94
----	--	----

List of Tables

1	Key Parameters Mentioned in the Later Sections and Their Meanings	26
2	Parameter settings in the simulation experiment	66
3	Tracking error for the simulation experiment by applying hard- constrained planning	70
4	Tracking error for the simulation experiment	72
5	Comparison of planning methods based on simulation results	75
6	RMSE of the UAVC's position during the landing process.	79
7	Parameter settings in the real-world experiment	84
8	Tracking error in each direction	89
9	RMSE of the UAVC's position during the landing process (without the dynamic downwash flow model).	93
10	RMSE of the UAVC's position during the landing process (with the dynamic downwash flow model).	95

1 Introduction

Unmanned aerial vehicles (UAVs) have undergone a remarkable evolution over the past decade, transitioning from niche research platforms to versatile systems deployed in a wide variety of civil, commercial, and military applications [1–5]. Their compact form factor, high maneuverability, and rapidly improving payload capabilities have enabled missions ranging from precision agriculture and infrastructure inspection to search and rescue operations in challenging environments [6–8], package delivery in urban settings [9, 10], and monitoring in remote regions [11–14]. As these applications proliferate, however, one fundamental limiting factor persists: the finite on-board energy storage of battery-powered UAVs constrains flight endurance, mission range, and overall autonomy [15].

Most lightweight multirotor platforms, including ubiquitous consumer and enterprise models such as the DJI Mavic and Matrice series, which depend on lithium-polymer (LiPo) batteries with energy densities of roughly 150–200 Wh/kg, yielding flight durations of only 20–30 minutes under no-payload conditions [16]. When carrying common payloads of 0.5–2 kg, this endurance typically drops by 30–50%, forcing frequent landings for battery swaps or recharge cycles [17, 18]. This persistent energy bottleneck inflates logistical overhead, requiring larger battery inventories, dedicated charging infrastructure, and increased ground support, and it limits UAV autonomy in remote or hazardous environments.

To address the endurance limitations of UAVs and enable sustained operations in remote locations, researchers have explored a range of enabling technologies, with a focus on high-energy-density batteries and in-flight wireless power transfer.

First, efforts to enhance battery performance have centered on advanced lithium-ion chemistries, such as nickel-manganese-cobalt (NMC) and nickel-cobalt-aluminum (NCA) formulations, which aim to achieve gravimetric energy densities exceeding 200 Wh/kg. By leveraging improved electrode materials and optimized electrolyte compositions, modern battery packs can extend the flight duration of small UAVs by approximately 30–50% compared to conventional cells. However, these advancements come with trade-offs: increased manufacturing costs, stricter thermal management requirements, and reduced cycle life due to accelerated electrode degradation [19, 20].

Meanwhile, as an alternative to landing for recharging, in-flight wireless power transfer systems have been developed. Resonant inductive coupling systems, for instance, can deliver power with efficiencies of 40–60% over air gaps of up to two meters, while directed microwave beaming experiments have demonstrated comparable end-to-end efficiencies over longer distances. These technologies eliminate downtime associated with landing and battery swapping but suffer from critical drawbacks: they require precise alignment between transmitters and receivers, exhibit rapid efficiency degradation as separation increases, and face regulatory constraints on radiated power that limit outdoor deployment [21, 22]. Given these limitations, the present work proposes the concept of an unmanned aerial vehicle carrier (UAVC) [23], which is a "mother ship" that serves as a mobile base for UAV take-off, landing, and recovery. This approach aims to overcome the shortcomings of existing technologies by providing a self-contained platform for sustained UAV operations.

This thesis delivers three major advances toward enabling reliable, fully autonomous UAVC-based operations: two complementary planning frameworks designed for different operational needs, one prioritizing safety guarantees and the other focusing on real-time efficiency, and a robust NMPC controller designed to be paired with the planner suited for real-world deployment. First, it proposes and

implements two complementary trajectory-planning algorithms for a quadrotor to approach and land on the UAVC in unknown environments:

- A *hard-constrained* planner that builds Safe Flight Corridors (SFC) via Jump Point Search (JPS) and convex decomposition, then solves a Mixed-Integer Quadratic Program (MIQP) to enforce collision avoidance and vehicle dynamics as strict constraints.
- A *soft-constrained* planner that combines a kinodynamic A* global search with continuous B-spline refinement under well-designed cost functions, trading absolute feasibility guarantees for real-time performance and smoother trajectories.

These two planners are designed to address different operational demands, forming a comprehensive trajectory planning framework for UAV landing on UAVC in unknown environments. The hard-constrained planner prioritizes strict constraint compliance, ensuring absolute collision avoidance and dynamic feasibility, which makes it indispensable for high-risk scenarios where even minor violations could lead to catastrophic consequences. In contrast, the soft-constrained planner sacrifices absolute feasibility guarantees to achieve higher computational efficiency and smoother trajectories, making it more suitable for common real-world environments where real-time adaptability and operational efficiency are important. Together, they cover a spectrum of operational needs: the hard-constrained approach provides a safety-critical solution for high-risk missions, while the soft-constrained method offers a pragmatic and efficient option for routine scenarios, collectively enhancing the robustness and applicability of the autonomous landing system.

Second, the thesis introduces a novel Nonlinear Model Predictive Control (NMPC) framework that explicitly accounts for the downwash disturbance generated by the UAV's rotors. By integrating a reduced-order aerodynamic disturbance

model into the model predictive control (MPC) prediction horizon, the controller attains robust disturbance rejection, ensuring a stable and precise touchdown on the platform of UAVC.

Finally, the effectiveness and robustness of the entire pipeline are validated through extensive simulation and real-world experiments. In Gazebo, the hard- and soft-constrained planners are compared fairly (and benchmarked against Ego-Planner), revealing their trade-offs in success rate, average speed, and computation time. The faster soft-constrained planner is then selected for hardware deployment, where the NMPC controller achieves satisfactory landing performance and a 68 % reduction in vertical oscillations during real-flight tests.

2 Literature Review

This section systematically reviews existing research on UAV autonomous landing and related technologies, aiming to lay a solid theoretical foundation for the methods proposed in this thesis while identifying critical research gaps to justify the necessity of the current work.

Autonomous landing of UAVs is a complex task that relies on the integration of multiple core components within a system framework. As illustrated in Fig. 1, the key elements of such a framework include perception (for environmental sensing and mapping), planner (for generating collision-free and dynamically feasible paths), and controller (for precise tracking of planned trajectories and maintaining stability under disturbances). While perception technologies are essential for enabling environmental awareness, the present work focuses primarily on the designation of global planner, local planner, and control strategies, which are critical for ensuring reliable and efficient landing on moving aerial carriers, especially in unknown environments.

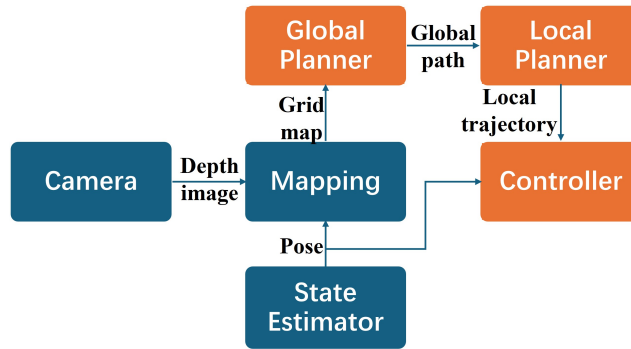


Figure 1: System architecture of the UAV, with the modules highlighted in orange marking the primary focus of our work.

Accordingly, this literature review is structured to comprehensively cover four relevant subsections. In Sec. 2.1, the classification of UAV docking stations is discussed, distinguishing between fixed and mobile docking systems, and elaborating on the characteristics and limitations of different mobile docking systems. Then Sec. 2.2 reviews global path planning methods, including sampling-based approaches and search-based methods, analyzing their advantages, disadvantages, and applicability in UAV navigation. Furthermore, Sec. 2.3 focuses on local trajectory planning, categorizing methods into hard-constrained optimization and soft-constrained optimization. It details the principles, implementation strategies, and performance trade-offs of each category, highlighting how they handle dynamic constraints and obstacle avoidance. Finally, Sec. 2.4 summarizes UAV control strategies, from traditional PID controllers to advanced Model Predictive Control (MPC) frameworks, discussing their effectiveness in handling disturbances and ensuring flight stability.

2.1 UAV Docking Station Typology and Capabilities

UAVs have seen explosive growth in recent years, owing largely to their precise hover capability, vertical take-off and landing, and relatively simple flight controllers. Despite these advantages, their operational range and endurance remain fundamentally constrained by the energy density of onboard lithium-polymer batteries. There are already many solutions with respect to this problem. Building a landing platform is one of the most valuable approaches since the landing platform can be used to transport the UAVs or supply power to the UAVs. Landing platforms can be classified as fixed and mobile UAV docking stations. Fixed UAV docking systems, which allow UAVs to achieve longer flight endurance, have been developed [24–28]. The visual approaches were adopted as the main method to navigate the UAV to find the fixed docking stations. An onboard camera is usually installed on the UAV to

identify the markers on the fixed docking stations as shown in Fig. 2. The UAV lands on the station by using a series of QR codes scaled for different approach altitudes.



Figure 2: An example of the fixed-docking system [28]

However, the fixed UAV docking stations are not flexible enough since they are not movable and may not be able to handle complex situations. As a result, we are mainly focusing on the mobile docking stations, which can be divided into UGV–UAV (Unmanned Ground Vehicle–Unmanned Aerial Vehicle), USV–UAV (Unmanned Surface Vehicle), and airborne UAV–UAV interaction systems [29]. Fig. 3 shows the classification of the current UAV docking systems, where UGV denotes the unmanned ground vehicle, and USV denotes the unmanned surface vehicle.

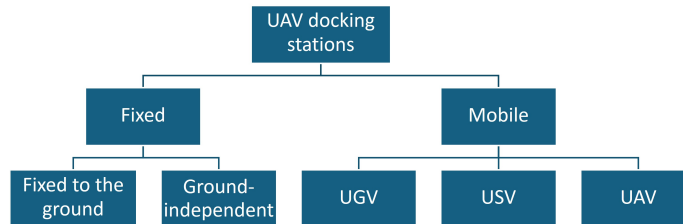


Figure 3: Classification of UAV docking systems

In UGV–UAV systems, the UAV lands directly on a UAV and it is the most common solution due to the simplicity of UGV design. Most of the UGV–UAV systems utilize visual-based trajectory planning approaches for landing [30–36] (see Fig. 4).

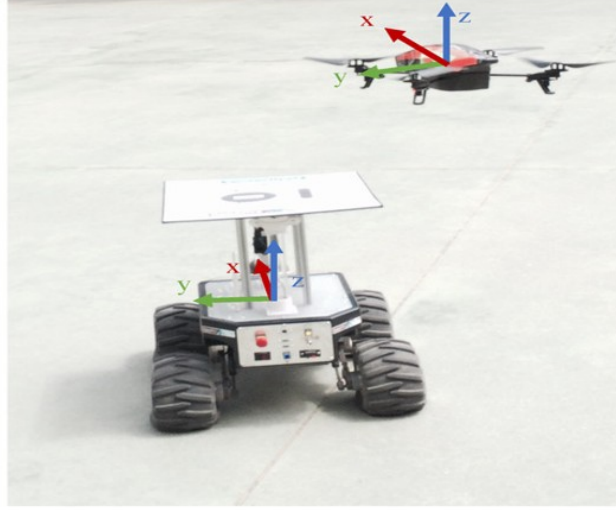


Figure 4: An example of the UGV-UAV system [35]

A visual-inertial odometry algorithm is utilized to estimate the state of the quadrotor [30]. Besides, to detect the UGV which is the landing target, an onboard camera that deals with the unstable visual information was used. However, the UGV–UAV systems will be easily limited when facing complex topography in which it may be difficult for the UAV to arrive quickly. As the second solution, in USV–UAV systems, UAVs are allowed to land on a water surface moving docking stations [37–41]. When UAVs have to be operated above water, it is a suitable solution (see Fig. 5).



Figure 5: An example of the USV–UAV system [39]

A trajectory optimization approach for quadrotors to track moving docking stations using visual feedback from an onboard camera was introduced [38]. Additionally, to ensure the landing safety of the UAV on the USV in open waters, a deep learning-based motion prediction method for the USV was applied. The system may be unstable when applied in severe wave conditions. Besides, the moving scope of the USV is limited to open waters. The UAVC, however, belongs to airborne UAV-UAV interaction systems and is capable of handling more stringent and complex environments compared to other systems such as UGV-UAV and USV-UAV systems. This is because the airborne UAV–UAV interaction systems can ignore many factors such as topography, electricity capacity of the working quadrotors, and human disturbance. Miyazaki et al. present a complete framework enabling a small UAV to intercept and physically dock with a larger, moving aerial platform in midair, opening the door to in-flight payload transfers, cooperative refueling, and multi-vehicle assembly [42]. The leader platform is minimally instrumented with a

passive fiducial marker and an RTK-GPS unit; the follower fuses high-rate visual detections with sporadic GPS updates to estimate the leader's motion over a finite horizon. Exploiting the quadrotor's differential flatness, the follower then computes an end-to-end, energy-optimal polynomial trajectory to a predicted rendezvous point and refines it online in a receding-horizon fashion as fresh measurements arrive. A lightweight, servo-actuated gripper on the follower effects mechanical engagement upon contact as shown in Fig. 6. However, the mechanical design of the gripper is complicated and can only grip one UAV. As a result, the aerial platform system that allows multiple quadrotors to land is more flexible compared to other approaches.



Figure 6: Transportation of working UAV [42]

A two-stage algorithm that generates safe, time-optimal landing trajectories for quadrotors touching down on both static and moving aerial vehicle carriers is proposed in [43]. The first stage applies lossless convexification to reformulate the inherently nonconvex landing-cone constraints and quadrotor dynamics into a convex optimal control problem in three degrees of freedom, yielding an efficient

computation of a nominal safe-landing path. The second stage takes this 3-DoF trajectory as an initial guess and employs sequential convex programming to refine it under the full six-degree-of-freedom quadrotor model, enforcing actuator limits, attitude dynamics, and precise touchdown requirements. While in [44], the authors address the problem of landing a small UAV onto a moving aerial carrier in midair, an operation made challenging by the downwash and unsteady flow interactions between the two vehicles. Although the previous two methods demonstrate precise, time-optimal touchdown on moving aerial platforms [43, 44], both approaches presuppose a fully known obstacle map and off-line trajectory synthesis as they directly incorporate obstacles as constraints within the optimization problem. In reality, however, UAV operations increasingly take place in unstructured, previously unmapped environments such as disaster-response over collapsed buildings, inspections of newly erected infrastructure, or package delivery in urban canyons. In these scenarios obstacles may appear unexpectedly, corridors can narrow without warning, and surface features evolve between flights, making any plan based on a static survey obsolete. A planner that depends on complete prior knowledge cannot react to these unknowns at runtime, nor can it guarantee collision-free landings when the scene diverges from its precomputed model. Motivated by this gap, a real-time trajectory planning and control solution for the UAV landing on an aerial platform in unknown environments remains a pressing need.

2.2 Global Path Planning Methods for UAVs

Global planner is a crucial aspect of robotics and autonomous systems, enabling them to navigate from one point to another efficiently and safely [45–47]. It can be broadly categorized into sampling-based path finding and search-based path finding.

Two prominent methods in this domain are Rapidly-exploring Random Trees

(RRT) [48] and Probabilistic Roadmaps (PRM) [49]. RRT is a common approach that builds a tree progressively by drawing random samples from the search space and extending the tree in the direction of each sample. This method is particularly effective in high-dimensional spaces and can quickly find a feasible path [50]. However, RRT has some limitations, such as high variance in search time and suboptimal path quality. RRT* has been proposed to address these issues. RRT* improves the path quality by continuously optimizing the path as the tree grows. In [51], the authors proposed kinodynamic RRT*, which extends the RRT* algorithm from holonomic robots to systems subject to linear differential constraints, yielding the first sampling-based planner that is both probabilistically complete and asymptotically optimal under kinodynamic rules. Each new sample is connected to its neighbors by solving a fixed-final-state, free-final-time optimal control problem that minimizes a cost combining time and control effort. For systems whose dynamics matrix is nilpotent, these steering problems admit closed-form solutions, so that tree extension and rewiring operations remain nearly as efficient as in geometric RRT*. The other common approach, PRM creates a roadmap by drawing random samples from the configuration space and linking those points into a graph. This graph is then searched to determine a path from the start configuration to the goal. PRM is especially advantageous in static environments because the roadmap can be computed in advance and reused. However, PRM can struggle with narrow passages and requires a large number of samples to ensure connectivity in complex spaces. PRM* has been developed to improve efficiency and path quality. PRM* incorporates optimization techniques to enhance path quality.

On the other hand, search-based path finding methods, like Dijkstra's and A* algorithms [52], use systematic exploration of the search space to find the optimal path. These algorithms are known for their completeness and optimality guarantees, making them suitable for environments where a precise and optimal path is required.

A* algorithm is a widely used pathfinding and graph traversal algorithm that combines the strengths of Dijkstra's algorithm and Greedy Best-First-Search. It uses a heuristic to guide its search, which helps in efficiently finding the shortest path. The algorithm keeps a priority queue of nodes to explore, ranking each node by its accumulated path cost plus an estimated cost from that node to the goal [53]. This estimate is provided by a heuristic function, typically denoted as $h(n)$. The choice of heuristic significantly impacts the performance of A*, with common heuristics including the Manhattan distance and Euclidean distance. A* algorithm guarantees both optimality and completeness. When its heuristic never overestimates the actual cost and respects the triangle inequality, it will always find the shortest path if one exists. One drawback is its large memory requirement, since it must keep every generated node in memory. This can become a serious limitation in very large or complex search spaces. Weighted A* (WA*) has been proposed to solve these issues, allowing for a trade-off between optimality and computational efficiency. By multiplying the heuristic by a factor greater than one, WA* can find solutions faster, though they may not be the shortest paths. This variant is particularly useful in real-time applications where speed is more critical than finding the optimal path.

2.3 Local Trajectory Planning: Hard-constrained and Soft-constrained Optimization

The primary function of the local planner is to transform a global path into actionable movement commands that respect the UAV's kinematics and dynamics. Local planning approaches can be generally classified into hard-constrained and soft-constrained methods.

2.3.1 Hard-constrained Optimization

Hard-constrained optimization in UAV path planning enforces mission requirements, physical limitations, and safety specifications as absolute constraints rather than soft preferences. Every candidate trajectory must strictly satisfy bounds on vehicle dynamics, including maximum velocity, acceleration, and angular velocity. Besides, the environmental restrictions like obstacle avoidance, no-fly zones, and mandated waypoints should also be satisfied strictly. By embedding these requirements directly into the optimization formulation, planners guarantee that any feasible solution is fully compliant with operational and regulatory standards. A minimum-snap trajectory planning method is presented in which trajectories are modeled as piecewise polynomial segments in [54]. The complete trajectory is obtained by formulating and solving a quadratic programming problem that enforces constraints like smoothness, continuity, and dynamic feasibility. Authors in [55] present a hybrid framework that combines sampling-based path extraction with a single convex quadratic program to achieve real-time, dynamically feasible trajectories in densely cluttered workspaces. First, a fast sampling-based search finds a collision-free polyline through the obstacle field. Next, the polyline is lifted into an SFC and a quadratic program enforces velocity, acceleration, and collision-avoidance as hard constraints. This two-stage approach avoids iterative re-planning loops and guarantees strict feasibility in each cycle. Richter et al. address the problem of generating dynamically feasible, collision-free paths for quadrotors operating in cluttered indoor spaces [56]. Exploiting the differential flatness of quadrotor dynamics, the authors represent trajectories as piecewise polynomial splines and formulate an unconstrained quadratic program that jointly optimizes all segment coefficients. By minimizing the snap (fourth derivative) of the trajectory, they ensure smoothness up to high derivatives while enabling efficient sparse computation even for long, multi-segment paths. Deits and

Tedrake apply iterative regional inflation by semidefinite (IRIS) programming to precompute a small set of convex, obstacle-free regions, then use integer variables to assign each polynomial segment to one of these safe regions, ensuring the entire continuous trajectory remains within free space [57]. To enforce collision-freeness over each segment, the method augments the mixed-integer linear or quadratic program with sums-of-squares (SOS) constraints. These SOS constraints certify that every point of a piecewise polynomial lies inside its assigned convex polytope, eliminating “corner-cutting” failures inherent to sample-based checks. The resulting MIQP balances between solver tractability and path fidelity. It uses IRIS-generated regions to drastically reduce the number of binary variables compared to face-based formulations, while SOS adds only polynomial-degree-dependent semidefinite constraints. Modern solvers can find globally optimal or near-optimal solutions with anytime guarantees, making the approach suitable for offline planning and, with warm starts, for fast replanning in dynamic settings.

Similarly, Liu et al. propose a convex-optimization framework that guarantees collision-free, dynamically admissible flight for quadrotors in cluttered indoor spaces [58]. The authors exploit the differential flatness of quadrotor dynamics to represent trajectories as piecewise polynomials and impose safety via the SFC, which is a connected union of convex polyhedra covering free space. By formulating flight within each polyhedron as linear inequality constraints in a single quadratic program, the method simultaneously minimizes snap (fourth derivative of position) and enforces velocity, acceleration, and obstacle-avoidance bounds, yielding globally smooth trajectories without resorting to sampling-based refinements or penalty-based constraints. For real-time applications, Mueller et al. introduce a computationally efficient motion primitive framework that enables fast trajectory generation for quadrotors [59]. By leveraging simplified models and precomputed primitives, the method supports rapid replanning while maintaining dynamic feasibility, making it

well-suited for onboard execution in constrained environments.

For safety under uncertainty, chance-constrained formulations offer probabilistic collision bounds within convex programs, assuming Gaussian uncertainty models [60]. Alternatively, control barrier functions (CBFs) embed forward invariance of safe sets directly into QP constraints, enabling real-time certification of safety for closed-loop systems [61]. Fast replanning in dynamic or partially known environments is addressed by [62], who maintain a safe trajectory inside certified corridors and a fast trajectory exploring new space. Only when certification succeeds is the fast path adopted, achieving safe and responsive updates. Multi-agent extensions introduce hard constraints for UAV separation and coordination. These are enforced using mixed-integer formulations [63], augmented with uncertainty-aware strategies like CBFs and chance constraints for robust deconfliction.

Hard-constrained optimization in UAV path planning enforces mission-critical requirements, physical limitations, and safety rules as non-negotiable constraints that every candidate trajectory must strictly satisfy. These constraints include bounds on UAV's dynamics and environmental restrictions (e.g., obstacle avoidance, no-fly zones, and mandatory waypoints). Key approaches involve modeling trajectories as piecewise polynomial segments, with optimization problems formulated as quadratic programs to ensure smoothness, continuity, and dynamic feasibility. For instance, minimum-snap trajectory planning uses piecewise polynomials and quadratic programming to enforce constraints, while methods leveraging quadrotor differential flatness represent trajectories as splines optimized via unconstrained quadratic programs. SFC, formed by convex decomposition of free space into connected convex polyhedra, further enables embedding obstacle avoidance and dynamic constraints into quadratic programs. MIQP assigns segments to safe regions, with SOS constraints ensuring collision-freeness across entire trajectories. The primary advantage of this approach is its guarantee of strict compliance with operational and safety stan-

dards, ensuring trajectories are always dynamically feasible and collision-free, which is critical for high-risk missions. It also facilitates finding globally optimal solutions in structured environments. However, its disadvantages include high computational complexity, especially for large-scale or unknown environments, as solving MIQP or complex quadratic programs can be time-consuming. Additionally, the rigidity of hard constraints may limit flexibility in rapidly changing scenarios, requiring frequent re-planning that can strain real-time performance.

2.3.2 Soft-constrained Optimization

Soft-constrained optimization in UAV path planning relaxes rigid feasibility requirements by permitting controlled constraint violations in exchange for penalization in the objective, thereby trading hard feasibility guarantees for improved tractability and solution quality. Rather than enforcing all safety, dynamic, and operational limits as hard constraints, the planner introduces slack variables or penalty functions into the optimization formulation. For instance, in [64], the authors proposed Covariant Hamiltonian Optimization for Motion Planning (CHOMP) which treats motion planning as the minimization of a cost functional over the continuous space of robot trajectories. CHOMP formalizes two complementary objectives, including trajectory smoothness and obstacle avoidance, as functionals defined on the path itself, rather than on discretized waypoints. By endowing the infinite-dimensional trajectory space with a Riemannian metric, CHOMP computes covariant functional gradients that are invariant to reparametrization, ensuring that the optimization behaves consistently regardless of time parameterization or spline basis. In [65], the authors proposed Stochastic Trajectory Optimization for Motion Planning (STOMP), which formulates motion planning as a derivative-free, sampling-based optimization over the continuous space of robot trajectories. Because no exact gradient is required,

STOMP naturally overcomes local minima that entrap gradient-based planners like CHOMP, enabling it to refine initially infeasible paths into smooth, collision-free motions. Continuous-time trajectory optimization for online UAV replanning formulates multirotor collision avoidance as a continuous-time optimal control problem solved in real time onboard the vehicle [66]. The authors represent the vehicle's path as a piecewise-polynomial spline parameterized by control points, and they embed both smoothness and obstacle-avoidance costs, drawn from a precomputed Euclidean signed-distance field, as integrals over the continuous trajectory. This avoids discretization artifacts, ensures dynamic feasibility through explicit derivative constraints, and yields a nonconvex cost functional that is minimized via gradient-based updates in a Hilbert-space metric, analogous to CHOMP but tailored to UAV kinematics and onboard computation. While in [67], uniform B-splines are employed to produce continuous, smooth trajectories that allow for local adjustments to satisfy kinodynamic constraints. These methods incorporate gradient-based information to steer trajectories away from obstacles by embedding penalty functions within the optimization framework. Although this strategy encourages safer distances from potential hazards, it remains vulnerable to convergence issues such as local minima and does not provide robust guarantees regarding overall success rate or kinodynamic feasibility.

In addition to the methods described in [67], other research has explored soft-constrained optimization in the context of trajectory generation and kinodynamic planning. Robust and agile quadrotor flight is demonstrated in [68], where gradient-based refinement minimized collision costs under speed constraints. Matrix-based B-spline representations [69] further enhanced control point optimization. Gao et al. propose an onboard receding-horizon planner that directly optimizes piecewise B-spline trajectories using functional gradients [70]. By embedding the environment in an Euclidean signed-distance field, they derive smooth repulsive costs and

combine these with snap and path-tracking penalties into a single objective. A few unconstrained gradient-descent iterations per control cycle update the spline's control points in millisecond time, and derivative bounds on the spline automatically enforce velocity, acceleration and jerk limits while guaranteeing collision avoidance. To reduce planning latency, [71] develops Ego-Planner, eliminating Euclidean Signed Distance Field (ESDF) construction by directly optimizing differentiable obstacle costs over short horizons.

Soft-constrained optimization in UAV path planning relaxes rigid feasibility requirements by allowing controlled constraint violations, which are penalized in the objective function to balance tractability and solution quality. Instead of enforcing all constraints strictly, it uses slack variables or penalty functions to handle safety, dynamic, and operational limits. Notable methods include CHOMP, which optimizes trajectory smoothness and obstacle avoidance via reparametrization-invariant gradients, and STOMP, a derivative-free sampling-based method that refines infeasible paths into smooth, collision-free trajectories. Continuous-time trajectory optimization, using piecewise-polynomial splines with integral-based costs for smoothness and obstacle avoidance, is also widely applied, along with weighted sum approaches and fast optimization algorithms for handling soft constraints. The main advantage of soft-constrained optimization is its improved computational efficiency and flexibility, making it suitable for real-time trajectory replanning in unknown environments. It avoids the computational burden of strict constraint enforcement, enabling faster convergence. However, its key disadvantages include the lack of guarantees for absolute feasibility and constraint violations may occur even when avoidable. Besides, selecting appropriate penalty weights is also a challenge, which can lead to suboptimal solutions or convergence issues. Additionally, it is more vulnerable to local minima, reducing reliability in complex environments where strict safety is paramount.

2.4 UAV Control Strategies: From Traditional Controllers to Advanced MPC

Proportional-Integral-Derivative (PID) controllers remain a staple in multirotor UAV regulation due to their simplicity and reliability. Multirotor models paired with PID controllers are introduced in [72, 73]. The traditional PID controllers are proposed based on the transfer function between the Euler angles and the control signal. However, the linear PID approach exhibits limitations when faced with aggressive flight, significant payload variation, or environmental disturbances. Hence, controllers designed to enable agile maneuvers in multirotor systems were introduced in [74–76]. To compensate for these aerodynamic disturbances, [74] proposes a novel feedback-linearizing control architecture that integrates explicit estimates of flapping-induced moments and thrust deviations. Rather than treating aerodynamic contributions as unmodeled uncertainty, the controller uses real-time rotor speed and vehicle attitude measurements to predict and counteract perturbations before they degrade performance. In [75], the authors propose an intuitive, data-driven method for executing and improving aggressive aerial acrobatics without relying on complex trajectory tracking or full nonlinear inversion. While [76] proposes a hybrid framework that combines model-based trajectory synthesis with experiment-driven parameter tuning to achieve acrobatic quadrotor flight on computationally limited hardware. Using the differential flatness of quadrotor dynamics, trajectories are constructed as sequences of polynomial segments whose boundary conditions specify positions, orientations and their derivatives. For each segment a feedforward term derived from the flat-output mapping is paired with a cascaded feedback controller to enforce dynamic feasibility and ensure smooth tracking.

MPC is increasingly becoming a preferred method for the control of UAVs due to its advanced capabilities in handling multi-variable control problems and

constraints [77–79]. Unlike traditional PID controllers, which are reactive and correct states based on tracking error, MPC takes a proactive approach by predicting future system behavior and optimizing control actions accordingly. Generally, by solving constrained optimization problems, MPC minimizes the tracking error in the predicted time horizon [80]. However, implementing MPC on the complex nonlinear dynamics of a quadrotor exceeds the computational capabilities of the standard embedded hardware, especially for early-age UAVs. As a result, feedback linearisation was applied to transform the system into a linear form, enabling Linear MPC (LMPC) to handle force-controlled position and trajectory tracking [81].

Since LMPC is mainly based on small-angle assumptions or used for position control, it may not be able to represent the rotational dynamics that are inherently nonlinear. Thus, when facing aggressive trajectory tracking, the NMPC shows better capability in tracking an aggressive polynomial trajectory [82]. A comprehensive survey by Nguyen et al. [83] synthesizes these advancements, noting that modern NMPC methods for UAVs increasingly focus on balancing model complexity, computational speed, and robustness. Their work underscored the growing role of NMPC in enabling UAVs to perform intricate tasks, ranging from precision landing to navigation in obstacle-rich environments, by dynamically adapting to changing conditions while enforcing strict constraints. Neunert et al. formulate trajectory planning and feedback control as a single optimal control problem solved in real time on-board agile robots [84]. The method employs the Sequential Linear Quadratic algorithm in a receding-horizon framework to compute both feedforward control sequences and time-varying feedback gains over multi-second horizons within only a few milliseconds. By iteratively linearizing the full nonlinear dynamics around the current trajectory and performing a backward-forward pass to update control laws, the scheme maintains stability and constraint satisfaction without resorting to separate planning and tracking layers. Kamel et al. formulate the attitude error

using a geometric metric on rotation matrices and embed it within a finite-horizon NMPC problem that enforces state and input constraints arising from the vehicle's physical limits [85]. Solved in a receding-horizon fashion, the controller computes the optimal motor thrust commands at each timestep, explicitly accounts for actuator allocation and disturbance rejection, and maintains stability even when one propeller fails. The NMPC is integrated into a cascaded control architecture in which a conventional outer-loop position controller generates a desired thrust vector and heading, while the inner-loop optimizer tracks that reference on $SO(3)$. The authors in [86] address safe motion planning for quadrotors operating under both model uncertainty and external disturbances. The authors begin by identifying a second-order closed-loop model of the vehicle via an online maximum-likelihood estimation of autopilot inner-loop gains and thrust coefficients. They then propagate uncertainty in model parameters and initial state as high-confidence ellipsoids in pose space and formulate an NMPC problem that explicitly penalizes ellipsoid penetration into cylindrical obstacles. Control inputs are chosen to minimize control effort while guaranteeing collision avoidance with a user-specified safety margin around each obstacle. Foehn et al. explore time-optimal waypoint flight for quadrotors, demonstrating how NMPC can generate trajectories that minimize travel time while respecting dynamic constraints and highlighting its utility in high-agility missions [87]. Unified control frameworks that integrate both position and attitude regulation have further expanded NMPC's applicability. Murilo and Lopes propose a unified NMPC framework for vertical take-off and landing (VTOL) UAVs, enabling simultaneous control of position and attitude within a single optimization loop [88]. This integration reduced cross-coupling errors and improved overall stability, particularly in transition phases between hover and forward flight.

Besides adopting LMPC to simplify the dynamic model and reduce computation burden, another common method is to divide the flight controllers into high-level

position control and low-level attitude control [80]. While separating controls can simplify the tracking problem, it may risk misrepresenting the quadrotor's true performance potential, leading to suboptimal or unsafe flight behaviors. For agile and accurate flight, especially in complex environments, the high-level controller should have a realistic understanding of the quadrotor's physical constraints and capabilities. Thanks to advances in computing power and nonlinear optimization solvers, it is feasible to run NMPC with a full nonlinear dynamic model on an on-board computer, enabling quadrotors to fully exploit their maneuvering capabilities. Bicego et al. propose an NMPC approach with an enhanced actuator model tailored for multi-rotor aerial vehicles with generic designs [89]. By refining the representation of actuator dynamics, their method improved control accuracy and robustness, ensuring better alignment between planned trajectories and actual vehicle behavior, which is critical for complex maneuvers where actuator limits and nonlinearities dominate performance. Torrente et al. extract intermediate states from the NMPC solution and forward them to a low-level controller [90]. Small et al. present an embedded NMPC approach for aerial navigation in cluttered spaces, optimizing trajectories in real time while ensuring collision avoidance [91]. By prioritizing computational efficiency, their method enabled deployment on resource-constrained UAV platforms, a key step toward practical autonomous navigation in unknown environments.

In summary, UAV control initially relies on cascaded PID loops for attitude and position regulation, prized for simplicity and real-time performance. However, pure linear PID schemes struggle with aggressive maneuvers, large payload changes, and unmodeled aerodynamic disturbances. To overcome these limits, researchers have developed feedback-linearizing controllers that estimate and cancel flapping-induced moments in real time, data-driven tuning methods for acrobatic maneuvers, and hybrid frameworks that pair polynomial trajectory synthesis with experiment-derived feedback gains.

MPC improves performance by forecasting future vehicle behavior and solving an optimization at each time step under hard bounds on states and inputs. LMPC variants exploit small-angle or linearized dynamics for onboard feasibility but cannot fully capture the quadrotor's nonlinear attitude response during aggressive flight. Advances in NMPC now run on embedded computers, incorporating the full 6-DoF equations of motion, actuator limits, and aerodynamic disturbances into a receding-horizon solver. NMPC implementations use sequential quadratic programming or sequential linear-quadratic methods to compute both feedforward trajectories and time-varying feedback gains, guaranteeing stability even under propeller failure or external wind gusts. Cascaded architectures often delegate high-level position references to an outer MPC loop while a low-level attitude controller tracks $SO(3)$ setpoints, but monolithic NMPC on modern hardware can fully exploit the vehicle's maneuvering envelope for high-accuracy trajectory tracking and robust disturbance rejection. However, most MPC frameworks ignore downwash interactions during the final landing approach, resulting in significant landing errors when UAVs land on aerial platforms.

2.5 Summary

This section presents a comprehensive literature review focusing on UAV autonomous landing and related technologies, aiming to establish a theoretical foundation for the proposed methods while identifying critical research gaps. The review is structured into four key subsections.

First, it discusses UAV docking station typology and capabilities, categorizing stations into fixed and mobile systems. Fixed docking stations, though useful for extending endurance, lack flexibility, while mobile systems include UGV-UAV, USV-UAV, and airborne UAV-UAV interaction systems. UAVC are highlighted for

their ability to handle complex environments, but existing research often relies on pre-known obstacle maps or offline trajectory synthesis, limiting their applicability in unknown settings. Next, global path planning methods for UAVs are reviewed, distinguishing between sampling-based approaches (e.g., RRT, RRT*, PRM, PRM*) and search-based methods (e.g., Dijkstra, A*, weighted A*). Sampling-based methods are efficient in high-dimensional spaces but may produce suboptimal paths, while search-based methods guarantee optimality and completeness but require significant memory, posing challenges in large or complex environments.

The review then focuses on local trajectory planning, dividing methods into hard-constrained and soft-constrained optimization. Hard-constrained methods enforce strict compliance with dynamic constraints (e.g., velocity, acceleration) and obstacle avoidance via piecewise polynomials and quadratic programming, ensuring feasibility in high-risk scenarios but with high computational complexity. Soft-constrained methods, by contrast, allow controlled constraint violations penalized in cost functions (e.g., CHOMP, STOMP), improving efficiency and flexibility but lacking absolute feasibility guarantees. Finally, UAV control strategies are summarized, from traditional PID controllers, which are simple but limited in handling aggressive maneuvers or disturbances, to advanced MPC frameworks. NMPC is emphasized for its ability to optimize control actions over a prediction horizon, effectively handling nonlinear dynamics and disturbances, making it suitable for precise landing and complex navigation. However, existing methods struggle with unknown environments and dynamic disturbances like downwash winds.

Overall, the literature review identifies gaps in addressing unknown environments, real-time adaptability, and dynamic disturbances, underscoring the need for the trajectory planning and control frameworks proposed in the thesis.

2.6 Notation

Table 1: Key Parameters Mentioned in the Later Sections and Their Meanings

Notation	Meaning
v_{max}	Maximum velocity of the UAV
a_{max}	Maximum acceleration of the UAV
j_{max}	Maximum jerk of the UAV
r_s	Safety radius ensuring UAV can stop in bounding box
ω_1	Weight of smoothness cost in soft-constrained planner
ω_2	Weight of collision cost in soft-constrained planner
ω_3	Weight of dynamic feasibility cost in soft-constrained planner
$\mathcal{A}, \mathcal{B}$	Parameters modulating dynamic feasibility cost function
$\mathbf{R}_{WB}$	Rotation matrix from body frame (B) to world frame (W)
$\boldsymbol{\omega}$	Angular velocity in body frame
$\mathbf{J}$	Inertia matrix of the UAV
$\mathbf{T}$	Total thrust generated by the rotors
$\mathbf{F}_\sigma$	External disturbance force
$\mathbf{g}_W$	Gravitational acceleration in world frame
m	Mass of the UAV
A_ϕ, A_θ	Control gains for roll and pitch inner-loop
$\lambda_\phi, \lambda_\theta$	Time constants for roll and pitch inner-loop control systems
F_{aero}	Thrust produced by downwash wind disturbance
ρ	Air density
C_d	Thrust coefficient
A_{eff}	Effective area of wind disturbance
V_d	Air speed acting on the UAVC platform
Q, R	Coefficients in the downwash wind speed model
$\mathbf{x}$	System state vector
$\mathbf{u}$	Control input vector
$\mathbf{Q}$	State error weighting matrix in NMPC
$\mathbf{R}$	Control effort weighting matrix in NMPC
$\mathbf{W}$	Terminal state weighting matrix in NMPC

3 Hard-constrained Optimization with Safety Flight Corridor

This section focuses on developing a trajectory planning method with strict constraints to ensure collision avoidance and dynamic feasibility for UAV landing. In Sec. 3.1, a global planner based on JPS and pruning rules to identify optimal paths is introduced. Then Sec. 3.2 details convex decomposition techniques to construct SFCs. In Sec. 3.3, a local planner is proposed which formulates an MIQP to generate dynamically feasible and strictly compliant trajectories.

3.1 Global Planner via Jump Point Search

JPS is adopted as the main approach in the hard-constrained optimization to find the global path from the initial to the goal position. Compared to the traditional A* algorithm, the solving speed of JPS is much faster by an order of magnitude and more [92]. Besides, JPS can still ensure the completeness and optimality of the finding path, which means that the path is always the shortest. Similar to the A* algorithm, JPS belongs to the search-based methods. Unlike traditional pathfinding methods that rely heavily on prior map knowledge, JPS is inherently adaptive to dynamically changing unknown environments, primarily due to its efficient pruning mechanism and jump point identification, which enable rapid response to newly detected obstacles. The basic application requirement of JPS is that the map should be a uniform-cost grid. 2-dimensional cases will be mainly considered because JPS relies on pruning rules and jump point identification logic that have strong symmetry

between 2D and 3D grid environments. The core mechanisms, such as eliminating redundant neighbor nodes, distinguishing natural and forced neighbors, and leaping to jump points, are rooted in geometric relationships that remain consistent across dimensions. Secondly, the 2D case enhances readability and intuitive understanding. Visualizing JPS in 2D helps readers easily grasp how JPS reduces computational overhead. A 3D case would involve more complex neighbor configurations (26 directions) and require extra figures and equations to explain, increasing cognitive load without improving understanding of JPS's role.

3.1.1 Pruning Rules

This section will introduce a set of pruning rules to determine whether a node should be generated or skipped. In other words, a series of nodes that are adjacent to some node q from the grid will be pruned. In unknown scenarios, the pruning rules of JPS play a critical role in reducing unnecessary computational overhead. By systematically eliminating redundant neighbor nodes (pruned nodes) that do not contribute to shorter paths, JPS significantly reduces the number of nodes that need to be explored. As shown in Fig. 7, node q has 8 neighbor nodes, and node q can get to them by a straight move or a diagonal move. Note that a straight move costs 1, while a diagonal move costs $\sqrt{2}$. node q can only be expanded by its parent node $p(q)$ and all the four situations are demonstrated here in the 2-dimensional case due to the symmetry. The black blocks represent the obstacles that the node q can not traverse, and the gray blocks are the pruned nodes. For the notation, $\pi = \langle n_0, n_1, \dots, n_k \rangle$ represents a path which starts at node n_0 and ends at node n_k . The function len measures the length of a path on the grid. Moreover, the function $y = q + k\vec{d}$ indicates that node y is reached by taking k unit moves from node q in direction $\vec{d}$.

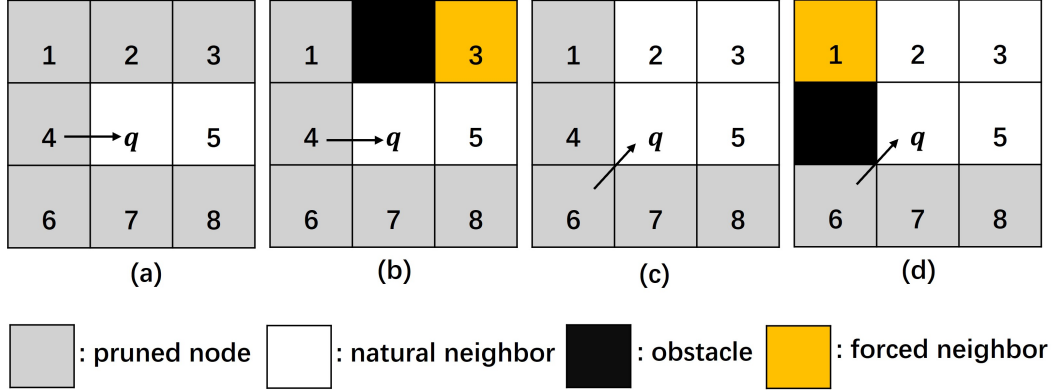


Figure 7: 2-dimensional pruning case. Four situations in the 2D pruning case are illustrated in which node q is reached from its parent $p(q)$ by either a straight or a diagonal move. When q is expanded, all of the nodes shaded in grey are pruned from consideration.

To find out the pruned nodes, the lengths of two paths should be compared, including path π and path π' . Path π starts with parent node $p(q)$, goes through node q and ends with ending node n and the other path π' also starts at node π and ends with n but will not visit node q . In general, two kinds of movement should be considered, which are straight moves and diagonal moves. For the straight moves, when the following constraint is satisfied,

$$\text{len}(\langle p(q), \dots, n \rangle \setminus q) \leq \text{len}(\langle p(q), q, n \rangle), \quad (1)$$

then node $n \in \text{neighbours}(q)$ should be pruned. In other words, when finding the shortest path from parent node $p(q)$ to node n , node n will be pruned if the length of the path π' is less or equal to the length of the path π . As shown in Fig. 7(a), when parent node $p(q) = 4$, node $\{1, 2, 3, 4, 6, 7, 8\}$ will be pruned. For the diagonal moves, similar to pruning rules that have been defined in the straight moves, when the following constraint is satisfied,

$$\text{len}(\langle p(q), \dots, n \rangle \setminus q) < \text{len}(\langle p(q), q, n \rangle), \quad (2)$$

node $n \in neighbours(q)$ should be pruned. The only difference between these two situations is that path π' must be strictly dominant. In Fig. 7(c), node $\{1, 2, 7, 8\}$ will be pruned when parent node $p(q) = 4$.

Neither of the situations mentioned considers obstacles around the node q , and the remained nodes after pruning will be called *natural neighbors* of q . However, some non-natural neighbors may not be pruned due to the existence of obstacles around the node q , and those non-nature neighbors are called *forced neighbors*. When the following constraint is satisfied:

- n is a non-neighbor node of q
- $len(\langle p(q), q, n \rangle) < len(\langle p(q), \dots, n \rangle \setminus q)$,

then node $n \in forced\ neighbours(x)$. As shown in Fig. 7(b), node 3 will be the forced neighbor compared to Fig. 7(a). Similarly, in Fig. 7(d), node 1 will be the forced neighbor. The identification of forced neighbors in JPS is key to handling unexpected obstacles in unknown environments. Forced neighbors arise when obstacles block the original optimal path, prompting the algorithm to explore alternative directions that would otherwise be pruned. In unknown environments, where obstacles may appear without warning, this mechanism ensures that the planner can dynamically adjust the path direction without re-exploring the entire grid.

3.1.2 Jump Point Algorithm

Firstly, the concept of a jump point is defined as follows. Node y will be a jump point from node q if value k is minimum in $y = q + k\vec{d}$ and one of the below conditions is satisfied:

- **Condition 1:** Node y is the target point.

- **Condition 2:** There is at least one *forced neighbor* of node y .
- **Condition 3:** $\vec{d}$ is a diagonal move and there should be another node $z = y + k_i \vec{d}_i$ such that z is a jump point from y by condition 1 or condition 2. Noting that $z = y + k_i \vec{d}_i$ denotes that the location of node z is derived by moving node y in $k_i \in N$ steps with direction $\vec{d}_i$.

Based on the definition of the jump point, Algorithm 1 for identifying individual jump point successors can be derived [92].

Algorithm 1 Function `jump`

Require: q : initial node, $\vec{d}$: direction, s : start node, g : goal node

Ensure: the next jump point from q along $\vec{d}$, or null if none exists

```

1:  $n \leftarrow \text{step}(q, \vec{d})$ 
2: if isInvalid( $n$ ) then
3:   return null
4: end if
5: if isGoal( $n, g$ ) then
6:   return  $n$ 
7: end if
8: if hasForcedNeighbor( $n$ ) then
9:   return  $n$ 
10: end if
11: if isDiagonal( $\vec{d}$ ) then
12:   for all  $\vec{d}_i$  in splitDiagonal( $\vec{d}$ ) do
13:     if jump( $n, \vec{d}_i, s, g$ )  $\neq$  null then
14:       return  $n$ 
15:     end if
16:   end for
17: end if
18: return jump( $n, \vec{d}, s, g$ )

```

The jump point algorithm further enhances JPS's suitability for unknown environments by enabling long-distance leaps through unobstructed regions. Instead of checking each individual node along a straight or diagonal direction, JPS identifies

jump points which are critical nodes where the path direction must change due to obstacles or target proximity. In unknown environments, this allows the planner to skip large unobstructed areas which are initially unknown but later confirmed safe via real-time perception and directly focus on key decision points. This not only accelerates path computation but also ensures that the generated path remains feasible even as new environmental details such as previously unseen open spaces or hidden obstacles are revealed during flight.

3.2 Convex Decomposition

Piece-wise linear paths can be generated by applying JPS. A convex polyhedron around each line segment of the paths can be obtained via convex decomposition. These convex polyhedrons make up the SFC. $P = \langle p_0 \rightarrow p_1 \rightarrow \dots \rightarrow p_n \rangle$ denotes the piece-wise linear path from node p_0 to p_n . O denotes the set of points that make up the obstacles. There are four steps to generate the SFC, including finding ellipsoids, finding polyhedra, and adding bounding boxes. Details about each step will be introduced in the following sections.

3.2.1 Find Ellipsoids

An ellipsoid that contains the path segment L will be found in this step. In addition, the generated ellipsoid will not contain any points in set O . An ellipsoid $\Gamma \subseteq \mathbb{R}^3$ can be denoted as [58]:

$$\Gamma(\mathbf{E}, \mathbf{d}) = \{\mathbf{p} = \mathbf{E}\bar{\mathbf{p}} + \mathbf{d} \mid \|\bar{\mathbf{p}}\| \leq 1\}, \quad (3)$$

$\{\|\bar{\mathbf{p}}\| \leq 1\}$ represents a sphere that will be deformed to the ellipsoid and $\mathbf{p} \in \mathbb{R}^3$. $\mathbf{E} = \mathbf{R}^T \mathbf{D} \mathbf{R}$ constitutes a 3×3 symmetric positive-definite matrix that describes the linear map deforming a unit sphere into an ellipsoid. In this expression, $\mathbf{R} \in \mathbb{R}^3$

is an orthogonal rotation matrix aligning the ellipsoid with the coordinate axes, and $\mathbf{D} = \text{diag}(a, b, c) \in \mathbb{R}^3$ scales the sphere into an ellipsoid with semi-axes a, b, c . $\mathbf{d} \in \mathbb{R}^3$ indicates the center of the ellipsoid. As a result, the problem of finding ellipsoids is transformed to finding feasible $\mathbf{E}, \mathbf{d}$ when the obstacles O and path segment L are known.

Denote the three axes in the initial sphere as $\{x, y, z\}$, and the direction of axis x is consistent with that of path line L . Additionally, the center of the initial sphere is located at the midpoint of L as shown on the left of Fig. 8, and its diameter matches the length of L . To form the ellipsoid, the sphere is first contracted along the y axis to produce the maximal ellipsoid with two equal axes as shown in the middle of Fig. 8. Then this intermediate ellipsoid is further contracted along the z axis to yield the final ellipsoid similar to the previous step. The final ellipsoid is obtained when there are no points of obstacles inside the ellipsoid as shown on the right of Fig. 8.

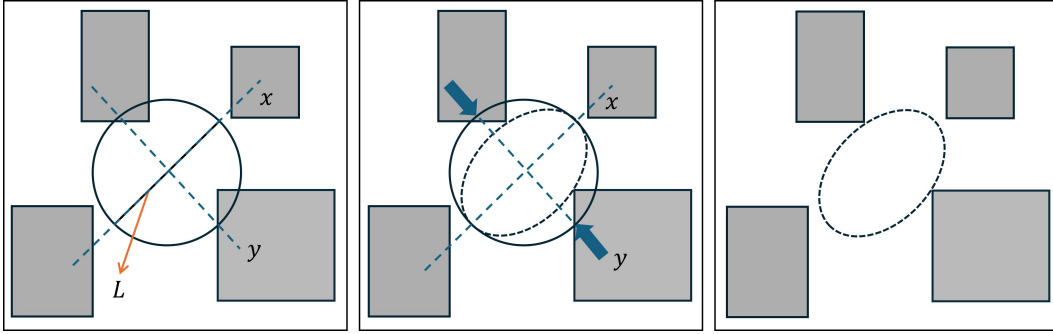


Figure 8: Ellipsoid generation around a path segment. A bounding sphere whose diameter matches the segment length is first contracted along y axis to remove nearby obstacles, then further contracted along z axis to yield a maximal ellipsoid enclosing the segment without intersecting any obstacles. The blue arrow indicates the contraction direction.

3.2.2 Find Polyhedron

Γ_0 denotes the generated ellipsoid in the last section, which contacts an obstacle at point $\mathbf{P}_0^c = \mathbf{P}^*$. At the point $\mathbf{P}_0^c$, a plane which is tangent to the ellipsoid can be found to generate a half space $\mathcal{S}_0 = \{ \mathbf{P} \in \mathbb{R}^3 \mid \mathbf{M}_0^T \mathbf{P} < h_0 \} \subseteq \mathbb{R}^3$. The parameter $\mathbf{M}_0 \in \mathbb{R}^3$ and $h_0 \in \mathbb{R}$ can be calculated as follows:

$$\begin{aligned} \mathbf{M}_0 &= \left. \frac{d\Gamma_0}{d\mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_0^c} = 2\mathbf{E}^{-1}\mathbf{E}^{-T}(\mathbf{p}_0^c - \mathbf{d}) \\ h_0 &= \mathbf{M}_0^T \mathbf{p}_0^c. \end{aligned} \tag{4}$$

After calculating $\mathcal{S}_0$, all the regions of obstacles in $\mathcal{O}$ which are outside $\mathcal{S}_0$ will be removed as shown on the left in Fig. 9. Then the ellipsoid should be expanded until it touches another obstacle point as shown in the middle of Fig. 9, which is $\mathbf{P}_1^c$. The new ellipsoid obtained is denoted as Γ_1 , and the latest half space is $\mathcal{S}_1$. Then all the regions of obstacles in $\mathcal{O}$ which are outside $\mathcal{S}_1$ will be removed as shown on the middle of Fig. 9. The process mentioned before should be implemented recursively to find the final polyhedron to get a series of half-spaces, which are $\{\mathcal{S}_0, \mathcal{S}_1, \dots, \mathcal{S}_n\}$.

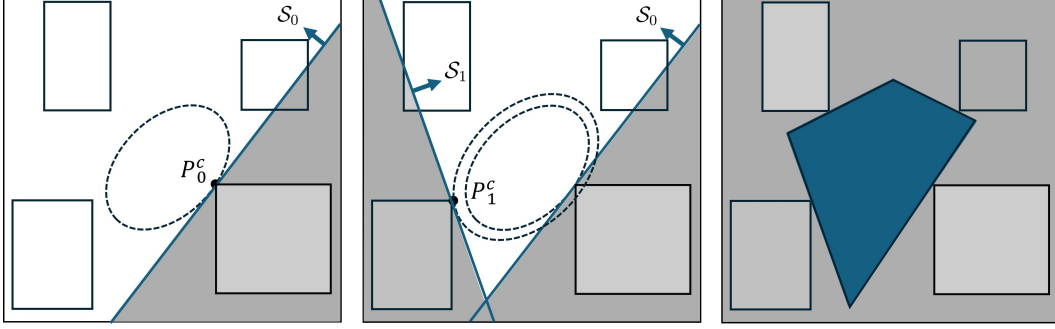


Figure 9: Process of constructing the convex polyhedron. At each iteration, the ellipsoid enclosing the path segment is grown until it first contacts an obstacle point. A tangent plane at that contact defines a half-space whose shaded region removes that obstacle from the free space. Repeating these ellipsoid expansions and half-space intersections yields a convex polyhedron that tightly encloses the segment while excluding all obstacles.

3.2.3 Bounding Box

Although the earlier approaches can generate polyhedra, they become costly because each point in O must be tested at least twice for intersection with the expanded ellipsoid. To cut down on these checks, we enclose the line segment L in a bounding box, which limits the candidate points for obstacle testing. This strategy both speeds up computation and keeps the resulting trajectory close to the original path. The bounding box is formed by six rectangles aligned with L . To guarantee a safe buffer from any obstacle, each face of the box is placed at a distance of at least r_s from the segment. The safety radius is calculated based on the maximum speed and acceleration of the MAV, ensuring the bounding box is large enough to guarantee safety and the UAV can stop within the bounding box during emergencies. Assume the maximum speed and acceleration of the UAV are v_{max} , a_{max} , the safety radius

should satisfy the following condition:

$$r_s \geq v_{max}^2 / 2a_{max}. \quad (5)$$

As shown on the left in Fig. 10, a bounding box is generated around each line segment. By iteratively applying this method, a series of bounding boxes can be generated as shown on the right in Fig. 10. The green area denotes the area of the bounding boxes.

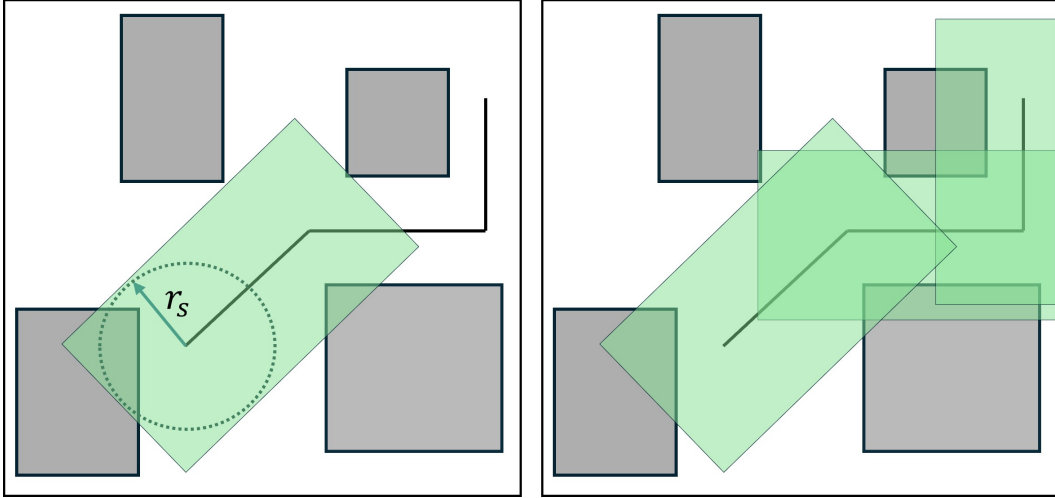


Figure 10: Process of formulating the bounding boxes. Left: A single path segment L is first enclosed in the minimal axis-aligned box and then uniformly expanded by the safety radius r_s to ensure sufficient buffer for dynamic stopping. Right: Repeating this for every segment produces a connected chain of overlapping bounding boxes (greened).

3.3 Local Planner via Mixed-Integer Quadratic Programming

The quadrotor can be modeled as:

$$\mathbf{x}^T = [\mathbf{x}^T \dot{\mathbf{x}}^T \ddot{\mathbf{x}}^T] = [\mathbf{x}^T \mathbf{v}^T \mathbf{a}^T], \quad (6)$$

and the control input is :

$$\mathbf{u} = \ddot{\mathbf{x}} = \mathbf{j}, \quad (7)$$

in which $\mathbf{x} \in \mathbb{R}^{9 \times 1}$ is the quadrotor's state vector, $\mathbf{x}, \mathbf{v}, \mathbf{a}, \mathbf{j} \in \mathbb{R}^{3 \times 1}$ denote its position, velocity, acceleration, and jerk, respectively. The trajectory can be divided into N intervals in the optimization problem solved by the local planner as shown in Fig. 11. The time intervals allocated to each trajectory interval are the same, which is $\tau \in [0, dt]$. n denotes the sequence of the trajectory interval, from $n = 0$ to $n = N - 1$. Besides, p denotes the polyhedron allocations, from $p = 0$ to $p = P - 1$.

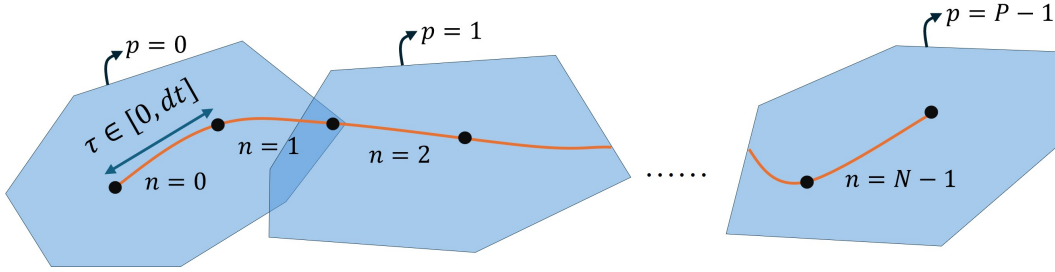


Figure 11: Visualization of the local-planner trajectory parameterization. The continuous path is divided into N successive intervals of equal duration dt , each described by a cubic polynomial with four control points, and each polynomial segment is assigned (via binary variables B_{np}) to one of the convex polyhedral cells in the SFC to enforce collision and dynamic constraints.

Assuming $\mathbf{j}(t)$ is constant in each trajectory interval from $n = 0$ to $n = N - 1$, the planned trajectory can be a spline line which is composed of third-degree polynomials stated as follows:

$$\mathbf{x}_n(\tau) = \alpha_n \tau^3 + \beta_n \tau^2 + \gamma_n \tau + \delta_n, \quad \tau \in [0, dt]. \quad (8)$$

According to the concept of Bezier Curves, the trajectory can also be expressed as

follows:

$$\mathbf{x}_n(\tau) = \sum_{i=0}^3 \binom{3}{i} \left(1 - \frac{\tau}{dt}\right)^{3-i} \left(\frac{\tau}{dt}\right)^i \mathbf{C}_{ni}, \quad \tau \in [0, dt]. \quad (9)$$

By solving (8)(9), four control points $\mathbf{C}_{ni} \in \mathbb{R}^3$ in each interval n can be obtained as follows:

$$\begin{aligned} \mathbf{C}_{n0} &= \boldsymbol{\delta}_n, \\ \mathbf{C}_{n1} &= \frac{\boldsymbol{\gamma}_n dt + 3\boldsymbol{\delta}_n}{3}, \\ \mathbf{C}_{n2} &= \frac{\boldsymbol{\beta}_n dt^2 + 2\boldsymbol{\gamma}_n dt + 3\boldsymbol{\delta}_n}{3}, \\ \mathbf{C}_{n3} &= \boldsymbol{\alpha}_n dt^3 + \boldsymbol{\beta}_n dt^2 + \boldsymbol{\gamma}_n dt + \boldsymbol{\delta}_n. \end{aligned} \quad (10)$$

An important property of Bezier Curves is the convex hull, which can be applied to constrain the generated curves inside polyhedra. The convex hull property states that a Bezier curve will always lie within the convex hull of its control points. As a result, the trajectory will be contained in the polyhedron generated before completely if all the control points $\mathbf{C}_{nj}$ of an interval n are in the same polyhedron. Here, the polyhedron can be denoted as $\{\mathbf{M}_p, h_p\}$, $p = 0 : P - 1$. Based on this property, binary variables B_{np} can be adopted to help the optimizer determine the specific interval allocation. B_{np} denotes that the interval n is inside polyhedron p . As a result, an MIQP problem can be developed for the desired trajectories in each replanning step as shown below:

$$\min_{\mathbf{j}_n, B_{np}} \sum_{n=0}^{N-1} \|\mathbf{j}_n\|^2 dt \quad (11a)$$

$$\text{s.t. } \mathbf{x}_n(\tau) = \boldsymbol{\alpha}_n \tau^3 + \boldsymbol{\beta}_n \tau^2 + \boldsymbol{\gamma}_n \tau + \boldsymbol{\delta}_n \quad \forall n \quad \forall \tau \in [0, dt] \quad (11b)$$

$$\mathbf{v}_n(\tau) = \dot{\mathbf{x}}_n(\tau) \quad \forall n \quad \forall \tau \in [0, dt] \quad (11c)$$

$$\mathbf{a}_n(\tau) = \dot{\mathbf{v}}_n(\tau) \quad \forall n \quad \forall \tau \in [0, dt] \quad (11d)$$

$$B_{np} = 1 \Rightarrow \begin{cases} \mathbf{M}_p^T \mathbf{C}_{n0} \leq \mathbf{h}_p \\ \mathbf{M}_p^T \mathbf{C}_{n1} \leq \mathbf{h}_p \\ \mathbf{M}_p^T \mathbf{C}_{n2} \leq \mathbf{h}_p \\ \mathbf{M}_p^T \mathbf{C}_{n3} \leq \mathbf{h}_p \end{cases} \quad \forall n \quad \forall p \quad (11e)$$

$$\sum_{p=0}^{P-1} B_{np} \geq 1 \quad \forall n \quad (11f)$$

$$B_{np} \in \{0, 1\} \quad \forall n \quad \forall p \quad (11g)$$

$$\mathbf{x}_0(0) = \mathbf{x}_{\text{init}} \quad (11h)$$

$$\mathbf{x}_{N-1}(dt) = \mathbf{x}_{\text{final}} \quad (11i)$$

$$\mathbf{x}_{n+1}(0) = \mathbf{x}_n(dt) \quad n = 0 : N - 2 \quad (11j)$$

$$\|\mathbf{v}_n(0)\|_{\infty} \leq v_{\max} \quad \forall n \quad (11k)$$

$$\|\mathbf{a}_n(0)\|_{\infty} \leq a_{\max} \quad \forall n \quad (11l)$$

$$\|\mathbf{j}_n\|_{\infty} \leq j_{\max} \quad \forall n. \quad (11m)$$

To solve this MIQP problem, Gurobi [93] can be applied due to its speed and efficiency. The value of dt can be obtained by applying the equation below:

$$dt = \max\{T_{v_x}, T_{v_y}, T_{v_z}, T_{a_x}, T_{a_y}, T_{a_z}, T_{j_x}, T_{j_y}, T_{j_z}\} / N. \quad (12)$$

Here, T_{v_i} , T_{a_i} , and T_{j_i} are obtained by applying $v_{\max}$, $a_{\max}$, and $j_{\max}$ on each axis

respectively as shown below:

$$\begin{aligned}
 T_{v_i} &= x_i/v_{max} \\
 T_{a_i} &= v_i/a_{max} \\
 T_{j_i} &= a_i/j_{max} \\
 i &= \{x, y, z\}.
 \end{aligned} \tag{13}$$

To summarize, the primary contribution of this section is not the invention of a new algorithm, but rather the rigorous application and adaptation of the JPS, convex decomposition, and MIQP framework to the specific problem of quadrotor landing on a UAVC in unknown environments, and its systematic evaluation. By integrating JPS for efficient global pathfinding (with pruning rules to reduce redundant node exploration), convex decomposition for constructing SFCs that ensure collision-free convex regions, and MIQP for generating dynamically feasible trajectories (strictly enforcing constraints on velocity, acceleration, jerk, and obstacle avoidance), this section establishes a robust hard-constrained optimization pipeline tailored to high-risk UAV landing scenarios. The systematic evaluation of this pipeline, including validating its strict compliance with dynamic and collision constraints through simulations, further clarifies its value in addressing the safety-critical needs of UAV autonomous landing on UAVC.

4 Soft-constrained Optimization via Kinodynamic Path Search and B-spline Refinement

This section presents a trajectory planning method that balances real-time performance and smoothness by allowing controlled constraint violations with penalty costs. It includes two main subsections. Sec. 4.1 presents a global planner combining A* with kinodynamic path search to consider motion dynamics from the outset. Then Sec. 4.2 proposes a local planner, which applies B-spline optimization with well-designed cost functions to refine trajectories while balancing smoothness, obstacle clearance, and feasibility.

4.1 Global Planner via Kinodynamic Path Search

In the global planner section, the A* algorithm is a primary method for finding the shortest path from the start to the goal. However, paths generated by traditional A* algorithms take the form of piecewise linear polylines because they do not account for the continuous dynamic processes involved in state transitions. To address this limitation, the planner here integrates kinodynamic path search into the A* framework, enabling it to more robustly handle the uncertainties of unknown environments. In unknown environments, obstacles may appear unexpectedly, and environmental information is often continuously updated through real-time perception. The kinodynamic path search directly incorporates the UAV's dynamic states including position,

velocity, acceleration, and jerk into the state space model. This allows the planner not only to identify a geometrically feasible path but also to ensure that the path remains dynamically executable even when new obstacles are detected.

The UAV's states, including position, velocity, acceleration, jerk, can be represented as $\mathbf{x}(t), \mathbf{v}(t), \mathbf{a}(t), \mathbf{j}(t)$. In the context of quadrotor dynamics, this formulation can be interpreted as a linear time-invariant (LTI) system. Thus, the state vector $\mathbf{x}(t) \in \mathbb{R}^9$ of the UAV can be represented as below:

$$\mathbf{x}(t)^T = [\mathbf{x}(t)^T \quad \mathbf{v}(t)^T \quad \mathbf{a}(t)^T]. \quad (14)$$

For the control input of the system, let $\mathbf{u}(t) = \mathbf{j}(t)$, where $\mathbf{j}(t) \in [-j_{max}, j_{max}]^3 \subset \mathbb{R}^3$. Hence, the state space model of the system can be derived as follows:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{S}\mathbf{x} + \mathbf{T}\mathbf{u} \\ \mathbf{S} &= \begin{pmatrix} \mathbf{0} & \mathbf{I}^3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}^3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \mathbf{T} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}^3 \end{pmatrix}. \end{aligned} \quad (15)$$

The general solution for the state space model is derived as follows:

$$\mathbf{x}(t) = e^{\mathbf{S}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{S}(t-\tau)}\mathbf{T}\mathbf{u}(\tau) d\tau, \quad (16)$$

where $\mathbf{x}(0)$ is the system's initial state. The resulting solution represents the UAV system's trajectory under the imposed control input $\mathbf{u}(t)$.

To generate a trajectory that optimizes control cost and time, the cost of the trajectory can be defined as follows:

$$J(T) = \int_0^T \|\mathbf{j}(t)\|^2 dt + wT, \quad (17)$$

where T denotes the trajectory's duration and w is the corresponding weighting parameter. Due to the definition of the A* algorithm, the total cost of the generated

path can be defined as follows:

$$g_c = f_c + f_h, \quad (18)$$

where f_c represents the cumulative cost incurred from the initial state to the current state, while f_h represents the estimated cost required to reach the target from the current state. Furthermore, the trajectory can be defined through a set of K control inputs spanning from the initial point to the target. Hence, according to Eq. (17), f_c can be defined as

$$f_c = \sum_{i=0}^K \|\mathbf{j}_i\|^2 \tau. \quad (19)$$

Pontryagin's minimum principle can be applied to compute f_h to minimize the cost function $J(T)$ evaluated from the current state to the goal state. According to Eq. (17), with w treated as a fixed constant, the cost to be minimized can be written as

$$f_h(t) = \int_0^T \|\mathbf{j}(t)\|^2 dt, \quad (20)$$

Hence, the Hamiltonian function will be defined as below:

$$\begin{aligned} H(\mathbf{x}_k, j_k, \boldsymbol{\lambda}_k) &= j_k^2 + \boldsymbol{\lambda}_k^T \dot{\mathbf{x}}_k, \\ &= j_k^2 + \lambda_1 v_k + \lambda_2 a_k + \lambda_3 j_k, \end{aligned} \quad (21)$$

$$\dot{\boldsymbol{\lambda}}_k = -\nabla_{\mathbf{x}_k} H(\mathbf{x}_k^*, j_k^*, \boldsymbol{\lambda}_k) = (0, -\lambda_1, -\lambda_2), \quad (22)$$

where j_k is the component of $\mathbf{j}(t)$ in x, y, z directions and $k \in \{x, y, z\}$, T represents the ending time, $\boldsymbol{\lambda}_k = (\lambda_1, \lambda_2, \lambda_3)$ is the costate of the Hamilton function, $\mathbf{x}_k^*$ and j_k^* represents the optimal state and input respectively. According to Eq. (22), the costate can be solved as:

$$\boldsymbol{\lambda}_k(t) = \frac{1}{T} \begin{bmatrix} -2\alpha_k \\ 2\alpha_k t + 2\beta_k \\ -\alpha_k t^2 - 2\beta_k t - 2\gamma_k \end{bmatrix}. \quad (23)$$

The optimal input can be solved as:

$$\begin{aligned} j_k^*(t) &= \arg \min_{j_k(t)} H(\mathbf{x}_k^*(t), j_k(t), \boldsymbol{\lambda}_k(t)) \\ &= \frac{1}{2}\alpha_k t^2 + \beta_k t + \gamma_k. \end{aligned} \quad (24)$$

Thus, by integrating Eq. (24), the optimal state trajectory can be solved as:

$$\mathbf{x}_k^*(t) = \begin{bmatrix} \frac{\alpha_k t^5}{120} + \frac{\beta_k t^4}{24} + \frac{\gamma_k t^3}{6} + \frac{a_{k0} t^2}{2} + v_{k0} t + p_{k0} \\ \frac{\alpha_k t^4}{24} + \frac{\beta_k t^3}{6} + \frac{\gamma_k t^2}{2} + a_{k0} t + v_{k0} \\ \frac{\alpha_k t^3}{6} + \frac{\beta_k t^2}{2} + \gamma_k t + a_{k0}, \end{bmatrix}, \quad (25)$$

with the initial state $\mathbf{x}_k(0) = (p_{k0}, v_{k0}, a_{k0})$. By substituting the result in Eq. (24) to Eq. (20), the cost of the optimal trajectory can be calculated as:

$$J_k^*(T) = \frac{\alpha_k^2 T^5}{20} + \frac{\alpha_k \beta_k T^4}{4} + \frac{\alpha_k \gamma_k T^3}{3} + \frac{\beta_k^2 T^3}{3} + \beta_k \gamma_k T^2 + \gamma_k^2 T. \quad (26)$$

Assuming the initial state is given and that the terminal state is defined by $\mathbf{x}_k(T) = (p_{kf}, v_{kf}, a_{kf})$, parameters $\{\alpha_k, \beta_k, \gamma_k\}$ can be solved as follows:

$$\begin{bmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{bmatrix} = \frac{1}{T^5} \begin{bmatrix} 720 & -360T & 60T^2 \\ -360T & 168T^2 & -24T^3 \\ 60T^2 & -24T^3 & 3T^4 \end{bmatrix} \begin{bmatrix} \Delta p_k \\ \Delta v_k \\ \Delta a_k \end{bmatrix}, \quad (27)$$

$$\begin{bmatrix} \Delta p_k \\ \Delta v_k \\ \Delta a_k \end{bmatrix} = \begin{bmatrix} p_{kf} - p_{k0} - v_{k0}T - \frac{1}{2}a_{k0}T^2 \\ v_{kf} - v_{k0} - a_{k0}T \\ a_{kf} - a_{k0} \end{bmatrix}, \quad (28)$$

4.2 Local Planner via B-Spline Refinement with Cost Functions

Although the A* algorithm combined with kinodynamic path search techniques can generate trajectories that satisfy both the kinematic and dynamic constraints of a system, these trajectories are often suboptimal and may lie perilously close to

obstacles [68]. To mitigate these issues, B-spline trajectory optimization is employed to improve both the clearance and the smoothness of the generated paths. A B-spline curve can be characterized by its control points $\{C_0, C_1, \dots, C_N\} \subset \mathbb{R}^3$, a knot vector $[t_0, t_1, \dots, t_M]$, and its degree P . The relationship among the total number of control points, the number of knots, and the degree is given by $M = N + P + 1$. In our application, the degree is fixed at $P = 4$. Furthermore, a uniform B-spline is utilized so that the difference between consecutive knots is constant, i.e., $t_{k+1} - t_k = \delta t$, $\forall k \in \{0, 1, \dots, M - P\}$, where δt represents the uniform time interval. To improve the stability of the numeric calculation, time t can be normalized as $\theta(t) = (t - t_k)/\delta t$, where $t \in [t_k, t_{k+1})$. Hence, a fourth-order B-spline curve can be derived as follows [94]:

$$\mathbf{P}(t) = \boldsymbol{\theta}(t)^T \mathbf{M} \begin{bmatrix} C_{k-4}^T \\ C_{k-3}^T \\ C_{k-2}^T \\ C_{k-1}^T \\ C_k^T \end{bmatrix}, \boldsymbol{\theta}(t) = \begin{bmatrix} 1 \\ \theta(t) \\ \theta^2(t) \\ \theta^3(t) \\ \theta^4(t) \end{bmatrix}, \quad (29)$$

$$\text{where } \mathbf{M} = \frac{1}{4!} \begin{bmatrix} 1 & 11 & 11 & 1 & 0 \\ -4 & -12 & 12 & 4 & 0 \\ 6 & -6 & -6 & 6 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}. \text{ The convex hull property of B-splines}$$

enhances safety in unknown environments. Since the entire trajectory lies within the convex hull of its control points, adjusting control points to avoid newly detected obstacles ensures the entire path remains collision-free. This reduces the risk of trajectory segments inadvertently entering hazardous regions, a common concern in unknown environments where obstacle distribution is not fully known, by confining

adjustments to the convex hull of updated control points. Furthermore, an important attribute of B-spline curves is that their k^{th} derivative also exhibits B-spline curves. As a consequence, the control points associated with the velocity $\mathbf{V}_{ni}$, acceleration $\mathbf{A}_{ni}$, and jerk $\mathbf{J}_{ni}$ can be computed as follows:

$$\begin{aligned}\mathbf{V}_{ni} &= \frac{\mathbf{C}_{n,i+1} - \mathbf{C}_{n,i}}{dt}, \mathbf{A}_{ni} = \frac{\mathbf{V}_{n,i+1} - \mathbf{V}_{n,i}}{dt}, \\ \mathbf{J}_{ni} &= \frac{\mathbf{A}_{n,i+1} - \mathbf{A}_{n,i}}{dt},\end{aligned}\quad (30)$$

where dt denotes the time interval between successive control points. Due to the inherent properties of B-spline curves, it is sufficient to optimize only the subset of $N + 1 - 2P$ control points $\{\mathbf{C}_P, \mathbf{C}_{P+1}, \dots, \mathbf{C}_{N-P}\}$, when designing an optimized B-spline trajectory. Consequently, the following optimization problem is formulated:

$$\min_{\mathbf{C}} J = \omega_1 J_1 + \omega_2 J_2 + \omega_3 J_3, \quad (31)$$

where ω_1 , ω_2 , and ω_3 represent weights of smoothness, collision, and feasibility cost, respectively. To keep the penalties on the same order of magnitude, ω_3 is set to be much smaller compared to ω_1 and ω_2 in practice.

To enforce dynamic smoothness along the trajectory, the cost term J_1 is introduced as a smoothness penalty. This penalty is designed to discourage abrupt changes in the trajectory by penalizing the squared Euclidean norms of its higher-order derivatives, which are the velocity, acceleration, and jerk. Specifically, J_1 can be represented as

$$J_1 = \sum_{i=0}^{N-1} \|\mathbf{V}_i\|_2^2 + \sum_{i=0}^{N-2} \|\mathbf{A}_i\|_2^2 + \sum_{i=0}^{N-3} \|\mathbf{J}_i\|_2^2. \quad (32)$$

For the collision penalty, the main function is to ensure there are no collisions between the UAV and the obstacles [68]. By the property of the Bezier curve, if the convex hull is collision-free with the obstacles, the whole curve is also collision-free.

As a result, the distance (s) between the points in the convex hull and the voxel in the obstacle should satisfy $s > 0$.

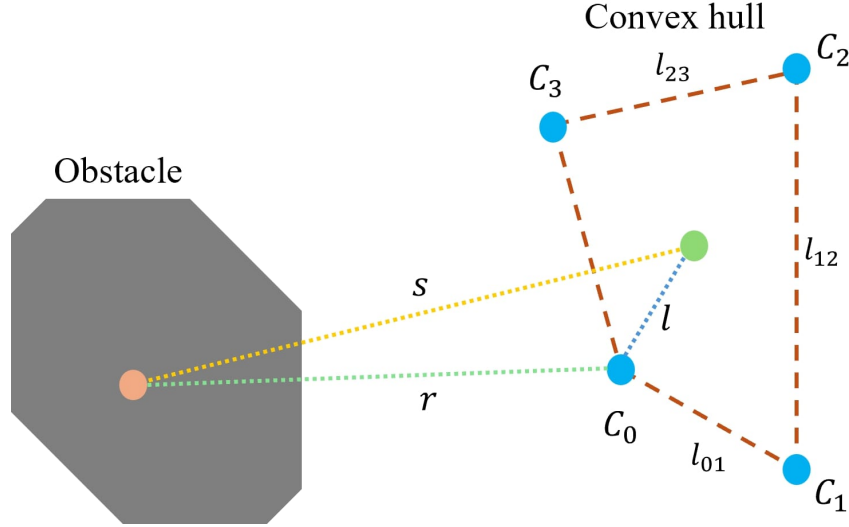


Figure 12: Collision-avoidance via the B-spline convex hull. The trajectory segment is confined within the convex hull of its four control points C_0 to C_3 . By guaranteeing each control point remains outside the obstacle's region and enforcing that the distance between consecutive control points does not exceed one third of the obstacle clearance radius r , the entire continuous curve is collision-free.

To ensure $s > 0$, Fig. 12 illustrates the principle of the collision avoidance. Due to the triangle inequality, $s > r - l$ can be derived, where r is the distance between the control points and the voxel in the obstacle, and l is the distance between any points in the convex hull and the control points. Similarly, $l \leq l_{01} + l_{12} + l_{23}$ can also be derived. By combining the previous inequalities, $s > r - l_{01} - l_{12} - l_{23}$ can be derived. Thus, if the following conditions can be satisfied

$$r > 0, \quad l_{j,j+1} < r/3 \quad (j \in \{0, 1, 2\}), \quad (33)$$

the whole generated curve will not collide with the obstacles. The advantage of this

procedure lies in its efficient and reliable collision avoidance mechanism tailored for B-spline trajectory optimization. By leveraging the convex hull property of B-spline curves (i.e., the entire curve lies within the convex hull of its control points), it shifts the collision check from verifying every point on the continuous curve to constraining the relationships between discrete control points and obstacles. This significantly reduces computational complexity, as checking distances between control points and obstacles, as well as between adjacent control points, is far more efficient than validating the infinite points on the curve. Moreover, the quantifiable conditions provide clear, implementable constraints for optimization algorithms. They enable systematic adjustment of control points during trajectory refinement, ensuring the curve inherently avoids obstacles without excessive conservative deviations. This balance between safety and efficiency makes the method particularly suitable for real-time trajectory planning in dynamic or unknown environments, where rapid computation and robust collision avoidance are critical.

Hence, the collision cost can be mathematically modeled as the cumulative repulsive forces exerted by obstacles on individual control points along the trajectory

$$J_2 = \sum_{i=P}^{N-P} j_c(r(\mathbf{C}_i)), \quad (34)$$

where j_c represents a differentiable cost function, with r_{thr} determining the minimum distance needed to clear obstacles. Thus, j_c can be represented as

$$j_c(r(\mathbf{C}_i)) = \begin{cases} 0 & \text{if } r(\mathbf{C}_i) > r_{thr} \\ (r(\mathbf{C}_i) - r_{thr})^2 & \text{if } r(\mathbf{C}_i) \leq r_{thr} \end{cases}. \quad (35)$$

The collision avoidance cost (J_2) plays a key role in handling the uncertainties of unknown environments. Since obstacles in unknown settings may be partially perceived or appear abruptly, the differentiable collision cost function (j_c) dynamically

penalizes control points that are too close to newly detected obstacles. By calculating the distance between control points and obstacle voxels, and enforcing a safety threshold, the planner generates repulsive forces that steer the B-spline trajectory away from newly identified hazards. This real-time penalty mechanism ensures that even if obstacles are not mapped in advance, the trajectory can be refined to maintain a safe clearance, addressing the unpredictability of unknown environments.

The dynamic feasibility cost function ensures that the generated trajectory adheres to the system's inherent kinematic and dynamic constraints. This function is formulated by imposing penalties on deviations of critical control variables, namely jerk, as well as its related derivatives, such as velocity and acceleration, from predetermined limits. Such a formulation guarantees that the trajectory not only conforms to the theoretical dynamic model but is also realizable within the physical bounds of the system. To employ a smooth, continuously differentiable penalty function, the basic penalty function for the velocity in a single dimension can be:

$$f_v(v_k) = \mathcal{A} \ln \left(1 + \exp \left[\mathcal{B} (v_k - v_{\max}) \right] \right), k \in \{x, y, z\}, \quad (36)$$

where $\mathcal{A}$ and $\mathcal{B}$ are adjustable parameters that, respectively, modulate the penalty's magnitude and the transition's gradient. A similar smooth formulation can be applied to the acceleration penalty $f_a(\cdot)$ and $f_j(\cdot)$. Then, leveraging the B-spline curves' convex hull property, the associated penalties are imposed on the control points as described below:

$$\begin{aligned}
F_v &= \sum_{k \in \{x,y,z\}} \sum_{i=P-1}^{N-P} f_v(V_{ik}), \\
F_a &= \sum_{k \in \{x,y,z\}} \sum_{i=P-2}^{N-P} f_a(A_{ik}), \\
F_j &= \sum_{k \in \{x,y,z\}} \sum_{i=P-3}^{N-P} f_j(J_{ik}).
\end{aligned} \tag{37}$$

As a result, the overall dynamic feasibility cost J_3 can then be assembled as

$$J_3 = \lambda_v F_v + \lambda_a F_a + \lambda_j F_j, \tag{38}$$

where λ_v , λ_a , and λ_j are weighting coefficients that balance the contributions of velocity, acceleration, and jerk penalties. To make the optimization decouple from sampling interval and balance the contributions of velocity, acceleration, and jerk penalties, λ_v , λ_a , and λ_j are set to $1/(dt)^2$, $1/(dt)$, and 1 respectively.

In summary, the primary contribution of this section is not the invention of a new algorithm, but rather the rigorous application and adaptation of the kinodynamic path search (integrated with A*) and B-spline refinement framework to the specific problem of quadrotor landing on a UAVC in unknown environments, and its systematic evaluation. By combining kinodynamic path search (which embeds the UAV's dynamic states, position, velocity, acceleration, and jerk, into the A* global planning process) with B-spline optimization (leveraging convex hull properties and a weighted cost function for smoothness, collision avoidance, and dynamic feasibility), this section develops a soft-constrained optimization pipeline that balances real-time performance and trajectory quality. The systematic evaluation of this pipeline, including comparing its speed, computation efficiency, and success rate against benchmark planners (e.g., Ego-Planner) in simulated environments and

applying it in real-world environments, highlights its adaptability to the real-world demands of UAV autonomous landing on UAVC.

5 Nonlinear Model Predictive Control with Downwash Wind Model

This section details the design of the controller that guides the quadrotor through precise approach paths and ensures stable touchdown on a UAVC under aerodynamic disturbances. Sec. 5.1 derives the complete six-degree-of-freedom rigid-body dynamics in both world and body frames, incorporates a first-order inner-loop attitude approximation, and introduces a downwash wind disturbance model. Then Sec. 5.2 formulates an NMPC scheme. The full dynamics including disturbances are discretized via a fourth-order Runge–Kutta integrator and control inputs are computed in real time through sequential quadratic programming to achieve high-accuracy trajectory tracking and robust disturbance rejection.

5.1 Dynamics Model of Quadrotors

In developing the quadrotor dynamics model, two essential coordinate systems are employed: the world (inertial) frame W and the body-fixed frame B , as depicted in Fig. 13. Within the body-fixed frame, the positive x_B -axis is designated as the UAV’s forward (nose) direction, while the y_B and z_B axes are defined in accordance with the right-hand rule.

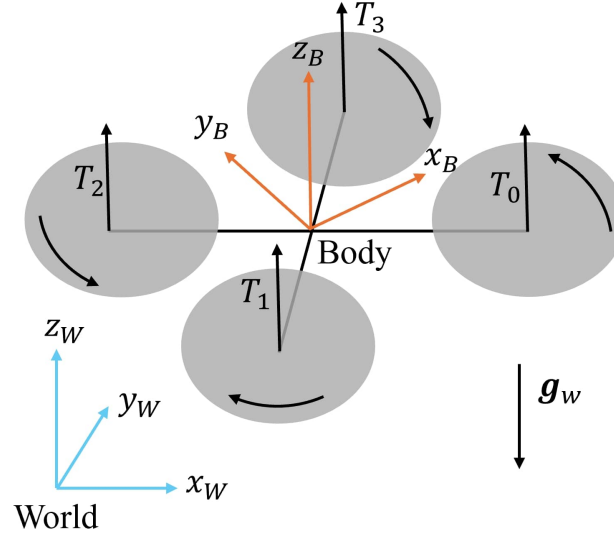


Figure 13: The UAV Model in the Body and World Frame

Assuming that the UAV has a 6-degree-of-freedom rigid body, the system model employed for the trajectory tracking of the UAV system can be presented in Eq. (39).

$$\begin{aligned}
 \dot{\mathcal{P}}(t) &= \mathcal{V}(t) \\
 \dot{\mathcal{V}}(t) &= \frac{1}{m}(\mathbf{R}_{WB}\mathbf{T} + \mathbf{F}_\sigma) - \mathbf{g}_W \\
 \dot{\mathbf{R}}_{WB} &= \mathbf{R}_{WB}[\boldsymbol{\omega}]_\times \\
 \dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{K} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix}),
 \end{aligned} \tag{39}$$

where $\mathcal{P} \in \mathbb{R}^3$ represents the position vector corresponding to the origin of the frame B as expressed in the frame W , and $\mathcal{V} \in \mathbb{R}^3$ signifies the UAV's velocity. The rotation matrix $\mathbf{R}_{WB} \in \mathbb{R}^{3 \times 3}$ denotes frame B 's orientation relative to W . The

total thrust produced by the rotors is indicated by $\mathbf{T} \in \mathbb{R}^3$, while $\mathbf{F}_\sigma \in \mathbb{R}^3$ accounts for external disturbance force and $\mathbf{g}_W \in \mathbb{R}^3$ represents gravitational acceleration. Additionally, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity expressed in B and $[\boldsymbol{\omega}]_\times$ represents the the 3×3 skew-symmetric matrix of $\boldsymbol{\omega}$, $\mathbf{J} = \text{diag}(J_x, J_y, J_z) \in \mathbb{R}^{3 \times 3}$ corresponds to the inertia matrix, and $[T_0, T_1, T_2, T_3]^T$ denotes the individual thrust outputs of the four rotors. The control allocation matrix $\mathbf{K}$ is

$$\mathbf{K} = \begin{bmatrix} -l_y & -l_y & l_y & l_y \\ -l_x & l_x & l_x & -l_x \\ -d & d & -d & d \end{bmatrix}, \quad (40)$$

where l_x, l_y are the displacement of the rotor, and d represents the rotor drag torque constant.

As for the attitude model, a low-level attitude controller can be assumed that functions as a first-order process, enabling precise tracking of the prescribed roll and pitch angles, ϕ_d and θ_d , respectively. The first-order approximation of the inner-loop dynamics encapsulates the essential low-level control behavior required by the NMPC. Accordingly, the inner-loop attitude model can be presented in Eq. (41).

$$\begin{aligned} \dot{\phi}(t) &= \lambda_\phi (A_\phi \phi_d(t) - \phi(t)) \\ \dot{\theta}(t) &= \lambda_\theta (A_\theta \theta_d(t) - \theta(t)) , \\ \dot{\psi}(t) &= \dot{\psi}_{\text{cmd}} \end{aligned} \quad (41)$$

where the angles ψ , ϕ , and θ represent the yaw, roll, and pitch of the UAV, respectively, while λ_θ and λ_ϕ specify the time constants for the inner-loop pitch and roll control systems, with A_θ and A_ϕ representing the associated control gains.

During the landing process, the dynamic model of the UAVC will be changed due to the downwash wind generated by the UAV. The disturbance can be modeled to maintain UAVC stability during landing. Assuming that the downwash wind will

only be generated under the region of the UAV, the thrust F_σ produced by the wind disturbance can be modeled as:

$$F_\sigma = \frac{1}{2} \rho C_d A_{eff} V_d^2, \quad (42)$$

where ρ represents the density of the air, m denotes the mass of the UAVC, C_d represents the thrust coefficient, A_{eff} is the effective area of the wind disturbance, and V_d is the air speed acting on the platform of the UAVC. Since V_d is related to the distance between the UAV and UAVC, V_d can be modeled as below:

$$V_d = Q e^{Rd}, \quad (43)$$

where Q and R are the coefficients gained from the real-world experiments, and d is the vertical distance between the UAV and UAVC. This exponential decay function was selected as it provides a simple yet physically plausible representation of fluid dynamic energy dissipation over distance, while remaining computationally lightweight for real-time evaluation within the NMPC horizon. Notably, because wind velocity produced by the rotor influences the downwash profile, the fitted parameters (Q , R) are unique to the UAV design. Any change in the design of the UAV, including the mass and rotor, requires repeating this calibration process to re-identify.

5.2 Generation of NMPC

The dynamic behavior of aerial vehicles is modeled by a system of nonlinear differential equations that comprehensively represent the complex aerodynamic interactions and coupling effects inherent in such systems. In this context, a theoretical framework based on NMPC is presented, which leverages the complete system dynamics

to optimize performance, particularly under conditions that demand aggressive trajectory tracking. The following vector characterizes the system's state $\mathbf{x} \in \mathbb{R}^9$

$$\mathbf{x} = \begin{bmatrix} \mathcal{P}^T & \mathcal{V}^T & \theta & \phi & \psi \end{bmatrix}^T, \quad (44)$$

where $\mathcal{P}$ and $\mathcal{V}$ denote the position and velocity vectors, respectively, and ψ , θ , and ϕ are the yaw, pitch, and roll angles. Accordingly, the UAV's control input can be expressed as

$$\mathbf{u} = \begin{bmatrix} \theta_{\text{cmd}} \\ \phi_{\text{cmd}} \\ T_{\text{cmd}} \end{bmatrix}, \quad (45)$$

with θ_{cmd} , ϕ_{cmd} , and T_{cmd} corresponding to the commanded pitch, roll, and thrust inputs, respectively.

The subsequent formulation defines the NMPC optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \int_0^T \left[(\mathbf{x}(t) - \mathbf{x}^{\text{ref}}(t))^T \mathbf{Q} (\mathbf{x}(t) - \mathbf{x}^{\text{ref}}(t)) \right. \\ & \quad \left. + (\mathbf{u}(t) - \mathbf{u}^{\text{ref}}(t))^T \mathbf{R} (\mathbf{u}(t) - \mathbf{u}^{\text{ref}}(t)) \right] dt \\ & \quad + (\mathbf{x}(T) - \mathbf{x}^{\text{ref}}(T))^T \mathbf{W} (\mathbf{x}(T) - \mathbf{x}^{\text{ref}}(T)), \quad (46) \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad t \in [0, T], \\ & \mathbf{u}(t) \in \mathcal{U}_C, \quad t \in [0, T], \\ & \mathbf{x}(0) = \mathbf{x}(t_0). \end{aligned}$$

In this formulation, the function $f(\cdot)$ denotes the nonlinear dynamic system as described in Eq. (39), $\mathbf{Q}$ is the state error weighting matrix, $\mathbf{R}$ is the control effort weighting matrix, and $\mathbf{W}$ is the terminal state weighting matrix. To convert the continuous dynamic system into a discrete formulation, an explicit fourth-order Runge-Kutta method is employed, utilizing the current state $\mathbf{x}_k$, the control input $\mathbf{u}_k$, and the time step Δt . Consequently, the dynamic system presented in Eq. (39)

can be discretized as

$$\mathbf{x}_{k+1} = f_{RK4}(\mathbf{x}_k, \mathbf{u}_k, \Delta t). \quad (47)$$

Accordingly, the NMPC problem can be reformulated in discrete form as

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{k=0}^{N-1} \left((\mathbf{x}_k - \mathbf{x}_k^{\text{ref}})^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}) \right. \\ & + (\mathbf{u}_k - \mathbf{u}_k^{\text{ref}})^T \mathbf{R} (\mathbf{u}_k - \mathbf{u}_k^{\text{ref}}) \\ & \left. + (\mathbf{x}_N - \mathbf{x}_N^{\text{ref}})^T \mathbf{W} (\mathbf{x}_N - \mathbf{x}_N^{\text{ref}}) \right) \\ \text{subject to} \quad & \end{aligned} \quad (48)$$

$$\mathbf{x}_{k+1} = f_{RK4}(\mathbf{x}_k, \mathbf{u}_k, \Delta t),$$

$$\mathbf{u}_k \in \mathcal{U}_C,$$

$$\mathbf{x}_0 = \mathbf{x}(t_0).$$

The objective function of Eq. (48) aims to minimize two key components over a finite prediction horizon, including the tracking error between the UAV's actual state and the reference trajectory, and the deviation of control inputs from their reference values, with an additional terminal cost to enhance stability. This balance ensures both accurate path following and efficient use of control efforts, critical for mitigating downwash-induced disturbances and maintaining landing stability. The constraints in Eq. (48) enforce physical feasibility and adherence to system dynamics. The discrete dynamics constraint, $\mathbf{x}_{k+1} = f_{RK4}(\mathbf{x}_k, \mathbf{u}_k, \Delta t)$, ensures the UAV's state evolution follows its nonlinear dynamics, discretized using a 4th-order Runge-Kutta method to accurately capture downwash wind disturbance. Control inputs are bounded within feasible ranges ($\mathbf{u}_k \in \mathcal{U}_C$) to prevent motor saturation or unstable maneuvers, while the initial condition constraint anchors the optimization to the UAV's real-time state, ensuring predictions align with actual flight conditions.

In summary, this section details the design of an NMPC framework integrated with a dynamic downwash wind model to enhance the stability of UAV landing on a

UAVC under aerodynamic disturbances. First, the six-degree-of-freedom rigid-body dynamics of the quadrotor are derived in both world and body frames, incorporating a first-order inner-loop attitude approximation. A critical addition is the downwash wind disturbance model, which quantifies the thrust generated by the UAV's rotor downwash on the UAVC. This model relates the wind speed to the distance between the UAV and UAVC via an exponential decay function, capturing the aerodynamic interaction during landing. Second, the NMPC scheme is formulated by defining the system state (position, velocity, attitude angles) and control inputs (commanded pitch, roll, thrust). The optimization objective minimizes the tracking error between the UAV's actual state and the reference trajectory, as well as the deviation of control inputs from their reference values, with a terminal cost to enhance stability. The continuous nonlinear dynamics are discretized using a fourth-order Runge–Kutta integrator. This framework ensures high-accuracy trajectory tracking and robust rejection of downwash-induced disturbances, ultimately stabilizing the UAV during touchdown on the UAVC.

6 Experimental Verification and Result Analysis

This section focuses on validating the effectiveness and robustness of the proposed trajectory planning methods and control framework through comprehensive experiments. Sec. 6.1 introduces the overall system framework, including the communication architecture based on the Robot Operating System (ROS), the software environment, and the interaction mechanism between key modules (e.g., perception, trajectory planning, and control), which enables the seamless operation of perception, planning, and control components. Subsequently, Sec. 6.2 describes the design of simulated experiments, covering the selection of the Gazebo simulator, the configuration of virtual sensors and UAV/UAVC models, scenario settings with unknown obstacles, and the dynamic tuning of controller parameters to ensure consistency with real-world conditions. Building on this simulated experiment design, Sec. 6.3 presents experiments conducted in a simulated environment to evaluate the performance of both hard-constrained and soft-constrained planners, as well as the stability of the proposed controller. To verify the practical viability of the methods, Sec. 6.4 presents the setup of real-world experiments, including the hardware specifications of the UAV and UAVC (e.g., onboard computing units, perception sensors, and flight controllers), the experimental environment layout, and parameter configurations for planners and controllers, providing a detailed account of the physical implementation to validate the practical feasibility of the proposed methods. Finally, Sec. 6.5 details real-world experiments, which further verify the practical applicability of

the soft-constrained planner and the NMPC framework integrated with a dynamic downwash wind model. This includes determining the downwash wind model, and analyzing trajectory tracking performance and UAVC stability during landing.

6.1 System Architecture and Implementation Environment

This subsection details the practical deployment of the autonomous landing strategy. As illustrated in Fig. 14, state information is relayed from the UAVC to the UAV through ROS's multi-node communication framework. Its peer-to-peer publish-subscribe architecture allows camera drivers, IMU filters, state estimation algorithms, trajectory planning modules and motor controllers to operate as independent nodes that communicate via standardized message topics. Configuration and orchestration of these modules are simplified through ROS launch files and the parameter server. The ROS control framework ensures real-time hardware abstraction and high-rate control loop execution. All software executes on Ubuntu 20.04 with the ROS Noetic distribution, providing the reliability and modularity necessary for iterative UAV control experimentation [95]. In practice, a real-time kinematic (RTK) positioning system can be used as the primary means of measuring the UAVC's position. Upon receiving the UAVC state information, the UAV autonomously computes an optimal real-time landing trajectory onto the UAVC platform.

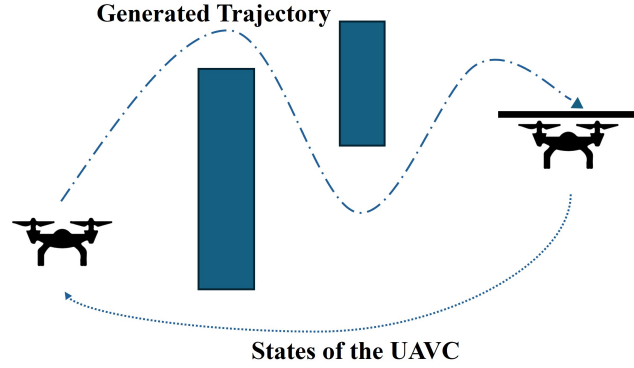


Figure 14: Overview of the system

Fig. 15 presents the system architecture of the UAV and the UAVC. The process commences with the transfer of the UAVC's state to the global planner. Then the obstacle detection in the unknown environment is implemented by capturing depth images, converting point clouds to the UAV's body frame using calibrated intrinsics and extrinsics, and projecting them into the global frame in the mapping module. The system maintains a voxel-based occupancy grid, inflates occupied cells to enforce safety margins, identifies connected clusters of occupied voxels, and estimates obstacle positions and dimensions using voxel centroid and span calculations. The mapping method used is based on MIT ACL Laboratory's *acl-mapping* project¹. It focuses on environmental mapping for autonomous systems in unknown environments. A key feature is its selective publication of only ground-above points (filtering irrelevant ground data) to prioritize obstacle spatial information, critical for real-time collision avoidance and trajectory planning. Its mapping results use blue for unknown space and orange for occupied-known space, with a ROS launch

¹The *acl-mapping* project is hosted on GitLab: <https://gitlab.com/mit-acl/lab/acl-mapping>

command (`roslaunch global_mapper_ros global_mapper_node.launch`) enabling seamless integration with our research's ROS-based system. Next, the global planner will generate a global path starting from the initial point to the goal point. This preliminary path is refined by a local planner that utilizes the B-spline curves' convex hull property to achieve a smooth trajectory. The refined trajectory, including the position vector $\mathbf{P}$, velocity vector $\mathbf{V}$, and acceleration vector $\mathbf{A}$, is subsequently transmitted to the NMPC module. By integrating the optimized trajectory with real-time state feedback from the UAV, the NMPC module addresses the control challenge using a fourth-order Runge-Kutta integration method paired with a multiple shooting technique. In particular, a boundary value problem (BVP) can be solved on each interval with continuity constraints enforced between successive segments [96]. Given the imposed constraints and the system dynamics' intrinsic nonlinearity, the optimization problem is solved as a nonlinear program (NLP). The NLP can be tackled via a Sequential Quadratic Programming (SQP) strategy, where the quadratic subproblems are resolved using an active-set method implemented in the qpOASES solver [97]. The computed control commands are then relayed to the UAV's low-level PID controller provided by PX4 to ensure precise maneuvering. Additionally, the NMPC is integrated within the UAVC's control module, and through incorporating a dynamic downwash wind model into the overall dynamic representation of the UAVC, it effectively stabilizes the UAV during the landing phase.

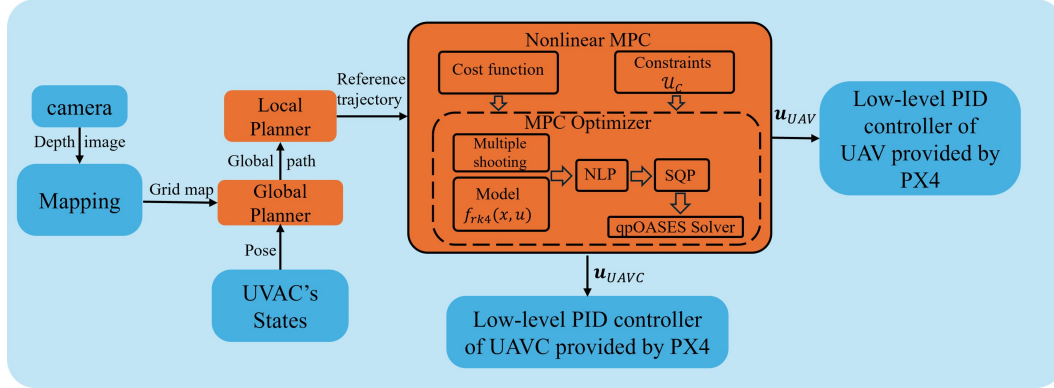


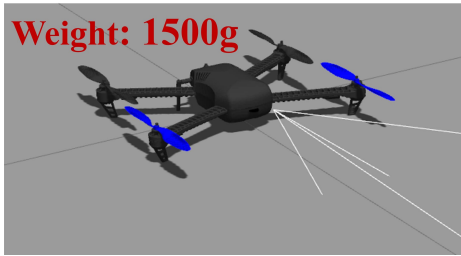
Figure 15: System Structure for the UAV and the UAVC, with the modules highlighted in orange marking the primary focus of our work.

6.2 Simulated Experiment Setup

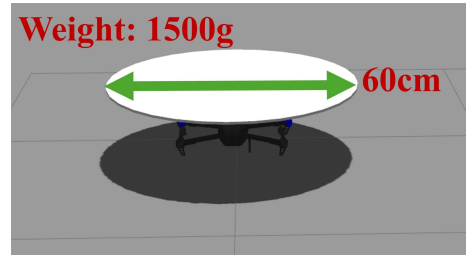
The simulated experiments were conducted using Gazebo, an open-source 3D robotics simulator maintained by Open Robotics, selected for its capability to replicate realistic and complex flight environments. Gazebo features a high-fidelity physics engine that accurately models key dynamics including gravity, inertia, and contact forces, which are essential for simulating quadrotor flight and UAV–UAVC interactions. Its advanced sensor models such as the IMU and depth camera, provide realistic perception and state estimation, closely mirroring real-world conditions. Gazebo’s modular architecture, plugin-based framework, and extensive library of prebuilt assets enabled efficient scenario construction and customization. Additionally, support for dynamic asset loading and distributed multi-server simulation facilitated the execution of large-scale, high-performance experiments tailored to the UAV landing task [98].

To maintain consistency with real-world experimental configurations, an Intel

RealSense D435 depth camera model was integrated into the simulation as the UAV's perception system, and PX4 was adopted as the flight control software to better replicate real-world operating conditions. The simulated UAV and UAVC are illustrated in Fig. 16, while the experimental environment is depicted in Fig. 17, where green cylinders represent obstacles in the unknown environment. As a result, there is no prior information about the obstacles for the UAV. In this setup, the UAV is programmed to compute an optimal trajectory that systematically avoids collisions with these obstacles, ensuring a successful landing on the UAVC platform.



(a) UAV used in the simulation



(b) UAVC used in the simulation

Figure 16: Simulated UAV and UAVC with basic parameters.

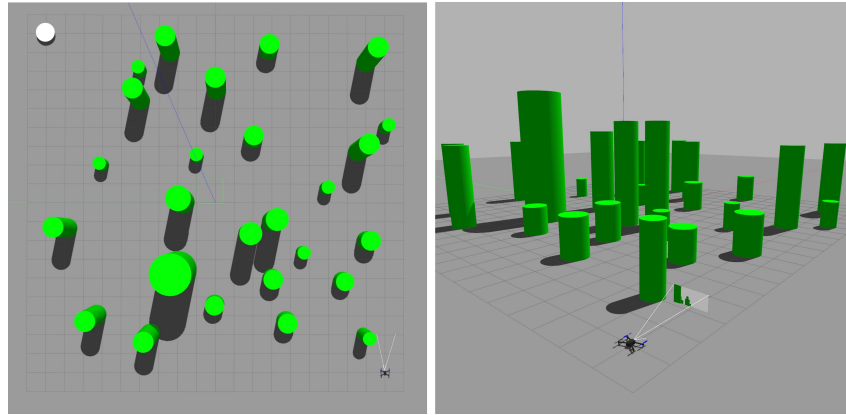


Figure 17: Scenario setting in the simulation experiment. Green cylinders represent obstacles.

A critical aspect of preparing the simulation was the calibration of the NMPC controller's weighting matrices, which directly influence the balance between trajectory tracking accuracy, control effort, and system stability. Given the sensitivity of these matrices to environmental conditions and mission requirements, a dynamic reconfiguration (as shown in Fig. 18) was employed to fine-tune their values in real time. This approach allowed for iterative adjustments to the state error weighting matrix ($\mathbf{Q}$) and control effort weighting matrix ($\mathbf{R}$), ensuring that the controller could adapt to the simulated dynamics of the UAV and UAVC. By visualizing the impact of parameter changes on key performance metrics including tracking error and thrust variation. This dynamic tuning process enabled the identification of optimal matrix values, laying the groundwork for robust controller performance in both simulated and subsequent real-world experiments. The parameters employed in the simulation experiment are detailed in Table 2.

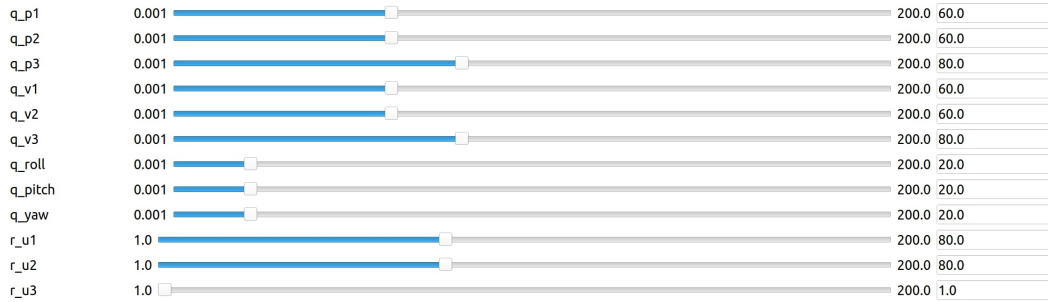


Figure 18: Dynamic reconfiguration

Table 2: Parameter settings in the simulation experiment

Parameters	Values	Parameters	Values
ω_1	2	$\mathbf{Q}_{\mathcal{P}}$	[60 60 80]
ω_2	2	$\mathbf{Q}_{\mathcal{V}}$	[60 60 80]
ω_3	0.1	$\mathbf{Q}_{\theta}$	20
$\mathcal{A}$	1	$\mathbf{Q}_{\phi}$	20
$\mathcal{B}$	0.5	$\mathbf{Q}_{\psi}$	20
A_{ϕ}	5	$\mathbf{R}$	[80 80 1]
A_{θ}	5	$\lambda_{\phi,\theta}$	0.9

6.3 Experiment Result in Simulated Environment

In the simulated experiment, the overall operational workflow is illustrated in Fig. 19, including a series of stages to ensure the UAV achieves a stable and precise landing on the UAVC. The process commences with the UAV's takeoff. Following takeoff, the UAV enters the trajectory generation and obstacle avoidance phase. Leveraging either the hard-constrained or soft-constrained planning algorithm, it dynamically computes optimal feasible paths to circumvent the green cylindrical obstacles distributed in the environment. This ensures collision-free navigation by strictly adhering to dynamic constraints and environmental restrictions. As the UAV advances along the planned trajectory and approaches the UAVC, the system switches to the landing phase to prioritize stability. At this stage, the proposed NMPC controller can keep the stability of the UAVC during landing. This controller adjusts control inputs in real time based on the relative positioning between the UAV and UAVC, mitigating disturbances induced by the UAV and ensuring the landing process proceeds smoothly and successfully.

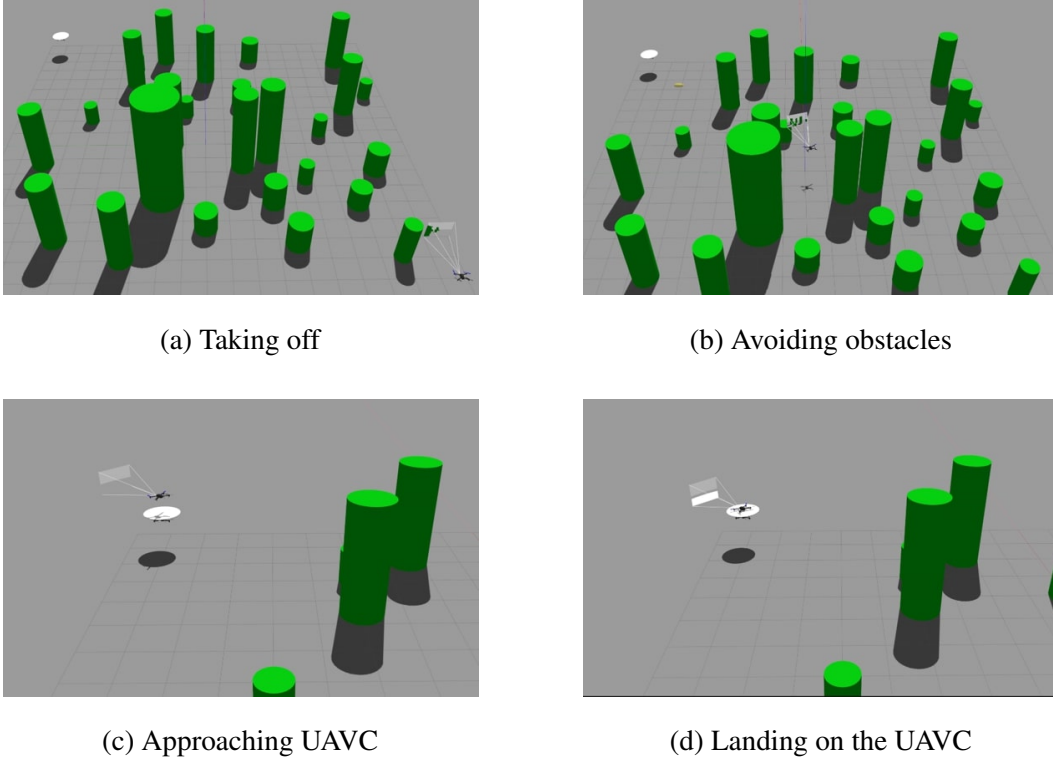


Figure 19: The overall working process in simulation

6.3.1 The Result of Simulated Experiment about Hard-constrained Planner

In the simulated environment, the proposed hard-constrained planning method has been verified. The hard-constrained planning framework, as detailed in Section 4, employs JPS as the global planner to generate an optimal initial path, followed by convex decomposition to construct an SFC consisting of connected convex polyhedrons. The local planner then formulates an MIQP to generate smooth, dynamically feasible trajectories within these corridors, strictly enforcing constraints such as maximum velocity, acceleration, as well as collision avoidance. The UAV was required to move from its original position to the UAVC's position. In Fig. 36, it represents an example of applying our proposed hard-constrained planner. Fig. 21

illustrates the trajectory generated by the hard-constrained planner and the UAV's tracking performance throughout the flight. The UAV achieved a maximum speed of 3.4 m/s and an average speed of 1.62 m/s, demonstrating its capability to navigate efficiently while respecting dynamic limits. To quantitatively assess trajectory tracking accuracy, the root mean squared error (RMSE) between the planned trajectory and the UAV's actual path was computed. The overall RMSE trend across the flight is visualized in Fig. 22, and detailed RMSE values for each spatial direction (X, Y, and Z axes) are summarized in Table 3.

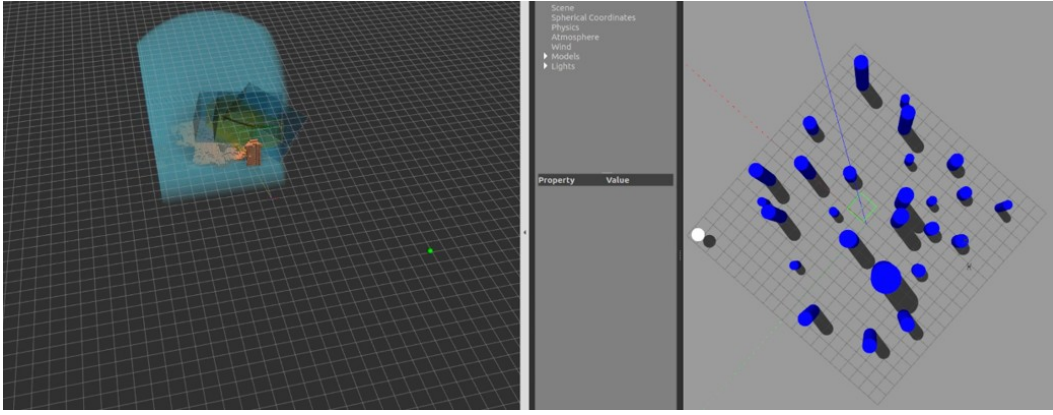


Figure 20: The blue point clouds represent the unknown space around the UAV. The orange grids represent the obstacles detected by the depth camera. The polyhedrons are the SFC built by the planner to constrain the UAV to fly in the safety areas.

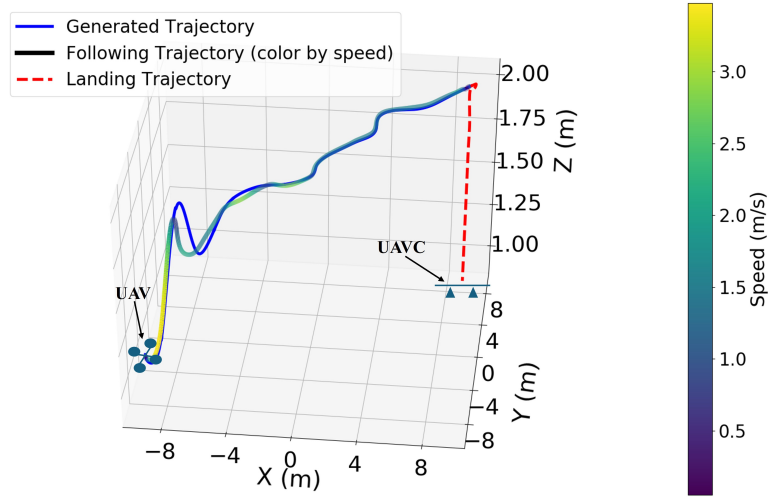


Figure 21: Generated trajectory and the following result by applying the proposed hard-constrained planner

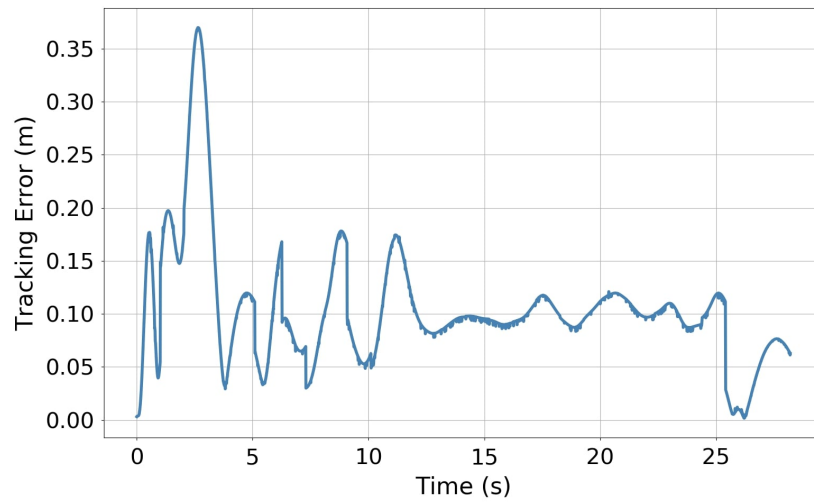


Figure 22: Tracking error during flight.

Table 3: Tracking error for the simulation experiment by applying hard-constrained planning

RMSE (X direction) [m]	0.08
RMSE (Y direction) [m]	0.08
RMSE (Z direction) [m]	0.04
RMSE (whole) [m]	0.12

6.3.2 The Result of Simulated Experiment about Soft-constrained Planner

To comprehensively evaluate the performance of the proposed soft-constrained planning method, complementary simulations were conducted in the same Gazebo environment, focusing on its ability to generate feasible trajectories and achieve precise tracking under dynamic conditions. Unlike hard-constrained approaches that enforce strict compliance with all constraints, the soft-constrained planner relaxes rigid feasibility requirements by allowing controlled constraint violations, which are penalized in the objective function to balance computational efficiency and trajectory quality. The soft-constrained planning framework employs a two-stage strategy: the global planner leverages the A* algorithm combined with kinodynamic path search techniques to generate an initial path that accounts for the UAV's dynamic states (position, velocity, acceleration, and jerk). The local planner then refines this path using B-spline trajectory optimization, which minimizes a composite cost function incorporating smoothness (penalizing abrupt changes in velocity, acceleration, and jerk), collision avoidance (via repulsive forces from obstacles), and dynamic feasibility (penalizing deviations from velocity, acceleration, and jerk limits). This design prioritizes real-time adaptability, making it suitable for dynamic environments where strict constraint enforcement may be computationally prohibitive.

Fig. 23 illustrates the trajectory generated by the soft-constrained planner and the UAV's tracking performance throughout the flight. The UAV achieved a maximum speed of 3.49 m/s and an average speed of 2.07 m/s, outperforming the hard-constrained method in average speed. This result is attributed to the soft-constrained approach's flexibility in balancing constraints and efficiency. To quantify tracking accuracy, RMSE between the planned trajectory and the UAV's actual path was computed. The temporal evolution of this tracking error is visualized in Fig. 24, while detailed RMSE values for the X, Y, and Z axes are summarized in Table 4. Notably, the RMSE in the Z-direction (0.04 m) was significantly lower than those in the X (0.13 m) and Y (0.11 m) directions, reflecting the planner's emphasis on precise altitude control, which is a critical requirement for subsequent landing phases. This performance confirms the effectiveness of the soft-constrained strategy. Despite allowing minor constraint violations, the penalty-based optimization ensures that the generated trajectories remain highly trackable, with the controller effectively mitigating deviations even in cluttered environments.

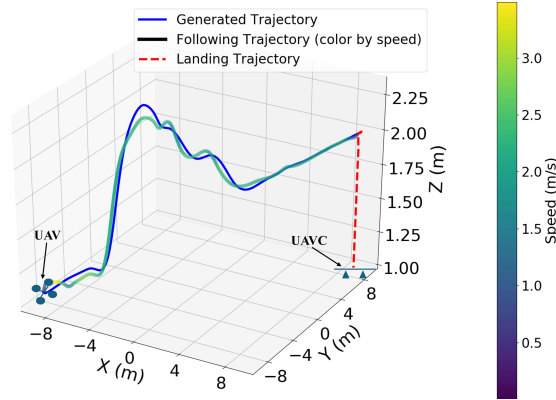


Figure 23: Generated trajectory and the following result by applying soft-constrained planner.

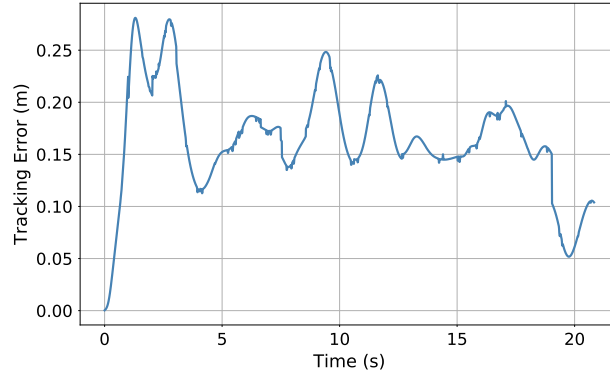


Figure 24: Tracking error during flight.

Table 4: Tracking error for the simulation experiment

RMSE (X direction) [m]	0.13
RMSE (Y direction) [m]	0.11
RMSE (Z direction) [m]	0.04
RMSE (whole) [m]	0.17

6.3.3 Comparison between Different Planners

To comprehensively assess the performance of the proposed hard-constrained planner and soft-constrained planner, a comparative study was conducted in a simulated environment, with Ego-Planner [71] selected as a benchmark. Ego-Planner is a representative gradient-based optimization method that does not rely on ESDF. Instead, it focuses on local trajectory refinement to enhance collision avoidance capabilities, making it a suitable candidate for evaluating the advantages of the proposed planners. All three planners were configured with identical maximum motion parameter limits, which were determined based on the physical performance of the simulated quadrotor and the requirement of simulating real-world flight safety.

Specifically, the maximum velocity (v_{max}) was set to 3.5 m/s , uniformly applied to the X, Y, and Z axes to prevent the UAV from exceeding the safe flight speed range; the maximum acceleration (a_{max}) was restricted to 2.0 m/s^2 , limiting the rate of speed change to avoid excessive inertial forces that could destabilize the UAV; and the maximum jerk (j_{max}) was set to 5.0 m/s^3 , constraining abrupt changes in acceleration to ensure trajectory smoothness and compatibility with the quadrotor's actuator response capabilities. These dynamic constraints were embedded into the core logic of each planner. For the hard-constrained planner, they were enforced as strict inequality constraints in the MIQP formulation, ensuring that the generated trajectory never violated these limits. For the soft-constrained planner, they were integrated into the dynamic feasibility cost function (J_3) through smooth penalty functions, where deviations from the limits were penalized to balance feasibility and efficiency. Ego-Planner, as the benchmark, adopted the same limits in its local gradient-based refinement process to eliminate differences caused by inconsistent dynamic constraints. The trajectories generated by these three planning approaches are visually depicted in Fig. 25, providing an intuitive illustration of their path-finding strategies.

Quantitative comparisons of key performance metrics, including maximum velocity, average velocity, success rate, and computation time, are summarized in Table 5. The success rate was determined from 20 tests carried out in various scenarios within the Gazebo simulator. A test was defined as successful only if the UAV could avoid all the obstacles and land on the UAVC successfully. The observation that Ego-Planner exhibits better real-time performance (with the shortest computation time of 6.45 ms) is consistent with its design as a lightweight planner that prioritizes efficiency by avoiding explicit ESDF construction. However, Ego-Planner demonstrates relatively low maximum velocity (2.63 m/s) and average velocity (1.58 m/s), coupled with a success rate of 80%. This performance profile

is likely due to its conservative local refinement strategy, which may compromise its reliability in unknown environments. The hard-constrained planner achieves a 100% success rate, ensuring strict compliance with dynamic constraints and obstacle avoidance requirements, but at the cost of a longer computation time (23.7 ms). In contrast, the soft-constrained planner demonstrates a higher maximum velocity (3.49 m/s) and average velocity (2.07 m/s) compared to the other two methods, with a high success rate of 95% and a moderate computation time (12.76 ms). This balance between efficiency, agility, and reliability highlights its potential for practical applications.

Given the above comparative analysis, the soft-constrained planner was chosen for the real-world experiment. The first reason is that the soft-constrained approach, with its lower computation time and flexible constraint handling, is more suitable for real-time trajectory replanning. Second, the soft-constrained planner's ability to balance trajectory smoothness, obstacle clearance, and dynamic feasibility aligns with the practical demands of autonomous landing on a moving UAV carrier, where both efficiency and robustness are critical. Thus, the soft-constrained planner is better suited to address the challenges of real-world operations.

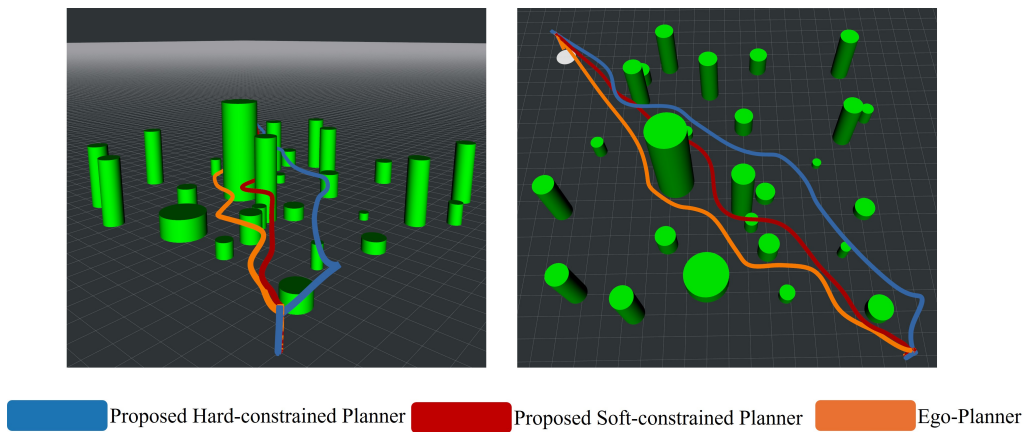


Figure 25: Trajectories generated by different planning methods

Table 5: Comparison of planning methods based on simulation results

Planning Method	Max. Vel. (m/s)	Avg. Vel. (m/s)	Success rate (%)	Computation Time (ms)
Ego-Planner	2.63	1.58	80	6.45
Hard-constrained method	3.40	1.62	100	23.7
Soft-constrained method	3.49	2.07	95	12.76

6.3.4 Stability Test of the Proposed Controller

To systematically evaluate the stability and disturbance rejection capability of the UAVC during the landing process, a simulation-based stability test was designed to analyze the dynamic responses of the UAVC under the proposed NMPC framework. The landing process was logically divided into three distinct phases for detailed assessment: pre-landing (the UAV is approaching but not yet in contact with the UAVC), landing (the UAV is in the process of touching down), and post-landing (the UAV has settled on the UAVC). It should be noted that due to the limitations of the Gazebo simulation environment, the aerodynamic downwash effect generated by the UAV's rotors could not be modeled. Thus, the analysis primarily focused on gravitational effects and excluded downwash-induced disturbances, with key observations concentrated on the pre-landing and post-landing phases. A critical indicator of UAVC stability is the variation in thrust required to maintain its position during the landing sequence. As illustrated in Fig. 26, the thrust percentage of the UAVC exhibits a clear increasing trend as the landing progresses. Specifically, the average thrust percentage during the pre-landing phase is approximately 0.40, which rises to around 0.60 in the post-landing phase. This increase is attributed to the additional load imposed by the UAV upon contact, requiring the UAVC to adjust

its thrust output to counteract the added weight and maintain stability. Notably, the actual relationship between thrust input and output is not strictly linear. This nonlinearity originates from two key factors inherent to the Gazebo simulation setup. First, the dynamic models of the UAVC's motors and electronic speed controllers (ESCs) in Gazebo are simplified compared to real-world hardware, meaning they do not fully replicate the ideal linear thrust-response characteristics. Second, the UAVC's maximum thrust is designed with redundancy to ensure operational safety and control flexibility, eliminating the need to reach the theoretically expected 80% thrust (a strict linear proportion to the doubled load) to maintain stable hovering.

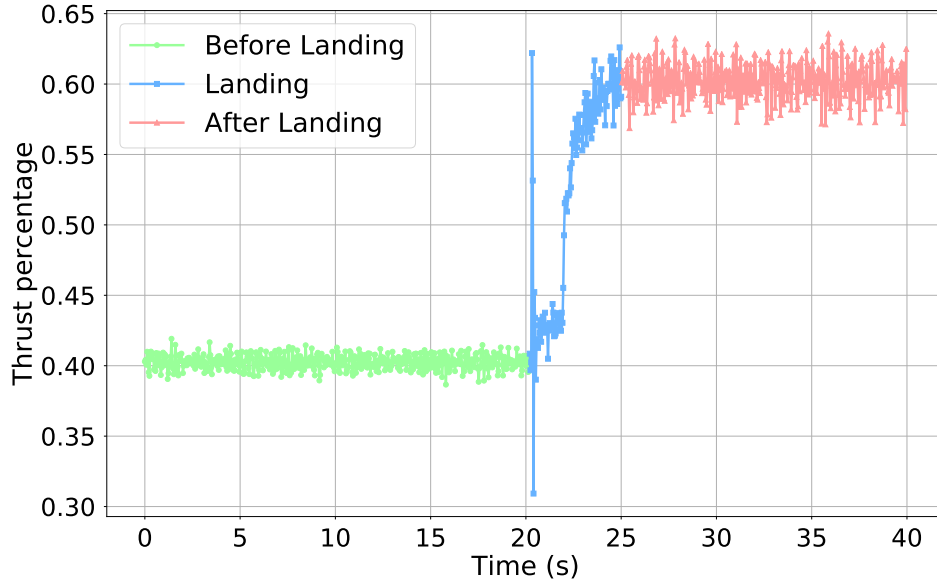
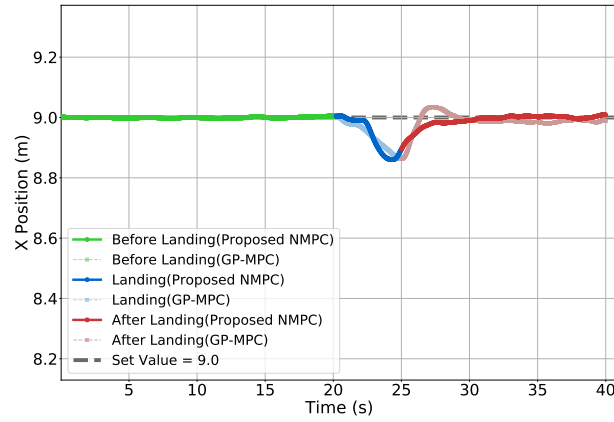


Figure 26: Thrust percentage variation during the landing process.

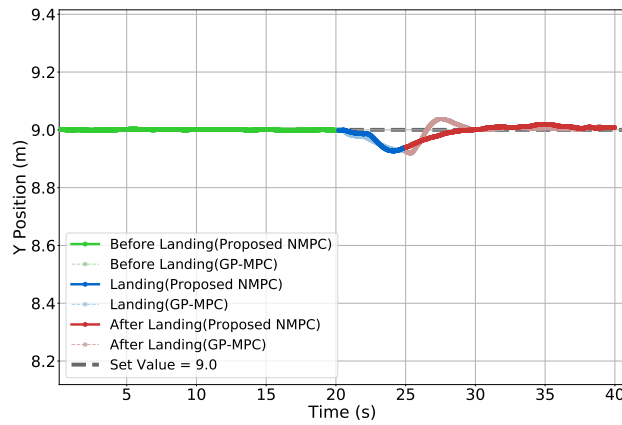
To further validate the superiority of the proposed NMPC in stabilizing the UAVC, a comparative experiment was conducted against the data-driven model predictive control (GP-MPC) method proposed in [90]. Both controllers were configured with identical key parameters: a sampling interval of 20 ms and a prediction horizon of 1 s, ensuring a fair comparison. The positional variations of the UAVC along the X,

Y, and Z axes throughout the landing process were monitored, and the results are presented in Fig. 27.

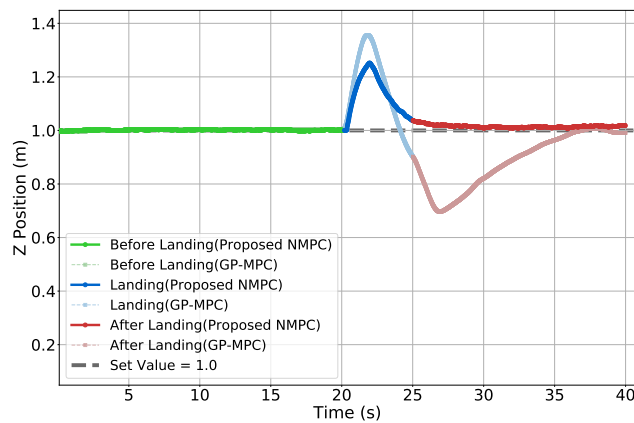
Quantitative analysis using root mean square error (RMSE) of the UAVC's position (summarized in Table 6) further confirms the performance advantage of the proposed NMPC. In the pre-landing phase, NMPC reduced the RMSE from (0.23, 0.15, 0.34) cm (with GP-MPC) to (0.19, 0.12, 0.21) cm, corresponding to improvements of 17%, 20%, and 38% along the X, Y, and Z axes, respectively. The most significant enhancement was observed in the post-landing phase: NMPC lowered the RMSE from (3.08, 2.25, 15.26) cm to (2.32, 1.86, 1.63) cm, achieving reductions of 25%, 17%, and 89% in each axis. These results highlight that the proposed NMPC framework exhibits stronger disturbance rejection capability, particularly in the vertical (Z) direction, where UAV-induced disturbances are most pronounced. As a result, our proposed NMPC framework ensures more stable and precise positioning of the UAVC during critical landing phases.



(a) X-axis position of UAVC



(b) Y-axis position of UAVC



(c) Z-axis position of UAVC

Figure 27: Variations in the UAVC's position along the X, Y, and Z axes during the landing process.

Table 6: RMSE of the UAVC's position during the landing process.

	X-axis [cm]	Y-axis [cm]	Z-axis [cm]
Pre-Landing RMSE (NMPC)	0.19	0.12	0.21
Pre-Landing RMSE (GP-MPC)	0.23	0.15	0.34
Post-Landing RMSE (NMPC)	2.32	1.86	1.63
Post-Landing RMSE (GP-MPC)	3.08	2.25	15.26

6.4 Real-World Experiment Setup

To validate the effectiveness of the proposed algorithms, real-world experiments were conducted using two unmanned aerial vehicles: a smaller quadrotor (UAV) and a UAVC serving as the landing platform. Both systems utilize PX4 firmware to ensure consistent low-level control and compatibility with the experimental framework. The smaller UAV is equipped with an onboard computer featuring a RockChip RK3588S system-on-chip (SoC), which integrates a quad-core Cortex-A76 and a quad-core Cortex-A55 (8 cores total). This configuration strikes a balance between high computational performance, which is critical for real-time trajectory generation, control command computation and power efficiency, making it suitable for resource-constrained UAV platforms.

For obstacle detection and environmental perception, the UAV employs an Intel RealSense D435 depth camera (Fig. 28), which provides a compact stereo depth sensor. This device delivers dense 3D point clouds at up to 30 frames per second

without requiring external synchronization hardware. Its global-shutter sensors minimize motion blur during high-speed maneuvers, while an 86-degree field of view ensures broad environmental coverage. Native support within the ROS ecosystem and cross-platform software development kits accelerate system integration, and its low power consumption and cost make it well-suited for small UAVs [99]. Notably, in the real-world experiments conducted for this work, no specific provisions have been made to address challenging environmental conditions such as low light, high glare, or dusty environments. Failure to account for these scenarios could degrade the depth camera's performance, potentially leading to sparse point clouds, increased noise, or inaccurate obstacle detection.



Figure 28: The Intel RealSense D435 used for obstacle detection

The UAV's flight control is managed by the NxtPX4 controller (Fig. 29), a PX4-compatible autopilot developed by the HKUST UAV Group and manufactured by MicoAir Tech [100]. It integrates a microcontroller, IMUs, and a barometer, with a compact form factor ($27 \times 32 \times 8 \text{ mm}$) and lightweight design (6.5 g) featuring a $20 \times 20 \text{ mm}$ mounting pattern. This makes it ideal for high-performance control experiments on compact multirotor platforms running PX4.

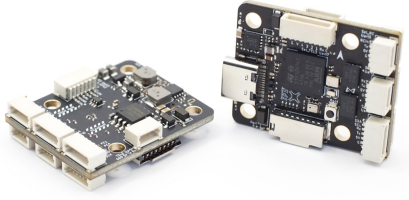


Figure 29: The flight controller used in the UAV

The UAVC is equipped with the Pixhawk 6C Mini controller (Fig. 30), distributed by Holybro [101]. This controller balances processing capability and compactness, supporting robust state estimation and control for larger UAV configurations like the UAVC.

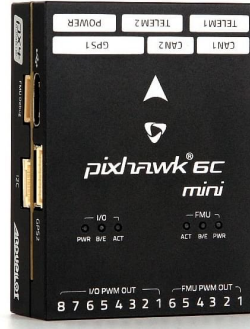
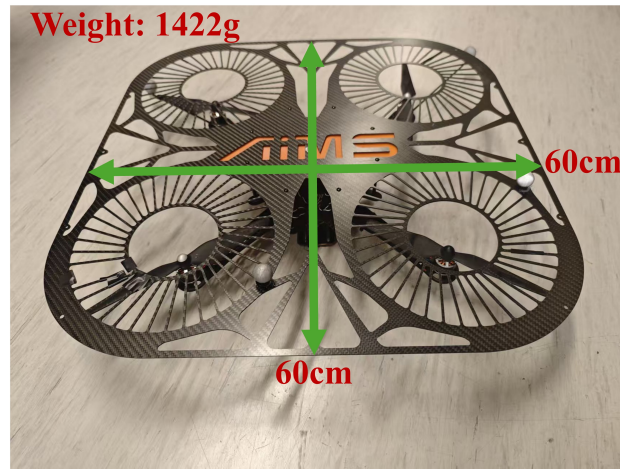


Figure 30: The flight controller used in the UAVC

Fig. 31 illustrates the UAV and UAVC employed in the real-world experiment, alongside their essential parameters, including the weights of both the UAV and UAVC and the dimensions of the platform. Besides, the parameter settings in the planner and controller can be found in Table 7.



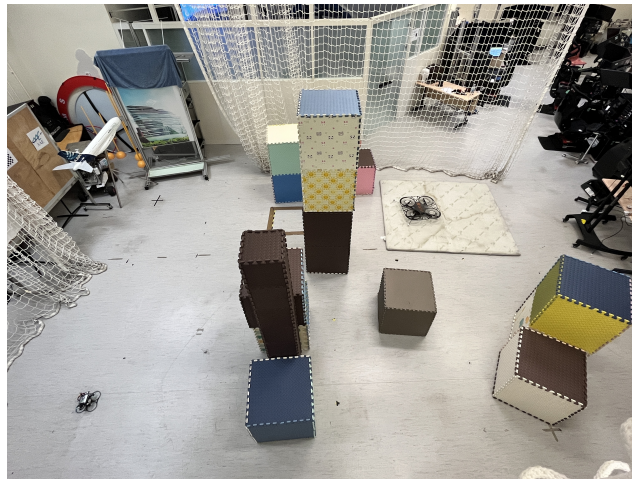
(a) UAV



(b) UAVC

Figure 31: The UAV and UAVC used in the experiment.

The configuration and setting of the experiment are illustrated in Fig. 32, which provides both a top-down and a side-view perspective of the environment. Additionally, the position of the obstacles is arranged arbitrarily so that there is no prior information about the obstacles for the UAV. The experiment area measures $4\text{ m} \times 6\text{ m}$, and the dimension of an obstacle block is $60\text{ cm} \times 60\text{ cm}$.



(a) Top view



(b) Side view

Figure 32: Environment in real-world experiment.

Table 7: Parameter settings in the real-world experiment

Parameters	Values	Parameters	Values
ω_1	2	$\mathbf{Q}_{\mathcal{P}}$	[80 80 100]
ω_2	2	$\mathbf{Q}_{\mathcal{V}}$	[60 60 80]
ω_3	0.1	$\mathbf{Q}_{\theta}$	20
$\mathcal{A}$	1	$\mathbf{Q}_{\phi}$	20
$\mathcal{B}$	0.5	$\mathbf{Q}_{\psi}$	20
$\mathbf{R}$	[80 80 1]	ρ	1.2 kg/m ³
C_d	1.1	A_{ϕ}	5
$\lambda_{\phi,\theta}$	0.9	A_{θ}	5

6.5 Experiment Result in Real-world Experiment

Based on the simulation results in Sec. 6.3.3, where the soft-constrained planner demonstrated a superior balance of computational speed and success rate, it was selected as the candidate for hardware implementation.

6.5.1 Determination of the Downwash Wind Model

To accurately model the dynamic behavior of the downwash wind generated by the UAV, a series of experiments was conducted to establish a quantitative relationship between the wind speed and the distance between the UAV and its supporting platform. The experimental setup employed an anemograph to continuously measure the wind speed induced by the UAV during flight. Measurements were taken at several predefined distances, ensuring comprehensive coverage of the flow field produced by the UAV's downwash. The experimental procedure was as follows. First, the UAV was flown under controlled conditions while the anemograph (as shown in

Fig. 33), positioned at distances varying along the longitudinal axis from the platform, recorded the wind speeds. This approach allowed for capturing the decaying trend of the downwash effect. The obtained data points were then processed using a non-linear least squares curve-fitting method to derive a mathematical representation of the association between the detected wind speed, denoted as V_d , and the distance d from UAVC's platform. The fitting procedure yielded a value $Q = 11.2$ and an exponent $R = -1.01$. Fig. 34 illustrates the experimental data (presented as discrete markers) alongside the fitted curve (the continuous line), demonstrating the effectiveness of the model in capturing the observed trends.



Figure 33: Anemograph used in the real-world experiment

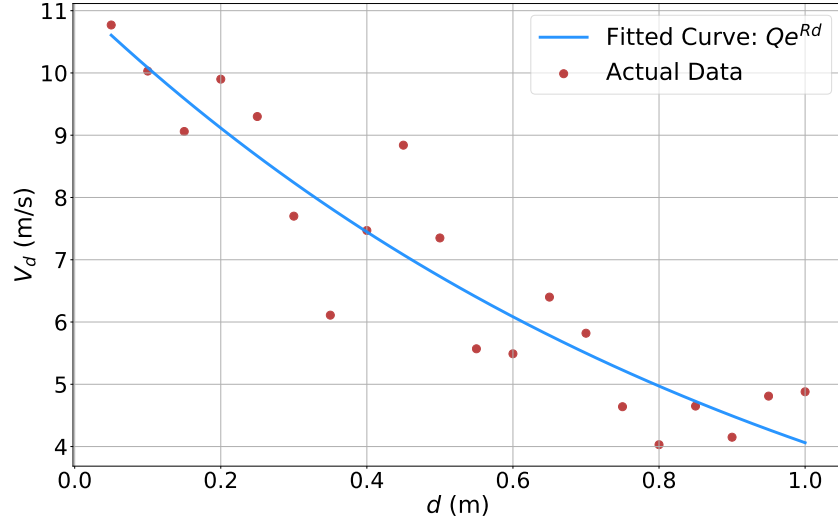


Figure 34: Experimental wind speed data (markers) and the fitted exponential decay curve (line)

The goodness of fit was quantified by calculating the RMSE and the coefficient of determination (R^2). The RMSE was determined to be 0.77, indicating that the average deviation between the observed data and the model is relatively low. Besides, an R^2 value of 0.87 suggests that approximately 87% of the variance in the measured wind speeds can be accounted for by the fitted model. Moreover, Spearman correlation coefficient (ρ_s) is applied to evaluate the monotonic relationship between observed data and predicted values. The value of ρ_s is around 0.914, which indicates that there is a strong positive correlation between the observed data and the prediction values. Such statistical indicators confirm the precision and robustness of the modeling approach.

6.5.2 Experiment Result

In the real-world experiment, the soft-constrained planner was adopted for trajectory planning, leveraging its advantages in real-time adaptability and computational

efficiency to avoid unknown environment obstacles. This section presents a detailed analysis of the experimental data, focusing on trajectory tracking performance, UAVC stability, and the impact of the dynamic downwash wind model. The soft-constrained planner generated smooth, collision-free trajectories by optimizing B-spline curves, which balance smoothness, obstacle avoidance, and dynamic feasibility through a weighted cost function. Throughout the experimental trials, the UAV was programmed to autonomously migrate from a predetermined initial position towards the UAVC, ultimately executing a controlled landing on its platform in the unknown environment. Fig. 35 presents an example of a real-world experiment. Fig. 36 illustrates the close alignment between the generated trajectory and the UAV's actual flight path, confirming the planner's ability to produce trackable paths in real-world conditions.



(a) Taking off



(b) Avoiding obstacles



(c) Approaching UAVC



(d) Landing on the UAVC

Figure 35: Snapshots of a real-world experiment.

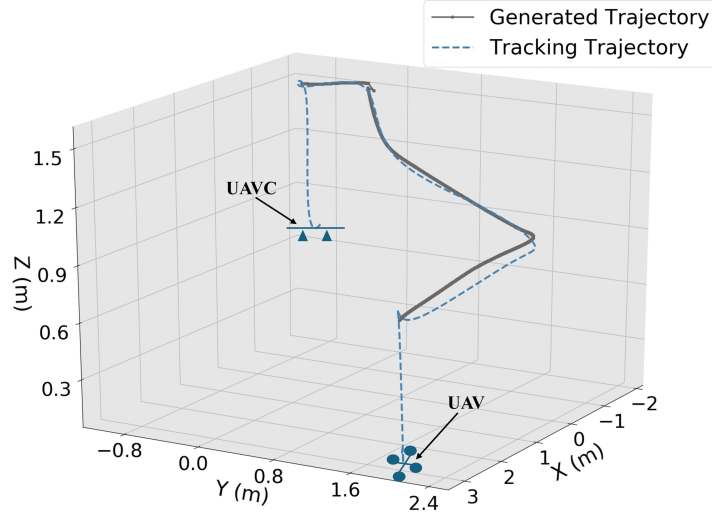


Figure 36: Generated trajectory and following result in real-world experiment

Quantitative analysis using RMSE further validates the tracking accuracy. As summarized in Table 8, the RMSE in the X-direction was 8.30 cm, 6.92 cm in the Y-direction, and 5.06 cm in the Z-direction, with an overall RMSE of 11.93 cm. The relatively smaller error in the Z-axis (5.06 cm) highlights the planner's effectiveness in maintaining precise altitude control, which is a critical requirement for safe landing, as vertical positioning directly affects the success of touchdown. The higher errors in the X and Y directions (8.30 cm and 6.92 cm) may be attributed to slight lateral disturbances from ambient wind, but they remain within acceptable bounds for landing accuracy, demonstrating the soft-constrained planner's robustness in balancing flexibility and precision. The temporal variation of tracking error (Fig. 37) shows that the error remains stable throughout the flight, with no abrupt spikes exceeding 0.25 m. This stability indicates that the NMPC controller, paired with the soft-constrained trajectory, effectively compensates for real-time disturbances, ensuring consistent tracking performance.

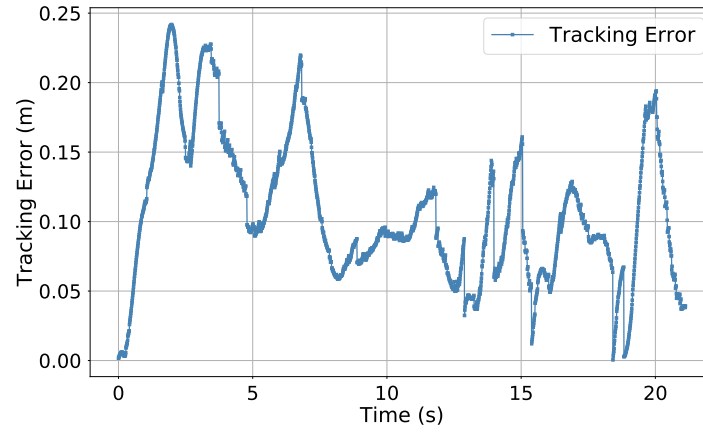


Figure 37: Tracking error of the UAV in the real-world experiment

Table 8: Tracking error in each direction

RMSE (X direction) [cm]	8.30
RMSE (Y direction) [cm]	6.92
RMSE (Z direction) [cm]	5.06
RMSE (whole) [cm]	11.93

After the UAV successfully acquired and tracked its real-time trajectory, it initiated the landing sequence onto the platform of the UAVC. During this maneuver, the downwash generated by the UAV exerts a considerable influence on the dynamic state of the UAVC. In the previously described NMPC framework, a dynamic model for the downwash flow was established, which captures how the thrust produced by the UAVC varies as a function of the relative positioning between the UAV and the UAVC. To facilitate analysis, similar to the simulation experiment, the entire landing process is segmented into three phases: pre-landing (UAV approaching but

not in contact), landing (UAV touching down), and post-landing (UAV settled on the UAVC). As illustrated in Fig. 38, the thrust percentage begins to increase during the landing phase and subsequently stabilizes in the after-landing phase. Specifically, the UAVC exhibits an initial average thrust percentage of approximately 0.41 prior to the landing process. As the UAV approaches and initiates its descent onto the UAVC, the thrust percentage increases to roughly 0.56. After the UAV has completed its landing, the thrust percentage stabilizes at approximately 0.57.

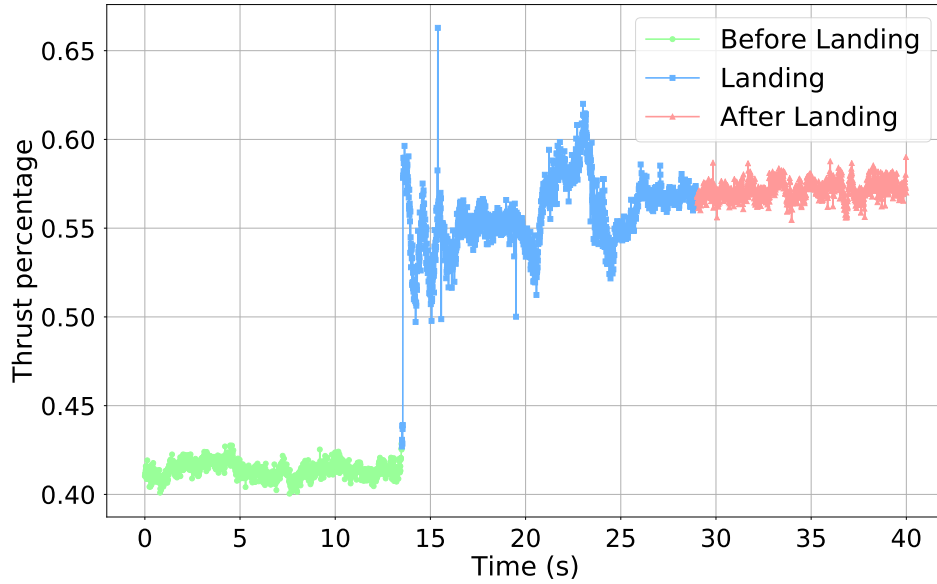
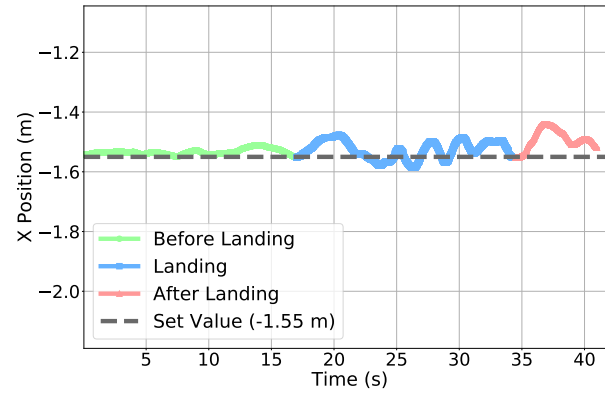


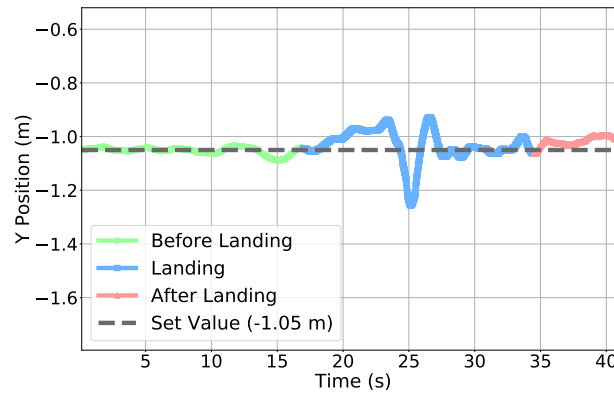
Figure 38: Thrust variation across the three landing phases.

Two distinct landing experiments were conducted to further evaluate the stability of the UAVC by applying our proposed NMPC, testing its ability to reject the very aerodynamic disturbances that could not be modeled in simulation and were the primary motivation for its design. In the first experiment, the NMPC framework was implemented without incorporating the dynamic model for the downwash flow. Fig. 39 depicts the variations in the UAVC's position along the X-, Y-, and Z-axes during the three defined phases of the landing process. Correspondingly, Table 9

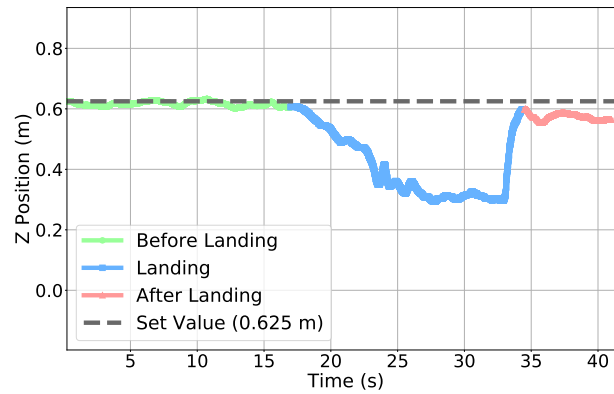
reports the RMSE values for each axis during the individual phases and over the entire landing procedure. Notably, the extraordinarily high RMSE in the Z-axis during landing indicates an obvious instability in that direction.



(a) X-axis position of UAVC



(b) Y-axis position of UAVC



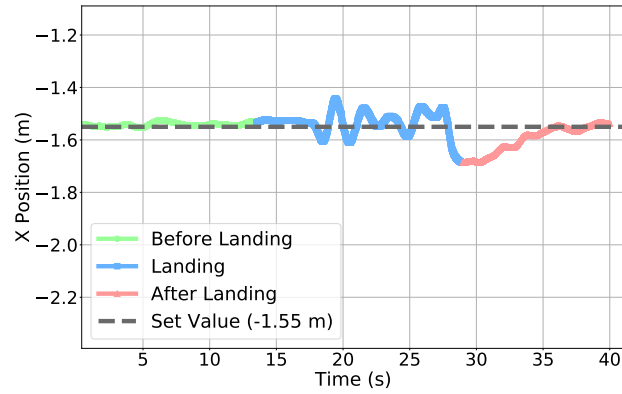
(c) Z-axis position of UAVC

Figure 39: Variations in the UAVC's position along the X-, Y- and Z-axes during the landing process when the dynamic model for the downwash flow is not considered.

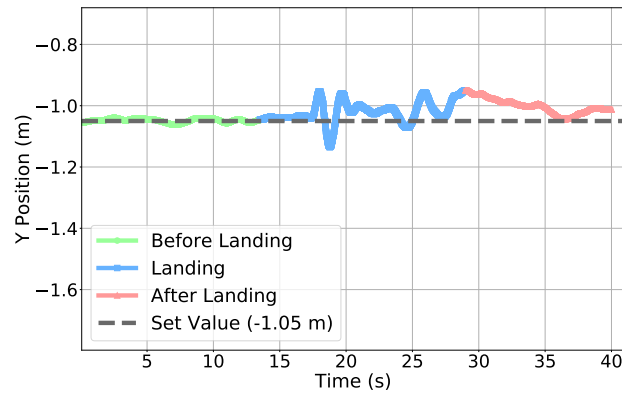
Table 9: RMSE of the UAVC's position during the landing process (without the dynamic downwash flow model).

	X-axis [cm]	Y-axis [cm]	Z-axis [cm]
RMSE before landing	2.12	1.29	1.38
RMSE during landing	3.99	6.24	24.18
RMSE after landing	7.21	3.77	5.50
RMSE (whole process)	4.11	4.42	15.95

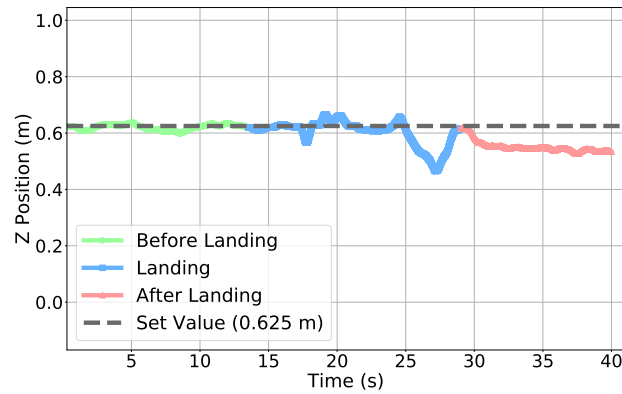
For the second experiment, the dynamic model for the downwash flow was integrated into the NMPC framework. Fig. 40 illustrates the changes in the UAVC's position along the X-, Y-, and Z-axes throughout the landing stages, while Table 10 provides the associated RMSE values for each phase. With the downwash model in the loop, the Z-axis RMSE over the entire landing dropped from 15.95 cm (no model) to 5.14 cm representing a 68% reduction. Furthermore, during the critical landing phase alone, it decreased from 24.18 cm to 5.06 cm, cutting vertical oscillations by nearly 79%. These results demonstrate that the downwash wind model considered in the NMPC framework enhances overall landing stability.



(a) X-axis position of UAVC



(b) Y-axis position of UAVC



(c) Z-axis position of UAVC

Figure 40: Variations in the UAVC's position along the X-, Y- and Z-axes during the landing process when the dynamic model for the downwash flow is considered.

Table 10: RMSE of the UAVC's position during the landing process (with the dynamic downwash flow model).

	X-axis [cm]	Y-axis [cm]	Z-axis [cm]
RMSE before landing	1.07	0.62	1.04
RMSE during landing	5.20	4.78	5.06
RMSE after landing	6.64	5.36	7.72
RMSE (whole process)	4.78	4.10	5.14

In summary, the detailed data analysis shows that the soft-constrained planner achieves precise path following with sub-decimeter accuracy, and that integrating a downwash wind model into the NMPC controller drastically reduces vertical oscillations, yielding stable and repeatable landings on a moving aerial carrier.

7 Conclusion

This thesis has systematically addressed the problem of automated landing of the UAV on a UAVC under disturbance conditions, with a focus on trajectory planning, control strategies, and experimental validation. Through theoretical analysis, simulation studies, and real-world experiments, several key contributions have been made, and potential directions for future research have been identified.

7.1 Summary of Contributions

First, this thesis proposes planning frameworks including hard-constrained and soft-constrained optimization for trajectory generation. The hard-constrained planner leverages JPS for global path planning and convex decomposition to construct SFCs, ensuring strict compliance with dynamic constraints (e.g., maximum velocity, acceleration) and obstacle avoidance through MIQP. In contrast, the soft-constrained planner adopts an improved A* algorithm with kinodynamic path search for global planning and B-spline optimization with well-designed cost functions for local refinement, balancing computational efficiency and trajectory smoothness via penalty-based cost functions. Comparative simulations demonstrate that the hard-constrained method guarantees a 100% success rate with strict feasibility, making it suitable for high-risk scenarios. While the soft-constrained approach achieves a higher average speed (2.07 m/s), lower computation time (12.76 ms), and an acceptable success rate (95%), making it suitable for the common real-world environments.

Second, an NMPC framework integrated with a dynamic downwash wind model is developed to enhance landing stability. The NMPC controller optimizes control

inputs in real time using a fourth-order Runge-Kutta method and SQP, effectively mitigating disturbances induced by the UAV’s downwash wind. Real-world experiments validate that incorporating the downwash model reduces the UAVC’s vertical position RMSE by 68% during landing, significantly improving stability. Third, comprehensive experimental validation is conducted, including simulated tests in Gazebo and real-world trials. Simulations verify the trajectory tracking accuracy of both planners (e.g., overall RMSE of 0.12 m for hard constraints and 0.17 m for soft constraints), while real-world experiments demonstrate the soft-constrained planner’s robustness with an overall tracking RMSE of 11.93 cm. These results collectively confirm the effectiveness of the proposed methods.

This work drives broader impacts across key domains. For persistent UAV missions, the UAVC (as a mobile “mother ship”) solves battery-induced endurance limits. It enables continuous operations by eliminating reliance on ground infrastructure, extending mission timelines from hours to days. In disaster response, the system’s ability to operate in unknown, cluttered environments lets the UAVC hover safely above disaster zones while deploying UAVs for survivor search or supply delivery. The NMPC controller ensures stable landings amid turbulence, which accelerates response and reduces human risk. While for remote-area logistics, the system bypasses poor ground infrastructure. It lets UAVCs serve as hubs for UAV deliveries (e.g., medical supplies), narrowing urban-rural service gaps.

7.2 Future Research

Future work will focus on several key directions to further enhance the proposed framework’s performance and expand its practical applicability. First, in the current implementation, the operator selects either the hard-constrained or soft-constrained planner before each mission. That choice is based on a risk assessment of the

flight profile together with the available computational resources. For operations judged to be high risk (e.g., flights over populated or safety-critical zones), the hard-constrained planner is used to enforce strict collision avoidance and to guarantee adherence to dynamic limits. By contrast, for routine inspection missions or when onboard processing capacity is limited, the soft-constrained planner is preferred because it delivers lower planning latency and smoother trajectories. While this manual selection method ensures clear accountability, it depends heavily on the operator's experience and preflight judgment. To remove this requirement in future work, an adaptive supervisor should be introduced that continuously monitors key metrics such as obstacle density, computing resource utilization, and the vehicle's distance from the landing platform. When environmental clutter or computation delays exceed preset thresholds, the supervisor will automatically switch from the soft-constrained planner to the hard-constrained one, and then switch back once the vehicle is operating in less demanding conditions. This automated approach will dynamically balance safety guarantees and computational efficiency throughout the mission, eliminating the need for real-time human intervention.

Second, computational efficiency and robustness remain critical areas for improvement. While the current soft-constrained and hard-constrained planners have demonstrated effectiveness in handling static obstacles and moderate disturbances, future research will incorporate dynamic factors such as moving obstacles, which are prevalent in real-world scenarios including dynamic urban environments and multi-agent operational spaces. This will involve developing real-time adaptive trajectory planning algorithms that can quickly perceive and respond to the motion of dynamic obstacles, possibly by integrating advanced perception modalities such as LiDAR and predictive models to forecast obstacle trajectories. Such enhancements will ensure the system maintains collision-free operations even in highly dynamic environments, thereby improving its robustness in unpredictable settings.

Additionally, extending the framework to support multi-UAV cooperative operations will be a core focus. The current work primarily addresses single-UAV landing on a UAVC, but many practical applications (e.g., large-scale surveillance, coordinated cargo delivery, or emergency response) require multiple UAVs to collaborate with a shared carrier. Future research will explore cooperative trajectory planning that generates collision-free, dynamically feasible paths for multiple UAVs. These paths will account for environmental constraints including obstacles and no-fly zones as well as dynamic interactions such as UAV-to-UAV collisions and downwash interference from nearby vehicles. Unlike single-UAV planning, multi-UAV coordination requires careful spatiotemporal management to ensure that the vehicles not only avoid crashes but also minimize mutual interference. To achieve this, a hybrid planning architecture combining centralized and distributed elements should be investigated. A central module will establish a global trajectory framework to prevent large-scale conflicts, while each UAV will run a local trajectory optimizer that adapts its path in real time based on on-board sensor data. Finally, the framework will integrate predictive models of UAV dynamics and downwash effects so that every trajectory fully accounts for the aerodynamic influence of one UAV on another.

Furthermore, adapting the system to varying environmental conditions will be investigated to broaden its applicability. The current framework has been validated under nominal wind conditions and controlled environments, but real-world operations often involve extreme weather such as strong crosswinds and turbulence or complex terrains. Moreover, robust control strategies, such as adaptive NMPC with online parameter tuning, will be developed to mitigate the impact of environmental uncertainties. These improvements will enable the system to operate reliably in diverse and harsh conditions, expanding its utility in practical scenarios ranging from offshore inspections to disaster relief.

In summary, by introducing an adaptive supervisor to choose a suitable planner,

advancing computational efficiency, enabling multi-UAV cooperation, and enhancing environmental adaptability, the proposed framework is expected to evolve into a more versatile and robust system, capable of addressing the complexities of real-world applications and contributing to the broader field of autonomous aerial robotics.

References

- [1] D. J. Pines and F. Bohorquez, “Challenges facing future micro-air-vehicle development,” *Journal of Aircraft*, vol. 43, no. 2, pp. 290–305, 2006.
- [2] F. Nex and F. Remondino, “UAV for 3D mapping applications: a review,” *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.
- [3] H. Shakhathreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,” *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.
- [4] B. Fan, Y. Li, R. Zhang, and Q. Fu, “Review on the technological development and application of UAV systems,” *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 199–207, 2020.
- [5] P. G. Fahlstrom, T. J. Gleason, and M. H. Sadraey, *Introduction to UAV systems*. John Wiley & Sons, 2022.
- [6] D. Erdos, A. Erdos, and S. E. Watkins, “An experimental UAV system for search and rescue challenge,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 5, pp. 32–37, 2013.
- [7] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, “An autonomous multi-UAV system for search and rescue,” in *Proceedings of the First Workshop on*

- Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, 2015, pp. 33–38.
- [8] M. Atif, R. Ahmad, W. Ahmad, L. Zhao, and J. J. Rodrigues, “UAV-assisted wireless localization for search and rescue,” *IEEE Systems Journal*, vol. 15, no. 3, pp. 3261–3272, 2021.
- [9] F. Betti Sorbelli, “UAV-based delivery systems: A systematic review, current trends, and research challenges,” *Journal on Autonomous Transportation Systems*, vol. 1, no. 3, pp. 1–40, 2024.
- [10] D. Dissanayaka, T. R. Wanasinghe, O. De Silva, A. Jayasiri, and G. K. Mann, “Review of navigation methods for UAV-based parcel delivery,” *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 1068–1082, 2023.
- [11] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.
- [12] Z. Yang, X. Yu, S. Dedman, M. Rosso, J. Zhu, J. Yang, Y. Xia, Y. Tian, G. Zhang, and J. Wang, “UAV remote sensing applications in marine monitoring: Knowledge visualization and review,” *Science of the Total Environment*, vol. 838, p. 155939, 2022.
- [13] H. Ren, Y. Zhao, W. Xiao, and Z. Hu, “A review of UAV monitoring in mining areas: Current status and future perspectives,” *International Journal of Coal Acience & Technology*, vol. 6, pp. 320–333, 2019.
- [14] R. S. De Moraes and E. P. De Freitas, “Multi-UAV based crowd monitoring

- system,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1332–1345, 2019.
- [15] B. Galkin, J. Kibilda, and L. A. DaSilva, “UAVs as mobile infrastructure: Addressing battery lifetime,” *IEEE Communications Magazine*, vol. 57, no. 6, pp. 132–137, 2019.
- [16] T. Zhao, Y. Zhang, M. Wang, W. Feng, S. Cao, and G. Wang, “A critical review on the battery system reliability of drone systems,” *Drones*, vol. 9, no. 8, p. 539, 2025.
- [17] M. N. Boukoherine, Z. Zhou, and M. Benbouzid, “A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects,” *Applied Energy*, vol. 255, p. 113823, 2019.
- [18] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, “Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends,” *Intelligent service robotics*, vol. 16, no. 1, pp. 109–137, 2023.
- [19] N. Nitta, F. Wu, J. T. Lee, and G. Yushin, “Li-ion battery materials: present and future,” *Materials today*, vol. 18, no. 5, pp. 252–264, 2015.
- [20] Encyclopedia.pub, “Battery Charging Techniques for Unmanned Aerial Vehicles,” <https://encyclopedia.pub/entry/24663>, 2025, accessed: 2025-07-20.
- [21] A. P. Sample, D. T. Meyer, and J. R. Smith, “Analysis, experimental results, and range adaptation of magnetically coupled resonators for wireless power transfer,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 544–554, 2010.

- [22] N. Shinohara, *Wireless power transfer via radiowaves*. John Wiley & Sons, 2014.
- [23] Z. Shen, C. Huang, H. Huang, Y. Wang, F.-Y. Wang, A. Jamalipour, D. T. Pham, L. Vlacic, and A. V. Savkin, “The emerging intelligent vehicles and intelligent vehicle carriers collaborative systems,” in *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2024, pp. 639–644.
- [24] K. Fujii, K. Higuchi, and J. Rekimoto, “Endless flyer: a continuous flying drone with automatic battery replacement,” in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. IEEE, 2013, pp. 216–223.
- [25] J. Wang, D. McKiver, S. Pandit, A. F. Abdelzaher, J. Washington, and W. Chen, “Precision UAV landing control based on visual detection,” in *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2020, pp. 205–208.
- [26] A. Antenucci, S. Mazzaro, A. Fiorilla, L. Messina, A. Massa, and W. Matta, “A ROS based automatic control implementation for precision landing on slow moving platforms using a cooperative fleet of rotary-wing UAVs,” in *2020 5th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2020, pp. 139–144.
- [27] P. H. Nguyen, M. Arsalan, J. H. Koo, R. A. Naqvi, N. Q. Truong, and K. R. Park, “Lightdenseyolo: A fast and accurate marker tracker for autonomous uav landing by visible light camera sensor on drone,” *Sensors*, vol. 18, no. 6, p. 1703, 2018.

- [28] G. Niu, Q. Yang, Y. Gao, and M.-O. Pun, “Vision-based autonomous landing for unmanned aerial and ground vehicles cooperative systems,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6234–6241, 2021.
- [29] C. G. Grlj, N. Krznar, and M. Pranjić, “A decade of UAV docking stations: A brief overview of mobile and fixed landing platforms,” *Drones*, vol. 6, no. 1, p. 17, 2022.
- [30] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, “Vision-based autonomous quadrotor landing on a moving platform,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 200–207.
- [31] S. Lange, N. Sünderhauf, and P. Protzel, “Autonomous landing for a multirotor UAV using vision,” in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN 2008)*, 2008, pp. 482–491.
- [32] V. Lippiello, R. Mebarki, and F. Ruggiero, “Visual coordinated landing of a UAV on a mobile robot manipulator,” in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2013, pp. 1–7.
- [33] Y. Feng, C. Zhang, S. Baek, S. Rawashdeh, and A. Mohammadi, “Autonomous landing of a uav on a moving platform using model predictive control,” *Drones*, vol. 2, no. 4, p. 34, 2018.
- [34] E. Barrett, M. Reiling, S. Mirhassani, R. Meijering, J. Jager, N. Mimmo, F. Callegati, L. Marconi, R. Carloni, and S. Stramigioli, “Autonomous battery exchange of UAVs with a mobile ground base,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 699–705.

- [35] P. R. Palafox, M. Garzón, J. Valente, J. J. Roldán, and A. Barrientos, “Robust visual-aided autonomous takeoff, tracking, and landing of a small UAV on a moving landing platform for life-long operation,” *Applied Sciences*, vol. 9, no. 13, p. 2661, 2019.
- [36] A. Paris, B. T. Lopez, and J. P. How, “Dynamic landing of an autonomous quadrotor on a moving platform in turbulent wind conditions,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9577–9583.
- [37] R. Polvara, S. Sharma, J. Wan, A. Manning, and R. Sutton, “Vision-based autonomous landing of a quadrotor on the perturbed deck of an unmanned surface vehicle,” *drones*, vol. 2, no. 2, p. 15, 2018.
- [38] T.-T. Nguyen, A. Crismer, G. De Cubber, B. Janssens, and H. Bruyninckx, “Landing UAV on moving surface vehicle: Visual tracking and motion prediction of landing deck,” in *2024 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2024, pp. 827–833.
- [39] P. M. Gupta, E. Pairet, T. Nascimento, and M. Saska, “Landing a UAV in harsh winds and turbulent open waters,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 744–751, 2022.
- [40] M. Aissi, Y. Moumen, J. Berrich, T. Bouchentouf, M. Bourhaleb, and M. Rahmoun, “Autonomous solar USV with an automated launch and recovery system for UAV: State of the art and design,” in *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*. IEEE, 2020, pp. 1–6.
- [41] G. Shao, Y. Ma, R. Malekian, X. Yan, and Z. Li, “A novel cooperative platform

- design for coupled USV–UAV systems,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4913–4922, 2019.
- [42] R. Miyazaki, R. Jiang, H. Paul, K. Ono, and K. Shimonomura, “Airborne docking for multi-rotor aerial manipulations,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4708–4714.
- [43] Z. Shen, G. Zhou, H. Huang, C. Huang, Y. Wang, and F.-Y. Wang, “Convex optimization-based trajectory planning for quadrotors landing on aerial vehicle carriers,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 138–150, 2023.
- [44] X. Dong, H. Li, Y. Cui, J. Xiang, D. Li, and Z. Tu, “Aerial landing of micro UAVs on moving platforms considering aerodynamic interference,” *IEEE Robotics and Automation Letters*, 2024.
- [45] L. Yang, J. Qi, J. Xiao, and X. Yong, “A literature review of UAV 3D path planning,” in *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE, 2014, pp. 2376–2381.
- [46] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Computer Communications*, vol. 149, pp. 270–299, 2020.
- [47] X. Bai, H. Jiang, J. Cui, K. Lu, P. Chen, and M. Zhang, “UAV path planning based on improved A* and DWA algorithms,” *International Journal of Aerospace Engineering*, vol. 2021, no. 1, p. 4511252, 2021.
- [48] S. M. LaValle and J. Kuffner, James J., “Rapidly-exploring random trees:

- Progress and prospects,” in *Algorithmic and Computational Robotics: New Directions*. Natick, MA, USA: A.K. Peters/CRC Press, 2001, pp. 303–307.
- [49] T. Siméon, J.-P. Laumond, and C. Nissoux, “Visibility-based probabilistic roadmaps for motion planning,” *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.
- [50] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [51] D. J. Webb and J. Van Den Berg, “Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5054–5061.
- [52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.
- [53] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, “A survey of path planning algorithms for mobile robots,” *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.
- [54] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.
- [55] L. Campos-Macías, D. Gómez-Gutiérrez, R. Aldana-López, R. de la Guardia, and J. I. Parra-Vilchis, “A hybrid method for online trajectory planning of mobile robots in cluttered environments,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 935–942, 2017.

- [56] C. Richter, A. Bry, and N. Roy, “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments,” in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 649–666.
- [57] R. Deits and R. Tedrake, “Efficient mixed-integer planning for UAVs in cluttered environments,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 42–49.
- [58] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.
- [59] M. W. Mueller, M. Hehn, and R. D’Andrea, “A computationally efficient motion primitive for quadcopter trajectory generation,” *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.
- [60] L. Blackmore, M. Ono, and B. C. Williams, “Chance-constrained optimal path planning with obstacles,” in *Robotics: Science and Systems (RSS)*, 2011.
- [61] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [62] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, “Faster: Fast and safe trajectory planner for navigation in unknown environments,” *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 922–938, 2021.
- [63] T. Schouwenaars, B. M. P. How, E. Feron, and J. P. Morrison, “Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees,” in *AIAA Guidance, Navigation, and Control Conference*, 2001.

- [64] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “Chomp: Covariant hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
- [65] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “Stomp: Stochastic trajectory optimization for motion planning,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4569–4574.
- [66] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online UAV replanning,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5332–5339.
- [67] 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 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 215–222.
- [68] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, “Robust and efficient quadrotor trajectory generation for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.
- [69] K. Qin, “General matrix representations for B-Splines,” in *Pacific Graphics* '98, 1998, pp. 37–43.
- [70] F. Gao, Y. Lin, and S. Shen, “Gradient-based online safe trajectory generation for quadrotor flight in complex environments,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3681–3688.

- [71] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, “Ego-planner: An ESDF-free gradient-based local planner for quadrotors,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.
- [72] P. Pounds, R. Mahony, and P. I. Corke, “Modelling and control of a quad-rotor robot,” *Proceedings of the 2006 Australasian Conference on Robotics and Automation*, 2006.
- [73] G. Hoffmann, S. Waslander, and C. Tomlin, “Quadrotor helicopter trajectory tracking control,” in *AIAA guidance, Navigation and Control Conference and Exhibit*, 2008, p. 7410.
- [74] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3277–3282.
- [75] S. Lupashin, A. Schöllig, M. Sherback, and R. D’Andrea, “A simple learning strategy for high-speed quadrocopter multi-flips,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 1642–1648.
- [76] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.
- [77] A. Bemporad and M. Morari, “Robust model predictive control: A survey,” in *Robustness in Identification and Control*. Springer, 2007, pp. 207–226.
- [78] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.

- [79] B. Kouvaritakis and M. Cannon, “Model predictive control,” *Switzerland: Springer International Publishing*, vol. 38, no. 13-56, p. 7, 2016.
- [80] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, “A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight,” *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.
- [81] M. Bangura and R. Mahony, “Real-time model predictive control for quadrotors,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.
- [82] M. Kamel, M. Burri, and R. Siegwart, “Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.
- [83] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, “Model predictive control for micro aerial vehicles: A survey,” in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1556–1563.
- [84] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, “Fast nonlinear model predictive control for unified trajectory optimization and tracking,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1398–1404.
- [85] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, “Fast nonlinear model predictive control for multicopter attitude tracking on $SO(3)$,” in *2015 IEEE Conference on Control Applications (CCA)*. IEEE, 2015, pp. 1160–1166.
- [86] G. Garimella, M. Sheckells, and M. Kobilarov, “Robust obstacle avoidance for aerial platforms using adaptive model predictive control,” in *2017 IEEE*

- International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5876–5882.
- [87] P. Foehn, A. Romero, and D. Scaramuzza, “Time-optimal planning for quadrotor waypoint flight,” *Science robotics*, vol. 6, no. 56, p. eabh1221, 2021.
- [88] A. Murilo and R. V. Lopes, “Unified NMPC framework for attitude and position control for a VTOL UAV,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 233, no. 7, pp. 889–903, 2019.
- [89] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, “Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs,” *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.
- [90] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, “Data-driven MPC for quadrotors,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.
- [91] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, “Aerial navigation in obstructed environments with embedded nonlinear model predictive control,” in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3556–3563.
- [92] D. Harabor and A. Grastien, “Online graph pruning for pathfinding on grid maps,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, 2011, pp. 1114–1119.
- [93] Gurobi Optimization, Limited Liability Company, “Gurobi Optimizer Reference Manual,” 2024. [Online]. Available: <https://www.gurobi.com>

- [94] K. Qin, “General matrix representations for B-splines,” in *Proceedings Pacific Graphics’ 98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No. 98EX208)*. IEEE, 1998, pp. 37–43.
- [95] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, D. Wheeler, and A. Y. Ng, “ROS: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.
- [96] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, “Model predictive control for trajectory tracking of unmanned aerial vehicles using Robot Operating System,” *Robot Operating System (ROS) The Complete Reference (Volume 2)*, pp. 3–39, 2017.
- [97] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, “qpOASES: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.
- [98] Open Robotics, “Gazebo simulator,” <https://gazebo.org/>, 2025, accessed: 2025-08-03.
- [99] Intel Corporation, “Intel RealSense Depth Camera D435 Datasheet,” Intel Corporation, Datasheet, 2022, accessed: 20 July 2025. [Online]. Available: <https://www.intelrealsense.com/wp-content/uploads/2019/05/Intel-RealSense-D400-Series-Datasheet.pdf>
- [100] M. Tech, “Nxtpx4v2 flight controller,” https://micoair.com/flightcontroller_nxtpx4v2/, 2024, accessed: 20 July 2025.
- [101] Holybro, “Pixhawk 6c mini technical specification,” <https://docs.holybro.com/autopilot/pixhawk-6c-mini/technical-specification>, 2024, accessed: 20 July 2025.