

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.

TIME-OPTIMAL MOTION PLANNING FOR
AUTONOMOUS FLIGHT USING SPATIAL
REFORMULATION

CHAN YIN YUEN

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Aeronautical and Aviation Engineering

Time-optimal motion planning for autonomous flight using
spatial reformulation

Chan Yin Yuen

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

July 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)

Chan Yin Yuen (Name of student)

Abstract

The advancement of Unmanned Aerial Vehicles (UAVs), particularly quadcopters, has facilitated their adoption in a wide range of applications, including search-and-rescue operations, surveillance, and agricultural automation. Among these, urban drone delivery has emerged as a key aspect of Urban Air Mobility (UAM), with the potential to revolutionize urban logistics by alleviating vehicular traffic congestion and reducing reliance on human labor in distribution networks.

A critical challenge in drone delivery lies in ensuring efficient and reliable navigation within dense and complex urban environments. These environments are characterized by limited airspace, numerous static and dynamic obstacles, restricted no-fly zones, and rapidly changing conditions, all of which complicate motion planning. Furthermore, the constraints of limited battery capacity, coupled with the anticipated growth in demand, underscore the necessity of minimizing flight time to achieve operational efficiency and scalability.

To address these challenges, this thesis proposes a time-optimal motion planning framework for drone delivery systems, specifically for quadcopters and cable-suspended quadcopter (CSPQ) systems. We present a complete system design with novel methodologies, with specific focuses on aspects that have been underexplored.

Firstly, the mathematical models of both systems are developed. For the CSPQ system in particular, we investigate the integration of physics-based modeling with deep learning techniques, as conventional physics-based models fail to fully capture the complex, nonlinear behaviors of CSPQ systems, such as downwash on the payload and ground effects. Although this approach is not adopted in the subsequent framework, it lays the groundwork for enabling neural network-based models in motion planning.

Secondly, we introduce an efficient global path planning method with a focus on homotopy paths, which are essential for subsequent trajectory

optimization but have often been overlooked. Additionally, the effect of wind dynamics in urban environments on the global path is also considered.

Thirdly, based on the predefined global path, we develop a novel path-parametric method for three key motivations: (1) it captures the natural progression along the path, (2) it integrates the geometric characteristics of the path, such as curvature and torsion into the dynamics, and (3) it transforms spatial constraints into convex formulations in the orthogonal components of the spatial states. We adopt a Gravity-Normal (GN) frame spatial reformulation method, which outperforms the conventional Frenet-Serret frame and Parallel-Transport frame in UAV scenarios. Subsequently, the GN frame allows us to define a flight corridor easily and intuitively, paving the way for optimal and safe trajectory optimization of the system.

Lastly, we describe the full framework in detail, including the offline planning module arising from previous discussions and the online trajectory tracking module. This thesis presents extensive experimental results, comparative analyses, and a detailed description of the system setup. In conclusion, we reflect on the motivations driving each chapter, highlight the existing limitations in aerial autonomy development, and outline potential research directions informed by our experiences.

Supervisor: Dr Kam K.H. Ng

Co-supervisor: Prof. Dr Li-Ta Hsu

Publications arising from the thesis

Journal articles (in reverse chronological order)

Chan, Y. Y., Ng, K. K. H., Wang, T., Hon, K. K., & Liu, C. H. (2025). Near time-optimal trajectory optimisation for drones in last-mile delivery using spatial reformulation approach. *Transportation Research Part C: Emerging Technologies*, 171, 104986. **(2025 SCI IF: 7.6, JCR: Q1)**

Chan, Y. Y., Ng, K. K. H., Lee, C. K. M., Hsu, L. T., & Keung, K. L. (2023). Wind dynamic and energy-efficiency path planning for unmanned aerial vehicles in the lower-level airspace and urban air mobility context. *Sustainable Energy Technologies and Assessments*, 57, 103202. **(2025 SCI IF: 7.1, JCR: Q1)**

Manuscript under review

Chan, Y. Y., Shu Y. X., Chen, W. H. & Ng, K. K. H. (2025). Time-Optimal Motion Planning of Cable-Suspended Payload Quadcopters Using Spatial Reformulation. *Aerospace Science and Technology*. **(under 1st review)**.

Conference paper (in reverse chronological order)

Chan, Y. Y., Ng, K. K. H., Liu, Y. (2025). Dynamic Modeling of Cable-Suspended Payload Quadcopter: A Physics-Informed Deep Learning Approach. In *AIAA AVIATION FORUM AND ASCEND 2025*.

Chan, Y. Y., Ng, K. K. H., Liu, C. H., & Hon, K. K. (2024). Wind Dynamics and Homotopy Path Planning for Unmanned Aerial Vehicle Trajectory Optimisation. In *AIAA AVIATION FORUM AND ASCEND 2024* (p. 4206).

Journal articles that earned honourable mention (in reverse chronological order)

Jin, Z., Ng, K. K., Zhang, C., **Chan, Y. Y.**, & Qin, Y. (2025). A multistage stochastic programming approach for drone-supported last-mile humanitarian logistics system planning. *Advanced Engineering Informatics*, 65, 103201. **(2025 SCI IF: 8.0, JCR: Q1)**

Keung, K. L., **Chan, Y. Y.**, Ng, K. K., Mak, S. L., Li, C. H., Qin, Y., & Yu, C. W. (2022). Edge intelligence and agnostic robotic paradigm in resource synchronisation and sharing in flexible robotic and facility control system. *Advanced Engineering Informatics*, 52, 101530. **(2025 SCI IF: 8.0, JCR: Q1)**

Acknowledgements

First and foremost, I would like to express my gratitude and appreciation to my chief supervisor, Prof Kam K.H. NG, Associate Professor of the Department of Aeronautical and Aviation Engineering at The Hong Kong Polytechnic University. The priceless time he spent with me during this research is gratefully appreciated. Prof Ng’s dedication in facilitating me with guidance and assistance has been invaluable. He has also provided me with valuable opportunities to improve my research abilities and interpersonal skills. I have benefited intellectually from his insights and experiences as my supervisor, a friend, a family member, and a life mentor.

I would also like to deliver my heartfelt thanks to my co-supervisor, Professor Li-Ta HSU, for encouraging and guiding me throughout my thesis. Additionally, I express my gratitude to Prof. Wen-Hua CHEN and Dr. Tianqi WANG, for enriching my learning experience by sharing his knowledge, research experiences, and future path in academia.

My heartfelt gratitude extends to all my groupmates, especially Dr Qinbiao Li, Dr Nana Chu, Dr Chengliang Zhang, Ye Liu, Zhongyi Jin, Yipeng Zhu, Xin Yuan, Cho Yin Yiu, Anson Yan, Congcong Du, Changxin Zhu, Rong Tang, Qinyu Zhou, Yuanyuan Bu, Qianyi Cui, Senna Yao, Tam Tsz Kin (so many close partners) in the “FJ005/Prof Ng’s research group” at the Hong Kong Polytechnic University. Over the past few years, their invaluable support and engaging discussions have been truly meaningful. Their laughter during challenging times has been a source of comfort and positivity, and I deeply appreciate their insightful suggestions and thoughtful conversations. Their constant encouragement and unwavering support have been instrumental in completing this research. I also wish to thank our dedicated technician, Yin Chiu KAN, for his efficient assistance in my daily work.

Throughout my life, my beloved family has been a constant source of love and encouragement, inspiring me to pursue my dreams. Their unconditional

love, steadfast support, and boundless patience have been my guiding light. I am profoundly grateful for their unwavering belief in me, which has been the driving force behind my academic journey.

Contents

Abstract	i
Publications arising from the thesis	iii
Acknowledgements	v
List of Figures	xiv
List of Tables	xv
List of Abbreviations	xvi
1 Introduction	1
1.1 Research background	1
1.1.1 Drone Delivery and Low-Altitude Economy	1
1.1.2 Time-Optimal Motion Planning	2
1.2 Motivation	4
1.2.1 Enhancing the Accuracy of UAV System Modeling	4
1.2.2 Incorporating Homotopy Awareness into Global Path Planning	5
1.2.3 Exploring Spatial Representations for Efficient and Safe UAV Motion Planning	6
1.3 Objectives and Thesis Structure	9
1.3.1 Objectives	9
1.3.2 Thesis Structure	10
2 Literature Review	12
2.1 Urban Air Mobility	12
2.2 UAV Modeling and System Identification	13
2.3 UAV Motion Planning Methods	14

2.3.1	Global Planning	14
2.3.2	Local Planning	16
2.4	Spatial Reformulation	19
2.5	Concluding Remarks	20
3	UAV modeling	22
3.1	Notation	23
3.2	Quadcopter Dynamics	24
3.2.1	Newton-Euler Approach	24
3.3	CSPQ Dynamics	28
3.3.1	Newton-Euler Approach	28
3.3.2	Euler-Lagrange Formulation	30
3.3.3	CSPQ Dynamics using PIATCN	36
3.3.3.1	Model Learning	36
3.3.3.2	Network Structure	37
3.3.3.3	Physics-inspired Loss Function	38
3.3.3.4	Results	39
3.3.3.5	Data Acquisition	40
3.3.3.6	Model Training and Prediction Performance	42
3.4	Concluding Remarks	44
4	Homotopy-based Trajectory Planner	45
4.1	Introduction	45
4.2	Methodology	48
4.2.1	Homotopy Class Path Planning	48
4.2.1.1	Definition of Homotopy	48
4.2.2	Augmented Lazy Theta* with Penalty Algorithm	51
4.2.3	TEB with Consideration of Wind Dynamics	54
4.3	Results	57
4.3.1	Augmented Lazy Theta* with Penalty Algorithm Performance	58
4.3.2	TEB with Consideration of Wind Dynamics Performance	60

4.3.3	Experimental Validation under ROS and Gazebo Simulation	62
4.4	Concluding Remarks	63
5	Path-Parametric Reformulation Method	65
5.1	Introduction	65
5.2	Path-Parametric Reformulation	67
5.2.1	Preliminaries on Conventional Curve-Framing Methods	68
5.2.1.1	Frenet-Serret Frame	69
5.2.1.2	Parallel Transport Rrame	72
5.2.2	Gravity-Normal Frame	75
5.2.3	Spatial Reformulation	80
5.3	Flight Corridor Formulation	83
5.3.1	Rationale	85
5.3.2	Determining the Dominant Side	87
5.3.3	Line Update	88
5.3.4	Flight Corridor Generation on $\mathbf{e}_2$ Dominant Side . . .	91
5.3.5	Flight Corridor Extension on $\mathbf{e}_2$ Dominant Side . . .	92
5.3.6	Flight Corridor Extension on $\mathbf{e}_3$ Dominant Side . . .	94
5.4	Concluding Remarks	96
6	Time-Optimal Planning Framework	98
6.1	Introduction	98
6.2	Hierarchical Framework	99
6.2.1	Time-Optimal Trajectory Generation	100
6.2.1.1	Direct Multiple Shooting for Global Time-Optimal Trajectory	101
6.2.2	Trajectory Tracking	103
6.2.2.1	Trajectory Handler	103
6.2.2.2	Nonlinear Model Predictive Control	103
6.3	Numerical Results	104
6.3.1	Simulation Setup	104
6.3.2	Global Optimization Results	105

6.3.2.1	Analytical Trajectories	111
6.3.2.2	Bezier Curve Trajectories	111
6.3.2.3	Flight Corridor Analysis	115
6.3.3	NMPC Results	116
6.3.3.1	Bezier Curve Trajectoryies	116
6.3.4	Urban Environment	117
6.3.4.1	A Simplified Scenario	118
6.3.4.2	Full Urban Scenario	123
6.4	Concluding Remarks	124
7	Conclusion	126
7.1	Conclusion of the thesis	126
7.2	Areas of future research	127

List of Figures

1.1	Illustration of quadcopter-based systems (credit from [Graham, 2019]). (a) Direct attaching the payload (b) Cable-Suspended Payload Quadcopter (CSPQ)	4
1.2	Summary of the objectives	10
3.1	Configuration of the Quadcopter in the study along with the inertial and body frame	24
3.2	CSPQ system setup	27
3.3	Configuration of the CSPQ system in the study, along with the inertial and body frame.	30
3.4	The overall methodological framework of the PIATCN model with attention mechanism for the state prediction function, $\hat{\mathcal{F}}_{\text{PIATCN}}$	36
3.5	The modified Iris quadcopter model in Gazebo.	40
3.6	Families of the trajectories executed by the CSPQ system during data acquisition.	41
3.7	Prediction of the vehicle position, linear velocity, orientation, and the position of the payload as a function of time for a single sequence of the test set. The black dotted line indicates the separation between past state and state prediction	41
4.1	(a) Top view of the selected terrain for the study. (b) Heightmap of the selected terrain	49
4.2	τ_1 and τ_2 are in different homotopy classes, where τ_2 and τ_3 are in the same homotopy class.	49
4.3	The 2D occupancy map at altitudes of 30m, 60m, 90m, 120m, 150 m, and 200 m	58

4.4	(a) Paths in 17 distinct homotopy classes. The blue path represents the shortest path. (b) Paths in 4 homotopy classes at altitude 150 m. The blue path represents the shortest path	59
4.5	Wind velocity distributions over the area of interest	60
4.6	(a) TEB optimized trajectory in 17 distinct homotopy classes at altitude 60 m. The blue trajectory represents the shortest path, which belongs to different homotopy classes to the best path in Fig. 4.4(a). (b) TEB optimized trajectory in 4 distinct homotopy classes at altitude 150 m. The blue trajectory represents the shortest path, which is the same homotopy class to the best path in Fig. 4.4(b)	61
4.7	(a) Screenshot of the gazebo GUI showing the simulated environment of the area of interest. (b) The top view of the simulation terrain using the OpenStreetMap data	62
5.1	Illustration of Frenet-Serret curve framing. (a) Example Frenet frame in a 2D curve (b) Example Frenet frame in a 3D curve	71
5.2	Illustration of Frenet-Serret curve framing. (a) The Frenet frame is undefined and changes abruptly at the straight line region. (b) The Parallel Transport frame is smooth throughout.	74
5.3	Spatial projection of the quadcopter's center onto the reference path $\Gamma(s)$. The resulting transverse distance $\mathbf{w}$ is expressed in the GN frame at s , especially along the two transverse axes $\mathbf{e}_2$ (green) and $\mathbf{e}_3$ (blue).	75
5.4	<i>Top</i> : comparison of the (a) FS, (b) PT and (c) GN frames on a lemniscate with height variation. <i>Bottom</i> : comparison of the sample flight corridor generated in (d) FS, (e) PT and (f) GN frames.	81
5.5	(a) The angle between $\mathbf{e}_1$ and xy plane in different frame construction methods. (b) The angle between $\mathbf{e}_1$ and the unit gravity vector $\mathbf{k}$	81
5.6	illustration of boundary update for a point	84

5.7	Illustration of the dominating side of the obstacle with respect to the flight corridor. (a) the dominant side at $\mathbf{e}_2$ (b) the dominant side at $\mathbf{e}_3$	87
5.8	Illustration of the safety corridor generation process	89
5.9	Illustration of a sample flight corridor in full 3D scenario with obstacles on $\mathbf{e}_2$ side	92
5.10	Close-up view on obstacle $\mathbf{O}_j$ during extension	93
5.11	Minimum distance between the top surface and the reference curve (shown as magenta line)	95
5.12	Close-up view on obstacle $\mathbf{O}_k$ during extension on $\mathbf{e}_2$ dominant side	95
6.1	The overall framework for the time-optimal motion planning problem	100
6.2	Illustration of the Direct Multiple Shooting method. The discretized state trajectory $\{\mathbf{x}\}$ contains $N + 1$ knot points and N intervals, and the control trajectory $\{\mathbf{u}\}$ is applied piecewise over these intervals. To ensure continuity, at each knot point, the state trajectory is computed by numerically integrating the state-space equation, and equality constraints are enforced between the integrated state and the corresponding next knot point.	102
6.3	The simulated CSPQ system model. The system consists of an IRIS quadcopter, a massless tether, and a spherical payload.	105
6.4	(a), (b) and (c): comparison of the FS, PT and GN frames on scenario 1 respectively. (d): Trajectory states plots resulting from the three frame constructions in scenario 1. (e), (f), (g): comparison of the FS, PT and GN frames on scenario 2 respectively. (h): Trajectory states plots resulting from the three frame constructions in scenario 2, where the FS frame fails to converge.	106

6.5	The resultant global time-optimal trajectory obtained for our proposed GN frame construction method in scenario 3. <i>left</i> : the isometric view showing the trajectory, with the gray tunnel visualizing the flight corridor generated in the GN frame. The color indicates speed in m/s. <i>Top right</i> : the top view of the trajectory. <i>Bottom right</i> : the side view of the trajectory. . . .	109
6.6	Jacobian (<i>left</i>) and Hessian (<i>right</i>) for scenario 3 for illustration	111
6.7	(a) and (h): the 2D projection of the computed trajectories on the xy-plane and the yz-projection in scenario 3. (b), (c) and (d): the plot for the progress variable and the two transverse coordinates over time. (e), (f) and (g): linear velocities over time. (i), (j) and (k): Euler angles over time. (l) and (m): α and β over time.	112
6.8	Comparison of the sample flight corridor generated in (a) FS, (b) PT and (c) GN frames along scenario 3.	115
6.9	NMPC Trajectory tracking response for Scenario 3	116
6.10	Graphical illustration of two test areas)	117
6.11	Visualisation of the resulting flight corridor	118
6.12	Velocities and angles plots	120
6.13	Obstacle avoidance	120
6.14	Graphical illustration of the urban environment	122
6.15	Graphical illustration of the paths generated for the 2 OD pairs	122

List of Tables

3.1	Comparison of RMSE Errors Between PI-TCN and Baseline Models	42
4.1	Statistics of the searched paths in different homotopy classes at altitudes of 60m and 150m.	59
6.1	CSPQ Parameters and Constraints	110
6.2	Performance comparison across all scenarios. The initial arc-length for scenario 1, 2 and 3 are 20.88, 19.60m and 272.3m respectively.	110
6.3	Comparison of Time-Optimal and NMPC Approaches for Two OD Pairs	121

List of Abbreviations

ACO	Ant Colony Optimization
ALSTM	An LSTM encoder-decoder model with attention
BVP	Boundary Value Problem
CFD	Computational Fluid Dynamics
ConOps	Concept of Operation
CSPQ	Cable-Suspended Payload Quadcopter
DDP	Differential Dynamic Programming
DNNs	Deep Neural Networks
DTEB	Distributed Timed Elastic Band
ENU	East-North-Up
ERK4	Explicit 4th-order Runge-Kutta
FAA	Federal Aviation Administration
FLU	Forward-Left-Up
FS	Frenet-Serret
GN	Gravity-Normal
HPIPM	High-Performance Interior Point Method
iLQR	Iterative Linear-Quadratic Regulator
IPOPT	Interior Point Optimizer
IVP	Initial Value Problem
JPS	Jump Point Search
LQR	Linear-Quadratic Regulator
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MPCC	Model Predictive Contouring Control
MPC	Model Predictive Control
NED	North-East-Down
NLP	Nonlinear Programming Problem
NMPC	Nonlinear Model Predictive Control

OCID	Observer/Controller Identification
OCP	Optimal Control Problem
OKID	Observer/Kalman Identification
PIALSTM	Physics-Informed Attention-Based LSTM
PIATCN	Couldn't read definition
PINN	Physics-Informed Neural Network
PMP	Pontryagin's Minimum Principle
PRM	Probability Road Map
PT	Parallel Transport
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
ROS	Robot Operating System
RRT	Rapid-Exploration Random Tree
RTI	Real-Time Iteration Scheme
SFC	Safe Flight Corridor
SITL	Software-In-The-Loop
SQP	Sequential Quadratic Programming
TCN	Temporal Convolution Network
TEB	Timed Elastic Band
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

In recent years, drone technology has seen rapid advancements, particularly in the context of autonomous aerial systems operating in urban environments. Among the most promising applications is drone-based delivery, which is emerging as a transformative solution to the last-mile logistics challenge. However, implementing reliable and time-efficient UAV delivery systems in dense urban airspace presents several engineering and algorithmic challenges. These include dynamic system modeling for load transportation, global path planning in complex obstacle-laden environments, and generating safe, time-optimal trajectories.

This chapter introduces the foundational concepts, outlines key challenges, and sets the stage for the research and framework proposed in this thesis.

1.1 Research background

1.1.1 Drone Delivery and Low-Altitude Economy

Over the past decade, UAVs have become extensively used in many fields and applications. In the military field, UAVs have been used for surveillance, reconnaissance, and combat operations [Gowroju and Santhosh Ramchander, 2023]. In the civilian field, there are various UAV applications such as surveillance and inspection missions, precision agriculture, and search and rescue in disaster environments [AL-Dosari et al., 2023].

One notable drone application would be drone-based delivery, which has begun to emerge for the future of e-commerce, the logistics sector, and as a crucial component of smart cities [Moshref-Javadi and Winkenbach, 2021]. It is widely considered a novel solution to the last-mile delivery problem, and the use of drones to deliver items and parcels has been tested by well-known organizations such as UPS, FedEx, Uber, and Amazon [Raivi et al., 2023, Silva et al., 2023].

In fact, last-mile parcel delivery, together with air taxi and air metro are the three most challenging use cases in the Urban Air Mobility (UAM) concept, as defined by NASA in 2019 [Hasan, 2019]. NASA estimated that possible implementation will be in operation by 2030, and all three areas of interest require advanced route planning and precise navigation systems. It is essential to enable safe and efficient aircraft movement during all phases of operations [Sridhar et al., 1998], and the UAM must be able to handle on-demand, high-volume, and short-range flights near urban airspace. [Gugan and Haque, 2023].

In the context of the UAM concept, drones operate in low-altitude situations because of the constrained airspace in metropolitan areas. In the meantime, the urban environment is complicated with more high-rise structures and geofences, such as restricted and altitude zones. UAV flight paths must adhere to regional flight regulations and reasonably avoid obstructions [Qadir et al., 2021]. The lack of study in large-scale complicated environments makes it essential to suggest a new path design, as the existing research is restricted to small and basic scenarios [Ma and Chen, 2022].

1.1.2 Time-Optimal Motion Planning

To facilitate large-scale UAV-based delivery operations, one of the key enabling components is the system’s autonomous capability. An autonomous UAV system typically comprises three primary modules: guidance, navigation, and control [Marshall et al., 2021]. The guidance module is responsible for planning a global path from the origin to the destination, ensuring that the route remains within feasible airspace and avoids obstacles. The navigation module determines the UAV’s pose, velocity, and other relevant state variables along the planned route, effectively generating a feasible flight trajectory. The control module produces the control commands necessary for the UAV to accurately follow this trajectory—commonly referred to as the tracking system. In this study, we assume that the onboard tracking controller is sufficiently capable of driving the UAV along a desired trajectory with high fidelity.

This work focuses on the guidance and navigation modules, with particular emphasis on generating a global reference path and computing the corresponding time-optimal flight trajectory. In the literature, this process is frequently referred to as a two-stage motion planning framework, where the guidance module first generates a collision-free geometric path, which is then refined into a dynamically feasible and smooth trajectory through trajectory optimization in the navigation module [Zhou et al., 2019].

The resulting trajectory must satisfy several constraints: (1) the UAV’s kinematic and dynamic equations, (2) actuator limitations, such as maximum thrust, and (3) collision avoidance with environmental obstacles [Spedicato and Notarstefano, 2017]. The optimization objective is to minimize total mission time or, alternatively, energy consumption—aligning with the operational goal of providing fast and efficient point-to-point delivery services. The limited onboard battery capacity further constrains both flight duration and payload capacity, making time-optimal planning a critical requirement.

It is worth noting that a time-optimal trajectory is not always equivalent to the energy-optimal one. Time-optimal control often leads to bang-bang behaviors, where control inputs rapidly switch between their limits, potentially increasing energy usage. Nonetheless, due to the strong correlation between travel time and energy consumption under many practical conditions [Luo et al., 2022], minimizing time remains a preferred strategy, so long as it does not introduce redundancy or inefficiencies into the cost function.

The objective of this study is to enable fast and safe payload transport by computing a time-optimal trajectory based on a predefined global path provided by any suitable global planner [Zhou et al., 2025]. This problem is naturally formulated as an Optimal Control Problem (OCP), where the solution trajectory must remain within a safe flight corridor while satisfying all system constraints. Compared to traditional methods, the OCP framework is particularly well-suited for this task, as it allows for the explicit incorporation of state and input constraints within its mathematical formulation [Geisert and Mansard, 2016].

While OCPs were traditionally solved offline, recent advances in optimization and control theory have enabled online trajectory generation via

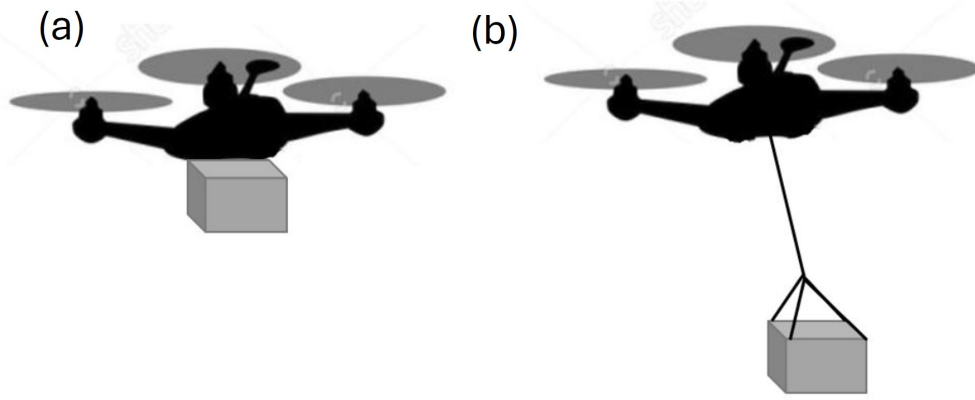


Figure 1.1: Illustration of quadcopter-based systems (credit from [Graham, 2019]). (a) Direct attaching the payload (b) Cable-Suspended Payload Quadcopter (CSPQ)

Model Predictive Control (MPC) [Mayne et al., 2000], with successful applications in real-time quadcopter control [Wang et al., 2024, Sarvaiya et al., 2025]. These developments motivate the use of time-optimal OCP formulations in the context of UAV motion planning, particularly for time-sensitive applications such as autonomous aerial delivery.

1.2 Motivation

To enable time-optimal and reliable UAV motion planning in complex urban environments, we observed several key challenges that must be addressed. They are listed as follows:

1.2.1 Enhancing the Accuracy of UAV System Modeling

The trajectory planner and controller must ensure rapid payload transport while minimizing swing-induced instability [Rao et al., 2023]. A critical requirement for achieving this is the development of an accurate dynamic model of the UAV system. In this thesis, we limit our scope to two quadcopter-based systems: the direct attachment and the Cable-Suspended Payload Quadcopters (CSPQ) as shown in Fig. 1.1.

While physics-based modeling can capture the primary behavior of the platform [Guerrero et al., 2015], the system is subject to highly nonlinear phenomena, such as aerodynamic forces and torques, propeller–wake interactions, structural vibrations, and downwash effects on the payload. They are inherently difficult to model analytically [Saviolo et al., 2022].

Traditional modeling approaches rely on experimental input–output data to construct mathematical representations of system dynamics, typically in the form of state-space models or transfer functions [Shi et al., 2019]. These methods serve to compensate for modeling inaccuracies caused by system nonlinearities, uncertainties, and external disturbances. However, such approaches are often constrained to collections of linear models designed for specific operating conditions and thus lack the capacity to fully capture the complex nonlinear behaviors exhibited by UAV systems.

Successfully learning accurate system dynamics directly from data holds significant potential for advancing robotic system development [Saviolo et al., 2022]. It enables rapid model identification and supports high-performance control across a wide range of operating conditions, with the scalability to generalize across diverse platforms [Shi et al., 2019]. Accordingly, the goal of this work is to fully exploit the expressive capacity of deep neural networks to learn the quadrotor’s dynamics purely from observational data, without relying on explicit analytical modeling [Torrente et al., 2021].

1.2.2 Incorporating Homotopy Awareness into Global Path Planning

A homotopy class of trajectories is defined as the set of continuous trajectories connecting the same start and end points, which can be smoothly deformed into one another without intersecting any obstacles in the environment. The presence of obstacles naturally gives rise to multiple distinct homotopy classes. [Bhattacharya, 2010] In the context of motion planning, homotopy class constraints refer to either restricting the planned trajectory to specific allowed homotopy classes or explicitly forbidding certain ones.

Homotopy class constraints arise naturally in path planning problems.

For example, to deploy a group of agents to explore an environment, the agent should avoid exploring the same homotopy class for optimized efficiency [Lluvia et al., 2021]. In certain applications, it may also be desirable to determine the least-cost path for each agent while either adhering to or explicitly avoiding specified homotopy classes [Bhattacharya et al., 2012]. In many such problems, the concept of visibility is inherently linked to homotopy structure. For example, in the tracking of uncertain agents within environments containing dynamic obstacles, the ability to reason about occlusion over specific time intervals becomes critical [Shapovalov and Pereira, 2020].

In certain applications, it may also be desirable to determine the least-cost path for each agent while either adhering to or explicitly avoiding specified homotopy classes. In many such problems, the concept of visibility is inherently linked to homotopy structure. For example, in the tracking of uncertain agents within environments containing dynamic obstacles, the ability to reason about occlusion over specific time intervals becomes critical [Yi et al., 2016]. In these scenarios, understanding the set of possible homotopy classes that an agent may follow during periods of occlusion can significantly enhance the efficiency of belief state propagation. By constraining the predicted trajectories to plausible homotopy classes, the system can reduce ambiguity, improve prediction accuracy, and enable more informed decision-making under uncertainty.

1.2.3 Exploring Spatial Representations for Efficient and Safe UAV Motion Planning

Another focus is the representation of the state trajectory and the safe flight corridor (SFC) for time-optimal motion planning. Conventional methods for corridor construction rely on convex approximations of the collision space. Seed points are selected along the initial path, and for each point, the maximum-volume shape, such as boxes [Lin et al., 2020], ellipsoids [Ren et al., 2022], or polytopes [Wu et al., 2025] that can be contained within the environment is computed. The final corridor is then constructed by overlap-

ping all the geometric objects. However, it is difficult to determine how the geometric shapes and volumes interplay with the dynamical feasibility of the trajectories computed by the optimization procedure [Wu et al., 2025]. Also, the resulting corridor is always irregular and not intuitive for end-users.

An emerging approach over the past decade is to reformulate the system dynamics using path-parameterized coordinates along the predefined reference trajectory [Verschuere et al., 2016, Spedicato and Notarstefano, 2017]. It projects system dynamics onto a moving frame along the path in free space, with the resulting states—path progress and orthogonal distance—combined with the path’s intrinsic properties (tangent, curvature, torsion) [Arrizabalaga and Ryll, 2022]. The spatial reformulation method was first introduced in autonomous driving, where vehicle dynamics were transformed into curvilinear coordinates along the road. This allowed nonlinear obstacle representations to be expressed as linear lateral constraints, enabling decoupled speed optimization along the tangential direction [Frasch et al., 2013]. Also, the notion of path progress and path properties is now embedded in the optimization problem. Verschuere *et al.* extended this approach to three-dimensional motion planning for robot manipulators [Verschuere et al., 2016], while Spedicato *et al.* applied a similar method to quadcopter systems, expressing the quadcopter pose in terms of decoupled tangential and transverse spatial states along the predefined path. This reformulation enabled the formulation of costs and constraints independent of the time variable [Spedicato and Notarstefano, 2017].

Among all these applications, the most common moving coordinate frame construction method along the given trajectory is the Frenet-Serret (FS) frame. It is well-defined for planar paths [Frasch et al., 2013, Kloeser et al., 2020, Li et al., 2024]. However, when it comes to 3D trajectories, the FS frame becomes undefined at inflection points, i.e., points where the curvature vanishes. Additionally, near those inflection points, the FS frame exhibits undesirable rotations (or distortions) about its tangential axis. Meanwhile, although the Parallel Transport (PT) frame presented in [Hanson and Ma, 1995] provides a smooth and singularity-free curve framing method, it contains undesired spinning and is highly dependent on the initial orientation

[Ramírez and Nahon, 2021]. Also, the PT frame can only be solved numerically, as it lacks a closed-form analytical solution. The model predictive contouring control (MPCC) method is closely related to spatial reformulation, which has shown its capability in trajectory tracking and autonomous drone racing [Romero et al., 2022b]. Yet, it requires high-accuracy arc-length parameterization for lag error and contour error construction, which also resorts to numerical methods only. Arrizabalaga *et al.* proposes the use of Pythagorean Hodograph (PH) splines with the advantage of analytical derivatives, but transforming from the original trajectory to PH spline is a computationally intense task in itself [Arrizabalaga and Ryll, 2022]. This motivates the search for a robust and efficient solution to the curve-framing method.

To address the limitations of existing curve-framing methods in achieving robust and efficient motion planning within constrained airspace, we adopted the Gravity Normal (GN) frame proposed by Ramírez and Nahon [2021]. Traditional frames such as the FS and PT frames often suffer from issues including instability, and lack of consistent orientation. In contrast, the GN frame provides a novel approach by aligning one of its transverse axes with the direction of gravity and the other transverse axis to a lateral direction, closely tied to inertial properties. This alignment not only improves frame stability but also offers a clear physical interpretation, making it highly effective for generating and navigating constrained flight corridors. As presented in the methodology section, the Gravity Normal (GN) frame facilitates the analytical computation of its axes, thereby enabling closed-form solutions for spatial reformulation. This capability directly addresses computational inefficiencies often associated with iterative methods required by other framing techniques. Furthermore, the GN frame inherently reduces sensitivity to variations in the curvature of the reference trajectory, making it particularly suitable for applications involving constrained airspace and tunnels, such as urban air mobility (UAM) corridors. ?

While the GN frame was originally developed for trajectory tracking in fixed-wing aircraft [Ramírez and Nahon, 2021], its potential for optimizing spatially constrained trajectories has remained underexplored. Recent adap-

tations, such as its use by Zhang et al. for controlling multi-axial machine tools Zhang et al. [2025], highlight its versatility across disciplines. In this work, we extend the GN frame to corridor-based trajectory optimization, leveraging its ability to represent and address complex spatial constraints in a computationally efficient manner. This extension enables the seamless integration of the GN frame into cooperative flight planning scenarios, where multiple UAVs operate within shared airspace. By utilizing the GN frame’s distinctive geometric properties, our approach bridges the gap between high-level corridor-based planning and low-level motion optimization, offering a scalable solution for real-world aerial robotics applications.

1.3 Objectives and Thesis Structure

1.3.1 Objectives

Motivated by the three aforementioned challenges, in this thesis the following objectives are set to be accomplished:

1. The mathematical models of both systems are developed. For the CSPQ system in particular, we investigate the integration of physics-based modeling with deep learning techniques. As described in section 1.2.1, conventional physics-based models fail to fully capture the complex, nonlinear behaviors of CSPQ systems, such as downwash on the payload and ground effects. Although this approach is not adopted in the subsequent framework, it lays the groundwork for enabling neural network-based models in motion planning.
2. Concerning challenge 2 described in section 1.2.2, we introduce an efficient global path planning method with a focus on homotopy paths, which are essential for subsequent trajectory optimization but have often been overlooked. Additionally, the effect of wind dynamics in urban environments on the global path is also considered.
3. Concerning challenge 3 described in section 1.2.3, based on the predefined global path, we develop a novel path-parametric method for three

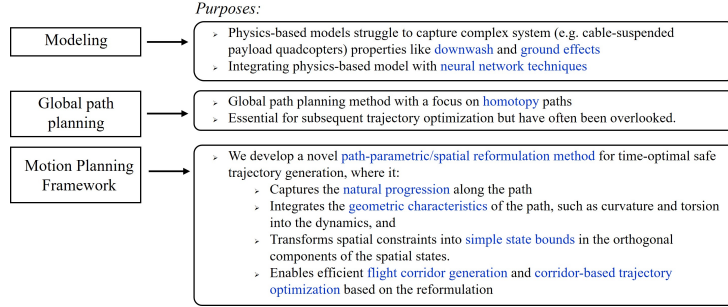


Figure 1.2: Summary of the objectives

key motivations: (1) it captures the natural progression along the path, (2) it integrates the geometric characteristics of the path, such as curvature and torsion into the dynamics, and (3) it transforms spatial constraints into convex formulations in the orthogonal components of the spatial states. We adopt a Gravity-Normal (GN) frame spatial reformulation method, which outperforms the conventional Frenet-Serret frame and Parallel-Transport frame in UAV scenarios. Subsequently, the GN frame allows us to define a flight corridor easily and intuitively, paving the way for optimal and safe trajectory optimization of the system.

A graphical summary is shown in 1.2. As a full solution, we propose the full framework including the offline planning module and the online trajectory tracking module, integrating the three objectives in detail.

1.3.2 Thesis Structure

After a brief introduction in Chapter 1, the rest of this thesis is organized as follows:

Chapter 2 provides a literature review on the current status of drone delivery, with a focus on motion planning. It includes four topics: Urban Air Mobility (UAM), current UAV modeling approaches, UAV motion planning approaches, and the emerging spatial reformulation techniques.

Chapter 3 presents the mathematical modeling of both the quadcopter and CSPQ systems. Regarding the complexity and nonlinearity of the CSPQ

systems, we explore and lay the groundwork for an alternative formulation using Physic-informed neural network approaches.

Chapter 4 addresses global path planning for UAV delivery systems, with a specific focus on path homotopy. In complex environments, multiple topologically distinct paths arise due to obstacles. Planning within a single class risks local optima, especially under nonlinear dynamics such as wind. Homotopy-aware planning improves performance by exploring topologically distinct paths.

Chapter 5 focuses on trajectory representation using spatial reformulation. Building on the global planner from Chapter 4, the trajectory is expressed in the spatial domain, where motion is parameterized by a progress variable. This approach simplifies time-optimal planning and enables a novel, intuitive, and simple flight corridor generation method, which is discussed afterwards.

Chapter 6 proposes a hierarchical framework for time-optimal motion planning and control, utilizing the techniques described in chapter 3 to 5. Both quadcopter and CSPQ models are used to demonstrate the approach.

Finally, the conclusion of the research, including a summary of contributions and suggestions for future research direction, is illustrated in Chapter 7.

Chapter 2

Literature Review

In this chapter, we will review the literature and related works in the following four topics: Overview on Urban Air Mobility (Section 2.1), Current UAV modelling approaches (Section 2.2), UAV motion planning approaches (Section 2.3) and the emerging spatial reformulation techniques (Section 2.4).

2.1 Urban Air Mobility

Urban air mobility (UAM) is a novel transportation system for flying vehicles carrying passengers or goods in point-to-point and short-distance mobility or delivery in urban or suburban contexts [Murça, 2021]. Compared to air traffic management, we need to place special care on UAM safety and ensure an efficient, affordable, safe air transport environment concerning wind dynamics and energy consumption [Roga et al., 2022]. According to the United Nations’s final report of World Urbanization Prospects: The 2018 Revision, over half of the world’s population resides in urban areas, and this proportion is expected to enlarge to nearly 70% by 2050 [Ritchie et al., 2024]. When developing a new transportation system, many concerns need to be considered, like social acceptance, air traffic management, regulation, and safety [Cohen et al., 2021]. For example, regulating aircraft maintenance, production, and operation and governing the licensing regime for pilots, aircrew, and aircraft are significant challenges for promoting UAM under government regulation [Cokorilo, 2020]. Managing flight risk is essential in safety issues, which includes ensuring human and property safety and confirming the stability of flight control systems and ground-based air traffic controllers [Li et al., 2021]. If all the barriers can be controlled, then the potential market of UAM in the business field is enormous.

One could expect that, with the increase of UAV adoption in urban air mobility and heightened urbanization progress, traffic congestion in airspace will subsequently increase and raise several safety concerns for human activ-

ities [Wang et al., 2022]. In 2016, Uber published the Elevate Whitepaper for describing the framework and requirements for providing air taxi service, including planned or non-scheduled on-demand flights. Benefiting from the rapid development of modern science and technology, the UAM helps transform mobility from 2D to 3D space movement [Fu et al., 2019]. The vehicle design and requirements, including the criteria of cruise speed, emission, and seat capacity, are essential for UAM implementation. Under UAM, there are different types of aerial vehicles, including Vertical Take-Off and Landing (VTOL), Short Take-Off and Landing (STOL), Small Uncrewed Aircraft (SUA), Uncrewed Aerial Vehicles (UAV), and Uncrewed Aircraft (UA). The passengers of air taxis and the users of logistics transportation services for delivering goods with Uncrewed control take advantage of UAM to shorten their traveling time or delivery time and save unexpected costs because of traffic congestion [Cohen et al., 2021].

2.2 UAV Modeling and System Identification

Notwithstanding the more complex case of cooperative load carrying for multiple UAVs, approaches for quadcopters carrying payload are mainly categorized into cable-suspended payload and load attached to the UAV's rigid body, where both systems are being used [Villa et al., 2020]. Directly attaching the payload to the UAV is mechanically simple, but the increased mass and inertia of the UAV reduce its maneuverability for dynamic response and disturbance rejection [Yu et al., 2024]. A cable-suspended payload with UAVs offers several operational benefits. It requires less payload space and weight for additional gripper structures, and there is only minimal added inertia on the UAV body. Meanwhile, a major drawback is that the payload adds two more passive degrees of freedom, generating swinging during flight and inducing instability.

The trajectory planner and controller of the Cable-Suspended Payload Quadcopter (CSPQ) should be capable of fast load transport with minimized swing [Rao et al., 2023]. One critical element for effective planning and control system design of the UAV is the accuracy of the dynamics model.

Although equations from physics-based principles can account for most of the system's behavior [Guerrero et al., 2015], the highly nonlinear effects generated by aerodynamic forces and torques, propeller interactions, vibrations, and other phenomena such as downwash on the payload have to be captured, and they are hard to model mathematically [Saviolo et al., 2022].

Conventional approaches collect experimental input-output data to produce a mathematical representation of the system's dynamics in the form of state-space equations or transfer functions, compensating for the model inaccuracy caused by nonlinearities, uncertainties, and disturbances. Common algorithms include Observer/Kalman Identification (OKID), which uses time domain representations; Comprehensive Identification from Frequency Responses (CIFER®) for frequency domain; Eigensystem Realization Algorithm (ERA); Observer/Controller Identification (OCID); and GLOMAP [Hoffer et al., 2014]. However, most of these algorithms limit themselves to multiple linear model representations for various operating conditions and are fundamentally insufficient to fully explain nonlinear behavior.

Considering the promising results from Deep Neural Networks (DNNs) as universal approximators with impressive learning capability, recent researchers have applied DNNs to model system dynamics [Harris et al., 2016, Bagherzadeh, 2018, Yu, 2017]. However, acquiring data is costly for certain complex physical or engineering systems. Under partial information, the outcome of the training process lacks robustness and generalization capability. In this regard, Physics-Informed Neural Networks (PINNs) aim to utilize prior knowledge from physical laws. It acts as a regularization agent that keeps the model prediction within physically possible bounds with lower training overhead, which demonstrated a substantially great performance on various systems, including quadcopters [Raissi et al., 2019, Gu et al., 2024].

2.3 UAV Motion Planning Methods

2.3.1 Global Planning

Literature addressing the UAV path planning problem is divided into 3 categories: search-based methods, sampling-based methods, and heuristics-based

methods. Search-based methods include Dijkstra’s algorithm, Bellman-Ford’s algorithm, the A* algorithm, and the like. The airspace is decomposed into graphs with nodes and edges, where nodes usually represent a discretized position and edges represent the cost, for example, the flight distance or energy cost, for a UAV trespassing between two connecting nodes. Zhang et al. adopted the JPS method [Zhang et al., 2021], an improved version of A*, to generate the reference path in urban airspace. However, the generated path is limited to the nodes and edges, which do not utilize the full airspace. Sampling-based methods include the Rapid-exploration Random Tree (RRT) algorithm and its improved version, i.e., RRT*, and Probability Road Map (PRM). For example, [Liu et al., 2024] introduced RRT path planning for last-mile super-chilling delivery. One limitation is that the inherent randomness induces a non-smooth trajectory. Heuristic-based algorithms employ customized heuristics to acquire near-optimal paths, leveraging their robust global optimization capabilities. The most influential methods include Ant Colony Optimization (ACO) , Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) [Yahia and Mohammed, 2023]. One drawback is that optimality is not guaranteed by the metaheuristic-based method. What is more, they shared a common drawback: these approaches are not guaranteed to generate a smooth and dynamic feasible trajectory.

Among them, the family of A* and Dijkstra algorithms are the most common way to solve path planning problems, with wide application especially in the popular navigation stack in robot operating system [Gatesichapakorn et al., 2019] . However, the distance measurement must be linear with a single objective only. Imposing additional operational constraints may require algorithm modification. Furthermore, it is challenging to model the UAV path planning in a free-space map. Based on the nature of path planning in urban air mobility, one could utilize the terrain constraints and formulate a safe network model using a roadmap-based planning algorithm such as Voronoi Diagram (VD) and Probabilistic Roadmap (PRM) [Huang et al., 2024]. The topological concept of retraction underpins a VD which refers to a skeleton-like structure formed by gradually reducing the open space [Fortune, 2017]. The Voronoi diagram can be perceived as a foundation, as it

has widespread usage for finding the best route in path planning and optimization. Dai and Cochran Jr. [Dai and Cochran Jr, 2010] used the Voronoi diagram and parameterized Cornu spirals to design a UAV route, beneficial for path planning and UAV control, which can improve the path in the initial stage by utilizing the Cornu spirals' continuous-curvature properties and attempting to match the Voronoi points with the shortest orthogonal distance. Zhang et al. [Zhang et al., 2018] presented the effect of different parameters on the overall performance of the algorithm proposed and the Voronoi graph by constructing a quantitative model, then discovered that the time required to establish the Voronoi graph and path smoothing is proportional to the number of threats. Pehlivanoglu [Pehlivanoglu, 2012] found that using a multi-frequency vibrational genetic algorithm (mVGA) with a Voronoi diagram in the initial population phase and additional genetic algorithms could be applied to solve the problem of path planning for the autonomous UAVs in a faster and more efficient way.

2.3.2 Local Planning

In the past 20 years, trajectory optimization of UAVs has been a popular research topic. Betts, I. Mir et al., and Rao summarized the related research on optimal trajectory planning [Betts, 1998, Mir et al., 2022, Rao, 2010]. Typically, trajectory optimization can be divided into hard-constrained methods and soft-constrained methods. Hard-constrained methods treat the trajectory optimization as a linear or nonlinear problem and solve them using optimization techniques [Hargraves and Paris, 1987]. A notable work is the minimum-snap trajectory generation [Mellinger and Kumar, 2011], in which trajectories are represented as piecewise polynomials and formulated as a quadratic programming (QP) problem. Soft-constrained methods formulate the problem as a nonlinear optimization problem, where smoothness and safety criteria can be incorporated as a penalty cost in the cost function Men and Requena Carrión [2020]. Ratliff et al. [Ratliff et al., 2009] rely on analytical functional gradient information, and Kalakrishnan et al. [Kalakrishnan et al., 2011] proposed a stochastic trajectory optimization framework

in which an estimated gradient is proposed. Zhou et al. [Zhou et al., 2019] utilized the convex hull property of B-splines to incorporate gradient information and dynamic constraints. However, since the optimization strategy of a single trajectory mentioned before is based on local optimization techniques, local minima often emerge due to the presence of obstacles. It leads to a set of topologically distinct candidate trajectories that should be figured out first, followed by optimizing each candidate to determine the true global optimum [McCammon et al., 2020, Rösmann et al., 2017a, Zhou et al., 2020].

In recent years, optimal control problem (OCP) formulations have been widely exploited concerning various costs such as energy consumption [Lim et al., 2019], time [Verschuere et al., 2016], noise [Udani et al., 2017], or multi-objective problems. OCP represents a natural framework and considers the complete dynamics of the system, and multiple classes of numerical optimization methods can be employed. OCP can also be implemented in a receding horizon control fashion, and it is more commonly known as Model Predictive Control (MPC). Nonlinear Model Predictive Control (NMPC) extends the capability of MPC, and it has gained popularity in the trajectory generation of UAVs [Arrizabalaga and Ryll, 2023]. NMPC provides a framework for finding the optimal control actions by considering a nonlinear cost function and incorporating constraints on both the states and inputs of the system. This approach is especially useful in dealing with the inherent nonlinearity in UAV dynamics, allowing for more accurate and effective trajectory generation. Also, NMPC can account for time-varying spatial bounds while enforcing constraints in the true physical limits, i.e., thrust commands. Thus, the near-time optimality of the solution is ensured.

There are two main classes of direct methods: direct shooting and direct transcription. For transcription methods, one key feature is the discretization of the trajectory. It approximates the trajectory as polynomial splines [Kelly, 2017], in which a spline is made up of a sequence of polynomial segments with a finite number of coefficients, and the polynomial derivatives match the values of the dynamics of the platform at a set of interior nodes within each subinterval. Thus, the problem can be transformed into a non-linear optimization or programming problem (NLP). In general,

it is relatively fast and applicable to a wide variety of problems [Bordalba et al., 2023], especially in the field of UAV trajectory, because its differential flatness property [Mellinger and Kumar, 2011] allows us to represent the trajectory in time-parameterized polynomials. The most popular transcription methods are direct collocation methods, which are classified into two families: local collocation and global collocation. In local collocation methods, the degree of the interpolating polynomial remains constant, and higher accuracy is achieved by increasing the number of subintervals or reducing the step size. Hermite-Simpson [Kim et al., 2020] and Gauss-Lobatto [Song et al., 2023] are two common collocation methods. One major advantage is its convergence speed. Since each constraint only relates to two adjacent state collocation nodes, only the entries near the diagonal of this matrix will be populated. The resulting sparsity in the Jacobian of the Lagrangian can speed up computation. In global collocation, also known as pseudospectral collocation, the representing polynomials, for example, Chebyshev-Gauss or Legendre-Gauss, have a higher degree, and higher accuracy is achieved by increasing the number of degrees of the polynomial. It is proven to converge faster for smooth problems [Benson et al., 2006]. Nevertheless, the solution may not accurately represent the exact physical behavior of the system, as dynamics are only enforced as constraints.

Another class of direct methods is the shooting method. Different from transcription methods that approximate trajectory as polynomials, the shooting method approximates through numerical simulation using system dynamics [Kelly, 2017]. Direct shooting is computationally hard, and a common extension is the multiple-shooting method [Kouzoupis et al., 2023]. Instead of representing the whole trajectory as a single simulation, the trajectory is divided into intervals, and each interval is represented by a simulation with additional constraints to avoid jumps in dynamics and constraints. Multiple shooting is more robust as it avoids integration drifts in long trajectories, and also simulation of the dynamics can be readily parallelized. It has been applied in a broad range of scenarios; for example, [Kouzoupis et al., 2023] adopted the multiple shooting framework to formulate the energy-efficient train control problem. [Diehl et al., 2005] used a multiple shooting scheme

for robot-arm control and solved it through a real-time iteration scheme (RTI).

2.4 Spatial Reformulation

Time-optimal trajectory optimization for a drone raises additional challenges, especially in an urban environment. A parcel delivery drone operates in a cluttered environment with buildings of various heights and sizes; thus, the feasible trajectories are often in nonregular shapes. Consequently, complex spatial constraints must be imposed on the UAV for obstacle avoidance. One popular approach is called spatial reformation proposed by [Verschueren et al., 2016], where the robot’s kinematics and dynamics are formulated as longitudinal coordinates along the path and transverse coordinates normal to the path. It introduces the progress variable s of the motion in the airspace. The time-optimal OCP can be expressed in a more natural form. By using s instead of time t , the infinite horizon trajectory optimization problem is approximated by a finite horizon problem where mission time is embedded within the system equations [Spedicato and Notarstefano, 2017].

The notion of spatial reformulation originates in the trajectory tracking problem with autonomous driving in 2D which uses the Frenet-Serret frame formula for reformulation [Werling et al., 2010] and [Kaminer et al., 2010] and recent applications extended it to drone trajectory [Arrizabalaga and Ryll, 2023]. Still, in this thesis, we show that when it comes to the 3D trajectory, the Frenet-Serret frame is not well-defined and a singularity-free reformulation is necessary. Second, the obstacles can be expressed in transverse dynamics only [Frasch et al., 2013] and spatial constraints are then enforced as simple spatial bounds [Spedicato and Notarstefano, 2017]. It is an analogy to the scenario in autonomous driving, where the vehicle is modeled in the space domain and in the 2D Frenet-Serret frame, allowing the formulation of driving corridor constraints (e.g., lane boundaries) as state bounds [Wang et al., 2023]. Regarding the solution to OCP and NMPC problems, existing methods resort to numerical methods, which are divided into 2 categories: indirect methods and direct methods [Benson et al., 2006]. Indirect meth-

ods rely on the calculus of variations and Pontryagin’s Maximum Principle. The optimization problem is formulated as a Hamiltonian two-point Boundary Value Problem (2PBVP), and the necessary conditions for optimality are derived and solved numerically. One major drawback is the optimality conditions must be derived for every different optimization problem, which requires expertise and heavy work. Any change in the system model means re-derivation is required.

2.5 Concluding Remarks

This chapter surveyed key areas relevant to UAV trajectory planning, UAM, UAV modeling, motion planning methods, and spatial reformulation techniques. UAM introduces new challenges in safety and regulation, requiring efficient and reliable UAV planning strategies.

The state-of-the-art approaches to UAV trajectory planning and control face several significant challenges, which this thesis aims to address. First, the rapid urbanization and increasing adoption of UAVs in Urban Air Mobility (UAM) introduce safety concerns, including airspace congestion and the need for efficient, reliable, and regulation-compliant planning frameworks. Existing motion planning methods often fail to ensure dynamic feasibility and smoothness, particularly in cluttered urban environments, as global planning methods are limited by coarse path representations, and local optimization methods are prone to getting stuck in local minima. Second, accurate system modeling is essential for effective trajectory planning, especially for complex configurations such as Cable-Suspended Payload Quadcopters (CSPQs). However, existing modeling approaches struggle to capture highly nonlinear effects, such as aerodynamic forces, propeller interactions, and payload-induced dynamics. While data-driven methods like Deep Neural Networks (DNNs) show promise, they often suffer from data inefficiency and lack physical consistency. Finally, spatial reformulation techniques, while promising, face challenges in handling 3D trajectories due to singularities in the Frenet-Serret frame and require further refinement for UAV-specific dynamics and constraints. This thesis addresses these gaps by developing novel modeling,

planning, and reformulation techniques to enhance the robustness, efficiency, and practicality of UAV trajectory planning in urban settings.

Chapter 3

UAV modeling

This chapter presents a comprehensive mathematical model for both quadcopter and CSPQ systems, which will serve as the foundation for subsequent discussions. The governing equations of motion for the quadcopter are first derived using the standard Newton-Euler approach, which is widely recognized for its robustness and reliability [Spedicato and Notarstefano, 2017]. For the more complex Cable-Suspended Payload Quadcopter (CSPQ) system, the equations of motion are also derived using the Newton-Euler formulation. Additionally, an alternative derivation based on the Euler-Lagrange formulation is provided.

The CSPQ system is inherently challenged by the increased modeling complexity arising from payload swing dynamics and intricate aerodynamic effects. As a preliminary effort to address these challenges, this chapter also explores the possibility of incorporating physics-based modeling with deep learning techniques. Specifically, this approach incorporates a Physics-Informed Neural Network (PINN) architecture, which integrates an attention-enhanced Temporal Convolutional Network (PIATCN). Comprehensive simulations demonstrate the effectiveness of this approach in predicting CSPQ dynamics across a range of flight scenarios. Still, the computational overhead and model complexity limit its applicability to real-time navigation and control algorithms.

Recurrent models, such as RNNs, LSTMs, and GRUs, often struggle with vanishing gradients and degraded memory retention over long horizons, particularly when faced with sharp urban flow transients. These limitations make them less effective for systems like the CSPQ, which exhibit sharp transitions and long-range dependencies. Temporal Convolutional Networks (TCNs) address these challenges through their convolutional receptive fields, dilation mechanisms, and inherent parallelism, enabling stable and efficient modeling of long-term temporal dependencies. This makes TCNs particularly well-suited for capturing the complex temporal dynamics of the CSPQ

system. Meanwhile, Physics-Informed Neural Networks (PINNs) embed governing physical equations directly into the loss function, providing a strong inductive bias that acts as a regularizer under conditions of sparse, uneven, or noisy data. By incorporating physical laws, PINNs not only improve data efficiency but also enhance generalization and ensure physical consistency in the learned model. This is especially valuable for capturing the intricate aerodynamic behaviors and payload-induced dynamics of the CSPQ system. Compared to purely data-driven models, which often suffer from data inefficiency and poor generalization, the physics-based approach of PINNs offers a significant advantage in maintaining accuracy across diverse scenarios, even those outside the training distribution. Together, these strengths make TCNs and PINNs an effective combination for modeling the CSPQ system.

3.1 Notation

Throughout this thesis, we adopt the following conventions for notation. A reference frame is denoted using curly brackets, such as $\{I\}$ or $\{A\}$. The leading superscript of a variable indicates the frame of reference, for example, $^I\mathbf{v}$ or ^Ax . For rotation matrices, the trailing subscript denotes the reference frame being described, such as $\mathbf{R}_A$.

Scalars are represented by lowercase lightface letters, such as x or y . Column vectors are represented by italic boldface lowercase letters, for example, $\mathbf{x}$. Unit vectors are denoted by upright bold lowercase letters, such as $\mathbf{e}$. Basis vectors of unit norm are indicated by a column vector with an overbar or sometimes upright bold letters for simplicity, for example, $\bar{\mathbf{x}}$ or $\mathbf{q}$. Matrices are denoted by boldface uppercase letters, such as $\mathbf{A}$ or $\mathbf{R}$. A block-diagonal matrix is represented as $\text{diag}(\cdot)$.

Geometric quantities, such as fixed points, lines, or planes, are represented by lightface lowercase sans serif typestyle letters, such as $\mathbf{p}$. Parametric curves are denoted by lightface uppercase letters, such as Γ . The directional derivative of the curve, although a vector quantity, is written in lightface style for consistency with the notation for the curve.

The inner product of two vectors is denoted as $\mathbf{a} \cdot \mathbf{b}$ or $\mathbf{a}^T \mathbf{b}$ interchangeably.

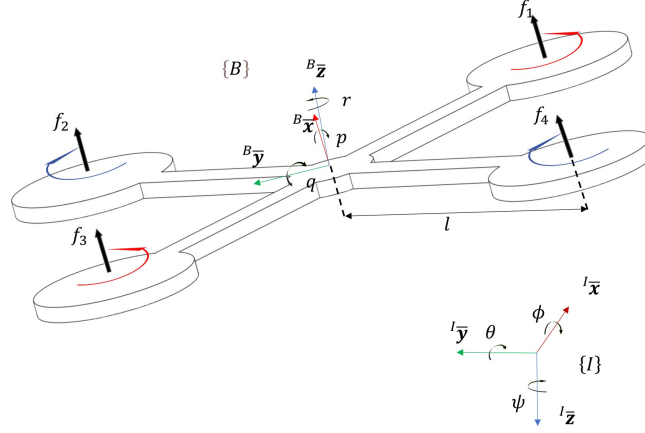


Figure 3.1: Configuration of the Quadcopter in the study along with the inertial and body frame

The hat operator, $\hat{\mathbf{a}}$, generates a skew-symmetric matrix corresponding to the cross product of two vectors, such that $\hat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$. The skew-symmetric matrix is defined as

$$\hat{\mathbf{a}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \quad (3.1)$$

3.2 Quadcopter Dynamics

3.2.1 Newton-Euler Approach

The quadcopter is modeled as a six-degree-of-freedom rigid body with four thrust inputs, as shown in Fig. 3.1. We adopted two reference frame systems to describe the motion of the quadcopter: the inertial frame $\{I\}$ is a fixed coordinate system in North-East-Down (NED) orientation where gravity is in the negative z -direction; and the body frame $\{B\}$ is attached to the center of mass of the quadcopter with its x -direction along the longitudinal direction and z -direction along the rotor thrust direction. The position ${}^I\mathbf{p} = [x \ y \ z]^T$ and velocity ${}^I\mathbf{v} = [\dot{x} \ \dot{y} \ \dot{z}]^T$ of the quadcopter are defined in $\{I\}$.

The body frame is attached to the center of mass of the quadcopter

specified by the "X" configuration, such that the center axis between motors 1 and 2 defines the x -direction of the body frame. The main advantage of the "X" configuration is that it opens the frontal space between motors 1 and 2 for the installation of onboard sensors, such as a solid-state LIDAR or a depth camera, etc. As shown in Fig. 3.1, the distance between the motors and the center of gravity measured on the body x - y plane is defined as l , with zero displacements in the body z -axis.

The quadcopter attitude is defined $\{I\}$ with three Euler angles, ϕ , θ , and ψ , respectively in terms of roll, pitch, and yaw angles, and can be combined into a single-column vector ${}^I\Phi = [\phi \ \theta \ \psi]^T$. Their time derivatives follow directly as ${}^I\dot{\Phi} = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$. Euler angles describe 3D orientations concisely and intuitively; however, at some angle configurations, two of the rotation axes align and cause a loss of one degree of freedom. This problem is commonly referred to as *gimbal lock* [Hanson, 2005]. For the yaw-pitch-roll sequence of Euler angles, gimbal lock arises when the pitch angle reaches $\pm\frac{\pi}{2}$. In this case, the roll and yaw rotation axes are aligned. Nevertheless, the gimbal lock problem can be avoided by setting explicit state bounds on the pitch angles.

In addition, the angular velocities ${}^B\omega = [\omega_p \ \omega_q \ \omega_r]^T$ are defined in $\{B\}$ alongside ${}^I\dot{\Phi}$, as gyroscopic data and thrust are always measured and act in the body frame of the robot. Note that ω and $\dot{\Phi}$ do not belong to the same reference frame, and the relationship between ω and $\dot{\Phi}$ can be described by the following equation:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix}}^{\mathbf{J}(\Phi)} \begin{bmatrix} \omega_p \\ \omega_q \\ \omega_r \end{bmatrix} \quad (3.2)$$

where $\mathbf{J}(\Phi)$ is the kinematic transformation matrix.

Four rotor thrust commands ${}^B\mathbf{f}_T = [f_1 \ f_2 \ f_3 \ f_4]^T$ are the inputs of the quadcopter. For simplicity, in this paper, it is assumed that the quadcopter motor dynamics can be neglected. The thrust and torque (${}^B F, {}^B \tau$)

acting on the body frame generated from the four rotor thrusts are governed by the following equations:

$${}^B\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 f_i \end{bmatrix}, \quad {}^B\boldsymbol{\tau} = \begin{bmatrix} \frac{l}{\sqrt{2}}(-f_1 + f_2 + f_3 - f_4) \\ \frac{l}{\sqrt{2}}(-f_1 - f_2 + f_3 + f_4) \\ c_\tau(f_1 - f_2 + f_3 - f_4) \end{bmatrix} \quad (3.3)$$

where l is the quadcopter's arm length, and c_τ is the rotor's torque constant of the quadcopter. It is assumed that all four rotors are identical.

Following the derivations in [Luukkonen, 2011] based on the Newton-Euler method, the quadcopter's equation of motion is given by

$$\dot{\mathbf{p}} = \mathbf{v} \quad (3.4a)$$

$$\dot{\mathbf{v}} = \mathbf{g} - \frac{1}{m}\mathbf{R}(\Phi)\mathbf{F} - \mathbf{R}(\Phi)\mathbf{D}\mathbf{R}^T(\Phi)\mathbf{v}_a \quad (3.4b)$$

$$\dot{\boldsymbol{\Phi}} = \mathbf{J}(\Phi)\boldsymbol{\omega} \quad (3.4c)$$

$$\dot{\boldsymbol{\omega}} = -\mathbf{I}^{-1}\hat{\boldsymbol{\omega}}\mathbf{I}\boldsymbol{\omega} + \mathbf{I}^{-1}\boldsymbol{\tau} \quad (3.4d)$$

where superscripts and subscripts in the above equations are omitted for simplicity, the dot operator $(\cdot)$ represents the derivative with respect to t , $\mathbf{R}(\Phi)$ is the rotation matrix formed by Euler angles in a yaw-pitch-roll sequence that maps any vector from the body frame to the inertial frame, which is written as

$${}^I\mathbf{R}_B(\Phi) = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \quad (3.5)$$

where for brevity, we now use $C_{(\cdot)}$ to denote $\cos(\cdot)$ and $S_{(\cdot)}$ to denote $\sin(\cdot)$. $\mathbf{g}$ is the gravity vector, m is the mass of the quadcopter, and $\mathbf{I}$ is the moment of inertia matrix.

Note that in the equation (3.4b), the term $\mathbf{R}(\Phi)\mathbf{D}\mathbf{R}^T(\Phi)\mathbf{v}_a$ represents the most dominant drag experienced by the quadcopter, and it is approxi-



For the quadcopter dynamics in first-order form, the full state of a quadcopter is defined as $\mathbf{x}_d = [\mathbf{p}^T \quad \mathbf{v}^T \quad \mathbf{\Phi}^T \quad \boldsymbol{\omega}^T]^T$, which includes both the position and attitude, as well as their derivatives. State inputs are defined as $\mathbf{u}_d = [f_1 \quad f_2 \quad f_3 \quad f_4]^T$, where the thrust ${}^B\mathbf{F}$ and torque ${}^F\boldsymbol{\tau}$ are obtained using equation (3.3). Therefore, the full state-space equation $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ in the inertial frame and body frame can be constructed using the state $\mathbf{x}_d$, input $\mathbf{u}_d$, and equation (3.4).

3.3 CSPQ Dynamics

3.3.1 Newton-Euler Approach

This section presents the mathematical modeling of the cable-suspended payload as a point mass spherical pendulum suspended from a single point and the computation of the forces exerted on the quadcopter due to the motion of the payload. Referring to Fig. 3.2, this time the inertial frame $\{I\}$ is in the East-North-Up (ENU) configuration. The body frame is attached to the center of mass, denoted as $\{B\}$, with its x -direction along the longitudinal direction and z -direction along the rotor thrust direction. The quadcopter body position is denoted as $\mathbf{p}_Q = {}^I[x_Q \ y_Q \ z_Q]^T \in \mathbb{R}^3$, and its velocity as $\mathbf{v}_Q = {}^I[v_x \ v_y \ v_z]^T \in \mathbb{R}^3$. The orientation of the quadcopter is described using Euler angles $\mathbf{\Phi} = {}^I[\phi \ \theta \ \psi]^T \in \mathbb{SO}(3)$.

The total thrust provided by the four rotors is denoted as $T_q \in \mathbb{R}$. Meanwhile, state-of-the-art controllers, e.g., PX4 [Meier et al., 2015], usually adopt a cascade control scheme with an off-board position/attitude control loop and an efficient onboard angular rate controller. Therefore, we define the body angular rate $\mathbf{\Omega} = {}^B[\Omega_p \ \Omega_q \ \Omega_r]^T \in \mathbb{R}^3$ as the system virtual control input.

Regarding the payload, its position is defined as $\mathbf{p}_L = {}^I[x_L \ y_L \ z_L]^T \in \mathbb{R}^3$. The cable is assumed to be taut, with its length denoted as l . The unit-length cable direction is given by $\mathbf{q} \in \mathbb{S}^2$. The payload position can then be expressed as

$$\mathbf{p}_L = \mathbf{p}_Q + l\mathbf{q}, \quad (3.6)$$

The angular velocity vector of the load is denoted as $\boldsymbol{\omega} \in \mathbb{R}^3$, and the time derivative of the unit vector $\mathbf{q}$ is related to $\boldsymbol{\omega}$ through the skew-symmetric matrix $\hat{\boldsymbol{\omega}} \in \mathbb{R}^{3 \times 3}$, such that $\dot{\mathbf{q}} = \hat{\boldsymbol{\omega}}\mathbf{q}$. Using the triple product identity, the derivative of the angular velocity of the load is

$$\begin{aligned} \ddot{\mathbf{q}} &= \dot{\hat{\boldsymbol{\omega}}}\mathbf{q} + \hat{\boldsymbol{\omega}}(\hat{\boldsymbol{\omega}}\mathbf{q}) \\ &= \hat{\dot{\boldsymbol{\omega}}}\mathbf{q} - \|\boldsymbol{\omega}\|^2\mathbf{q} \end{aligned} \quad (3.7)$$

since $\mathbf{q}^T \boldsymbol{\omega} = \mathbf{q}^T \dot{\boldsymbol{\omega}} = 0$. The equations of motion for the quadcopter and the load can be written as

$$m_Q \dot{\mathbf{v}}_Q = {}^I \mathbf{R}_B \mathbf{T}_Q - m_Q \mathbf{g} - T_L \mathbf{q} \quad (3.8a)$$

$$m_L \dot{\mathbf{v}}_L = T_L \mathbf{q} - m_L \mathbf{g} \quad (3.8b)$$

, where ${}^I \mathbf{R}_B$ is the rotation matrix from body frame to inertial frame, T_L is the scalar tension in the cable, $\mathbf{g} = {}^I \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ is the gravity vector, and $\mathbf{T}_Q = {}^B \begin{bmatrix} 0 & 0 & T_Q \end{bmatrix}^T$ is the body thrust vector. Differentiating equation (3.6) twice with respect to t , substituting equation (3.7), (3.8a) and (3.8b) into it, and taking the cross product of $\mathbf{q}$ on both sides, we obtain

$$\dot{\boldsymbol{\omega}} = -\frac{\hat{\mathbf{q}} ({}^I \mathbf{R}_B \mathbf{T}_Q)}{m_Q l} \quad (3.9)$$

Similarly, differentiating equation (3.6) twice with respect to t and combining equation (3.8a) and (3.8b), we get

$$\begin{aligned} (m_Q + m_L)(\dot{\mathbf{v}}_Q + \mathbf{g}) &= \frac{m_Q + m_L}{m_Q} {}^I \mathbf{R}_B \mathbf{T}_Q \\ &\quad - \frac{m_L}{m_Q} \mathbf{q}(\mathbf{q} \cdot {}^I \mathbf{R}_B \mathbf{T}_Q) \\ &\quad + m_L l \|\boldsymbol{\omega}\|^2 \mathbf{q} \end{aligned} \quad (3.10)$$

We define the complete state vector and the input vector to the system as

$$\mathbf{x} = [\mathbf{p}_Q^T \quad \mathbf{v}_Q^T \quad \mathbf{q}^T \quad \boldsymbol{\omega}^T \quad \boldsymbol{\Phi}^T]^T \quad (3.11)$$

$$\mathbf{u} = \begin{bmatrix} T_Q & \boldsymbol{\Omega}^T \end{bmatrix}^T \quad (3.12)$$

By denoting $\mathbf{F} = {}^I \boldsymbol{\Gamma}_B \mathbf{T}_Q$ and rearranging equation (3.10), we obtain the complete kinematic and dynamic equations of motion for the CSPQ system

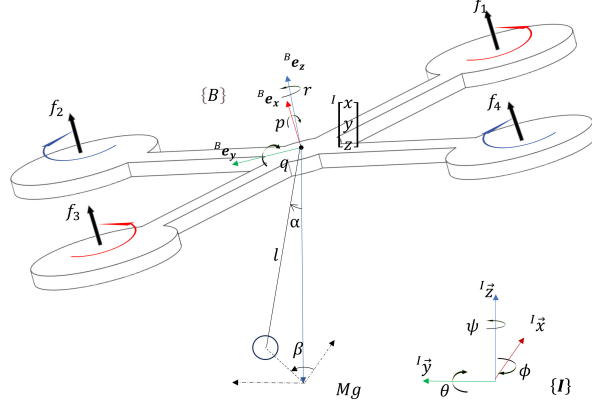


Figure 3.3: Configuration of the CSPQ system in the study, along with the inertial and body frame.

as

$$\dot{\mathbf{p}}_Q = \mathbf{v}_Q \quad (3.13a)$$

$$\begin{aligned} \dot{\mathbf{v}}_Q = & \frac{\mathbf{F}}{m_Q} - \frac{m_L}{m_Q(m_Q + m_L)} \mathbf{q}(\mathbf{q} \cdot \mathbf{F}) \\ & + \frac{m_L l \|\boldsymbol{\omega}\|^2}{m_Q + m_L} \mathbf{q} - \mathbf{g} \end{aligned} \quad (3.13b)$$

$$\dot{\mathbf{q}} = -\hat{\mathbf{q}}\boldsymbol{\omega} \quad (3.13c)$$

$$\dot{\boldsymbol{\omega}} = -\frac{\hat{\mathbf{q}}\mathbf{F}}{m_Q l} \quad (3.13d)$$

$$\dot{\Phi} = \mathbf{J}\boldsymbol{\Omega} \quad (3.13e)$$

3.3.2 Euler-Lagrange Formulation

In the previous section, the system input was chosen as the target angular rate. Alternatively, this section adopts the standard Euler-Lagrange approach to derive the nominal kinematics and dynamics model of the CSPQ [Zhang et al., 2023c], which facilitates setting the system input as the motor torque.

This time, the inertial frame $\{I\}$ is in ENU (East-North-Up) orientation, where gravity aligns with the negative z -direction. The body frame $\{B\}$ adopts the FLU (Forward-Left-Up) configuration, whereas the z -axis of $\{B\}$ is aligned with the quadcopter's thrust vector while pointing upwards when the vehicle is in hover. The position ${}^I\boldsymbol{\xi}_d = [x_d \ y_d \ z_d]^T$ and velocity ${}^I\mathbf{v}_d = [\dot{x}_d \ \dot{y}_d \ \dot{z}_d]^T$ of the quadcopter are expressed in $\{I\}$.

The quadcopter attitude is expressed in $\{I\}$ with the three Euler angles ϕ , θ and ψ , which are combined into a single-column vector ${}^I\boldsymbol{\Phi}_d = [\phi \ \theta \ \psi]^T$ with the time derivatives ${}^I\dot{\boldsymbol{\Phi}}_d = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$. The body angular velocities ${}^B\boldsymbol{\omega}_d = [\omega_p \ \omega_q \ \omega_r]^T$ are expressed in $\{B\}$. These angular velocities represent gyroscopic data, which, along with thrust, are always measured and act on the body frame of the robot. Throughout this paper, all vectors are denoted as bold lower-case characters, whereas matrices are denoted as bold upper-case characters. The relationship between $\boldsymbol{\omega}_d$ and $\dot{\boldsymbol{\Phi}}_d$ is governed by $\mathbf{J}(\boldsymbol{\Phi})$ in equation (3.2).

The payload is treated as a point mass of mass m tethered by a cable of length l . For simplicity, it is assumed that the cable is always rigid and taut. An additional payload frame $\{P\}$ is required to describe its position and velocity. Its origin is also attached to the center of mass of the quadcopter body, while the orientation is parallel to $\{I\}$. Thus, the position of the payload with respect to $\{I\}$ is defined by two swing angles ${}^P\boldsymbol{\mu} = [\alpha \ \beta]^T \in \mathbb{S}^2$ as shown in Fig. 3.3. α and β represent the rotation around the y -axis and the x -axis of $\{P\}$ respectively. At the initial moment, when the payload naturally droops, its position in the frame $\{P\}$ is given as $[0 \ 0 \ -l]^T$. Hence, the actual position of the payload in $\{I\}$ is given by the vector $\boldsymbol{\xi}_p = [x_p \ y_p \ z_p]^T$ as

$$\boldsymbol{\xi}_p = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} - l \begin{bmatrix} S_\alpha C_\beta \\ -S_\beta \\ C_\alpha C_\beta \end{bmatrix} \quad (3.14)$$

Therefore, the generalized state vector $\mathbf{q}$ for the whole system is defined as

$$\mathbf{q} = [\boldsymbol{\xi}_d^T \quad \boldsymbol{\Phi}_d^T \quad \boldsymbol{\mu}^T]^T \in \mathbb{R}^8 \quad (3.15)$$

The four motor thrust commands ${}^B\mathbf{u}_M = [u_1 \quad u_2 \quad u_3 \quad u_4]^T$ are the inputs of the quadcopter. The thrust and torque $({}^B\mathbf{F}, {}^B\boldsymbol{\tau})$ acting on the body frame $\{B\}$ are generated from the four-rotor thrusts and governed by the following equations:

$${}^B\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 u_i \end{bmatrix}, \quad {}^B\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_T \frac{l_d}{\sqrt{2}} (u_1 - u_2 - u_3 + u_4) \\ c_T \frac{l_d}{\sqrt{2}} (u_1 + u_2 - u_3 - u_4) \\ c_\tau (u_1 - u_2 + u_3 - u_4) \end{bmatrix} \quad (3.16)$$

where l_d is the quadcopter's arm length, c_T is the aerodynamic coefficient of the motor and c_τ is the rotor's torque constant. It is assumed that all four rotors are identical and motor transient dynamics are negligible.

The Lagrangian $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ is formulated as the total kinetic energy $\mathcal{K}$ (sum of quadcopter and payload kinetic energies $\mathcal{K}_d + \mathcal{K}_p$) minus the potential energy $\mathcal{V}$. The quadcopter body's total generalized kinetic energy $\mathcal{K}_d$ is expressed as

$$\begin{aligned} \mathcal{K}_d &= \frac{1}{2} M \dot{\boldsymbol{\xi}}_d^T \dot{\boldsymbol{\xi}}_d + \frac{1}{2} \boldsymbol{\omega}_d^T \mathbf{I}_d \boldsymbol{\omega}_d \\ &= \frac{1}{2} M \dot{\boldsymbol{\xi}}_d^T \dot{\boldsymbol{\xi}}_d + \frac{1}{2} \dot{\boldsymbol{\Phi}}_d^T \mathbf{J} \dot{\boldsymbol{\Phi}}_d \end{aligned} \quad (3.17)$$

where $\mathbf{I}_d$ is the quadcopter inertia matrix $\text{diag}(I_{xx}, I_{yy}, I_{zz})$. I_{xx} , I_{yy} , and I_{zz} are the moments of inertia about its three axis in $\{B\}$. To express the second term in terms of $\dot{\boldsymbol{\Phi}}_d$, the jacobian matrix $\mathbf{J}$ is denoted as

$$\begin{aligned} \mathbf{J} &= \mathbf{W}^T \mathbf{I}_d \mathbf{W} \\ &= \begin{bmatrix} I_{xx} & 0 & -I_{xx} S_\theta \\ 0 & I_{yy} C_\phi^2 + I_{zz} S_\phi^2 & (I_{yy} - I_{zz}) C_\phi S_\phi C_\theta \\ -I_{xx} S_\theta & (I_{yy} - I_{zz}) C_\phi S_\phi C_\theta & I_{xx} S_\theta^2 + I_{yy} S_\phi^2 C_\theta^2 + I_{zz} C_\phi^2 C_\theta^2 \end{bmatrix} \end{aligned} \quad (3.18)$$

Similarly, denote $I_p \in \mathbb{R}$ as the moment of inertia of the payload, which assumes uniformly distributed mass moments, the total generalized kinetic

energy of the payload $\mathcal{K}_p$ is denoted by

$$\mathcal{K}_p = \frac{1}{2} m \dot{\boldsymbol{\xi}}_p^T \dot{\boldsymbol{\xi}}_p + \frac{1}{2} I_p \dot{\boldsymbol{\mu}}^T \dot{\boldsymbol{\mu}} \quad (3.19)$$

The total potential energy $\mathcal{V}$ of the system results from the sum of the potential energy of the quadrotor and the payload as

$$\mathcal{V} = Mgz_d + mg(z_d - lC_\alpha) \quad (3.20)$$

where g is the gravity scalar. By equation (3.17), (3.19) and (3.20), the explicit form of $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ is written as

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}(M+m)(\dot{x}_d^2 + \dot{y}_d^2 + \dot{z}_d^2) + I_p(\dot{\alpha}^2 + \dot{\beta}^2) \\ & + \frac{1}{2}I_{xx}\dot{\phi}^2 + \frac{1}{2}(I_{yy}C_\phi^2 + I_{zz}S_\phi^2)\dot{\theta}^2 \\ & + \frac{1}{2}(I_{xx}S_\theta^2 + I_{yy}S_\phi^2C_\theta^2 + I_{zz}C_\phi^2C_\theta^2)\dot{\psi}^2 \\ & - I_{xx}S_\theta\dot{\psi}\dot{\phi} + (I_{yy} - I_{zz})C_\phi C_\theta S_\phi\dot{\psi}\dot{\theta} \\ & + \frac{1}{2}ml^2C_\beta^2\dot{\alpha}^2 + \frac{1}{2}ml^2\dot{\beta}^2 + mlC_\beta\dot{y}_d\dot{\beta} \\ & + ml\dot{\beta}S_\beta(S_\alpha\dot{x}_d + C_\alpha\dot{z}_d) - ml\dot{\alpha}C_\beta(C_\alpha\dot{x}_d - S_\alpha\dot{z}_d) \\ & - (M+m)gz_d + mglC_\alpha C_\beta \end{aligned} \quad (3.21)$$

With $\mathcal{L}$ defined, the full CSPQ model is derived from the Euler-Lagrange equations under external generalized forces:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \begin{bmatrix} {}^I\mathbf{f} \\ {}^B\boldsymbol{\tau} \\ \mathbf{0}^{2 \times 1} \end{bmatrix} \in \mathbb{R}^8 \quad (3.22)$$

where ${}^I\mathbf{f} = {}^I\mathbf{R}_B {}^B\mathbf{F}$ is the force applied to the quadrotor in $\{I\}$. ${}^I\mathbf{R}_B$ is the rotation matrix formed by the Euler angles that maps any vector from $\{B\}$ to $\{I\}$ and it is omitted for brevity. Details for Euler angle representation and Euler-Lagrange formulation are available in Craig et al. [Craig, 2006]. For $\boldsymbol{\mu}$, it is assumed that there is no external force acting on the payload,

and thus the force is a zero vector. Solving equation (3.22), it is able to formulate the CSPQ to the standard form of the robot dynamic equation as

$$\mathcal{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathcal{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathcal{G}(\mathbf{q}) = \mathcal{B}(\mathbf{q})\mathbf{u} \quad (3.23)$$

where $\mathcal{M}(\mathbf{q}) \in \mathbb{R}^{8 \times 8}$ is the inertia matrix. $\mathcal{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{8 \times 8}$ captures Coriolis and centrifugal forces. $\mathcal{G}(\mathbf{q}) \in \mathbb{R}^8$ is the gravity vector. The generalized control inputs are chosen to be $\mathbf{u} = [F \quad \tau_\phi \quad \tau_\theta \quad \tau_\psi]^T \in \mathbb{R}^4$ using equation (3.16). $\mathcal{B}(\mathbf{q}) \in \mathbb{R}^{8 \times 4}$ maps $\mathbf{u}$ into generalized forces, and it is not invertible because the system is underactuated. Specifically, $\mathcal{M}$ is given by

$$\begin{aligned} \mathcal{M}(\mathbf{q}) &= \begin{bmatrix} \mathbf{m}_{11} & \mathbf{0}_{3 \times 3} & \mathbf{m}_{13} \\ \mathbf{0}_{3 \times 3} & \mathbf{J} & \mathbf{0}_{3 \times 2} \\ \mathbf{m}_{31} & \mathbf{0}_{2 \times 3} & \mathbf{m}_{33} \end{bmatrix}, \\ \mathbf{m}_{11} &= (M + m)\mathbf{I}_3, \\ \mathbf{m}_{13} &= ml \begin{bmatrix} -C_\alpha C_\beta & S_\alpha S_\beta \\ 0 & C_\beta \\ S_\alpha C_\beta & C_\alpha S_\beta \end{bmatrix}, \\ \mathbf{m}_{31} &= \mathbf{m}_{13}^T, \\ \mathbf{m}_{33} &= \begin{bmatrix} ml^2 C_\beta^2 + I_p & 0 \\ 0 & ml^2 + I_p \end{bmatrix} \end{aligned} \quad (3.24)$$

where $\mathbf{I}_m$ denotes a $m \times m$ unitary matrix. $\mathcal{C}$ is given by

$$\begin{aligned}
\mathcal{C}(\mathbf{q}, \dot{\mathbf{q}}) &= \begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{c}_{13} \\ \mathbf{0}_{3 \times 3} & \mathbf{c}_{22} & \mathbf{0}_{3 \times 2} \\ \mathbf{0}_{2 \times 3} & \mathbf{0}_{2 \times 3} & \mathbf{c}_{33} \end{bmatrix}, \\
\mathbf{c}_{13} &= ml \begin{bmatrix} C_\alpha S_\beta \dot{\beta} + S_\alpha C_\beta \dot{\alpha} & S_\alpha C_\beta \dot{\beta} + C_\alpha S_\beta \dot{\alpha} \\ 0 & S_\beta \dot{\beta} \\ -(S_\alpha S_\beta \dot{\beta} - C_\alpha C_\beta \dot{\alpha}) & -(S_\alpha S_\beta \dot{\alpha} - C_\alpha C_\beta \dot{\beta}) \end{bmatrix}, \quad (3.25) \\
\mathbf{c}_{22} &= \mathbf{C}_{3 \times 3}, \\
\mathbf{c}_{33} &= ml^2 \begin{bmatrix} 0 & -2S_\beta C_\beta \dot{\alpha} \\ S_\beta C_\beta \dot{\alpha} & 0 \end{bmatrix}
\end{aligned}$$

where the elements in the matrix $\mathbf{C}_{3 \times 3}$ are

$$\begin{aligned}
C_{11} &= 0 \\
C_{12} &= (I_{yy} - I_{zz}) \left(\dot{\theta} C_\phi S_\phi + \dot{\psi} S_\phi^2 C_\theta \right) + (I_{zz} - I_{yy}) \dot{\psi} C_\phi^2 C_\theta - I_{xx} \dot{\psi} C_\theta \\
C_{13} &= (I_{zz} - I_{yy}) \dot{\psi} C_\phi S_\phi C_\theta^2 \\
C_{21} &= (I_{zz} - I_{yy}) \left(\dot{\theta} C_\phi S_\phi + \dot{\psi} S_\phi^2 C_\theta \right) + (I_{yy} - I_{zz}) \dot{\psi} C_\phi^2 C_\theta + I_{xx} \dot{\psi} C_\theta \\
C_{22} &= (I_{zz} - I_{yy}) \dot{\phi} C_\phi S_\phi \\
C_{23} &= -I_{xx} \dot{\psi} S_\theta C_\theta + I_{yy} \dot{\psi} S_\phi^2 S_\theta C_\theta + I_{zz} \dot{\psi} C_\phi^2 S_\theta C_\theta \\
C_{31} &= (I_{yy} - I_{zz}) \dot{\psi} C_\theta^2 S_\phi C_\phi - I_{xx} \dot{\theta} C_\theta \\
C_{32} &= (I_{zz} - I_{yy}) \left(\dot{\theta} C_\phi S_\phi S_\theta + \dot{\phi} S_\phi^2 C_\theta \right) \\
&\quad + (I_{yy} - I_{zz}) \dot{\phi} C_\phi^2 C_\theta + I_{xx} \dot{\psi} S_\theta C_\theta - I_{yy} \dot{\psi} S_\phi^2 S_\theta C_\theta - I_{zz} \dot{\psi} C_\phi^2 S_\theta C_\theta \\
C_{33} &= (I_{yy} - I_{zz}) \dot{\phi} C_\phi S_\phi C_\theta^2 - I_{yy} \dot{\theta} S_\phi^2 C_\theta S_\theta - I_{zz} \dot{\theta} C_\phi^2 C_\theta S_\theta + I_{xx} \dot{\theta} C_\theta S_\theta
\end{aligned}$$

$\mathcal{G}$ is denoted as $\mathcal{G}(\mathbf{q}) = [\mathbf{0}_{1 \times 2} \quad (M + m)g \quad \mathbf{0}_{1 \times 3} \quad mglS_\alpha C_\beta \quad mglC_\alpha S_\beta]^T$,
and $\mathcal{B}$ is denoted as

$$\mathcal{B}(\mathbf{q}) = \begin{bmatrix} \mathbf{b}_{11} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 1} & \mathbf{I}_3 \\ \mathbf{0}_{2 \times 1} & \mathbf{0}_{2 \times 3} \end{bmatrix}, \quad \mathbf{b}_{11} = \begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi S_\theta C_\phi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix} \quad (3.26)$$

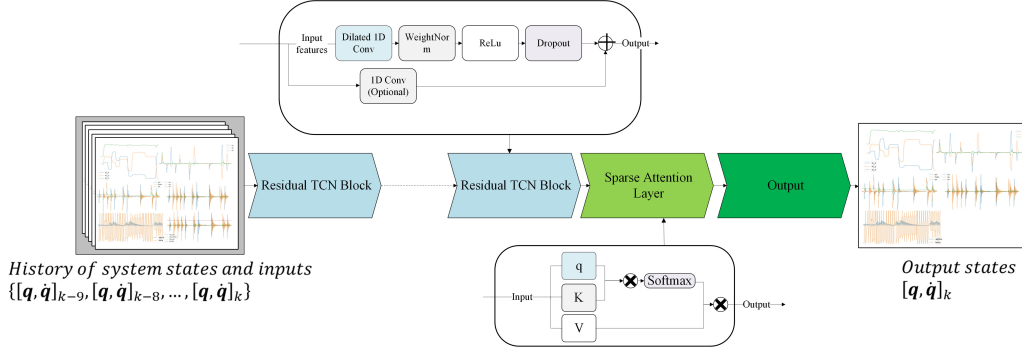


Figure 3.4: The overall methodological framework of the PIATCN model with attention mechanism for the state prediction function, $\hat{\mathcal{F}}_{\text{PIATCN}}$

By extending the state vector to incorporate its derivative, i.e. $\mathbf{X} = [\mathbf{q}^T \quad \dot{\mathbf{q}}^T]^T$, using equation (3.23), we can obtain the full state space form as

$$\dot{\mathbf{X}} = \mathcal{F}(\mathbf{X}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathcal{M}(\mathbf{q})^{-1} (\mathcal{B}\mathbf{u} - \mathcal{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{G}(\mathbf{q})) \end{bmatrix} \quad (3.27)$$

Using equation (3.27), we obtain the state transition equation

$$\mathbf{X}_{k+1} = \hat{\mathcal{F}}_{EK4}(\mathbf{X}_k, \mathbf{u}_k, \Delta t) \quad (3.28)$$

, where $\hat{\mathcal{F}}_{EK4}$ represents the 4th order Runge-Kutta numerical integrator for the state space equation and Δt is the integration time step. $\hat{\mathcal{F}}$ captures most of the nonlinearity in CSPQ, but it is insufficient for the following issues. The first issue is to accurately identify the parameters, such as moment of inertia $\mathbf{I}_d$, I_p , M , c_τ and l , not to mention m always varying depending on the mass of the actual payload. Another issue is the external effects, such as wind disturbances, discrepancies in mechanical components, motor dynamics, and software control delays.

3.3.3 CSPQ Dynamics using PIATCN

3.3.3.1 Model Learning

Physics-based methods capture the majority of system dynamics; however, nonlinear effects, such as ground effect and catenary dynamics, play a signif-

ificant role and can induce instability. In this section, we approximate $\hat{\mathcal{F}}_{EK4}$ using a physics-inspired temporal convolutional network with an attention mechanism. It leverages past flight states and control inputs to predict the CSPQ's full dynamic state for future time steps. Concretely, we opt for the network $\hat{\mathcal{F}}_{PIATCN}$ such that

$$\{\mathbf{X}_{k+1}\}_N = \hat{\mathcal{F}}_{PIATCN}(\{\mathbf{X}_k, \mathbf{u}_k\}_M, \{\mathbf{u}_{k+1}\}_N; \Theta) \quad (3.29)$$

Here, $\{\mathbf{X}_k, \mathbf{u}_k\}_M = [\mathbf{X}_{k-M}^T, \mathbf{u}_{k-M}^T, \dots, \mathbf{X}_k^T, \mathbf{u}_k^T]^T$ represent the sets of past states and control inputs of length M . $\{\mathbf{u}_{k+1}\}_N = [\mathbf{u}_k^T, \dots, \mathbf{u}_{k+N-1}^T]^T$ is the set of the proposed input to the system with $N - 1$ steps. In many modern control approaches, such as model predictive control, it is common for the control algorithm to output a series of control commands in the prediction horizon. Different from equation (3.28), given the set of upcoming control inputs, we attempt to extend the state transition equation to predict the CSPQ state for the next N steps, i.e. $\{\mathbf{X}_{k+1}\}_N = [\mathbf{X}_{k+1}^T, \dots, \mathbf{X}_{k+N}^T]^T$. Θ corresponds to the network model parameters for system identification, which need to be optimized.

3.3.3.2 Network Structure

As illustrated in Fig. 3.4, the proposed PIATCN architecture consists of three main components: (1) temporal convolutional network (TCN) layers, (2) a Bahdanau attention layer [Bahdanau et al., 2014], and (3) an output layer. The TCN layers extract time-correlated features from the sequence of past flight states and control inputs, generating a compact hidden-state vector. This hidden state is then passed to both the attention layer and the output layer, which together predict the quadrotor's full system dynamics.

In recent years, Long Short-Term Memory (LSTM) networks have been widely regarded as the standard neural network architecture for time series analysis within the deep learning community. More recently, Temporal Convolutional Networks (TCNs) have emerged as a competitive alternative for time series processing [Bai et al., 2018, Saviolo et al., 2022, Lara-Benítez et al., 2021]. TCNs offer several advantages, including flexible receptive field

sizes, lower memory requirements, and the ability to leverage parallel computation. Furthermore, TCNs inherently mitigate issues such as vanishing or exploding gradients, which are commonly encountered in recurrent architectures like RNNs and LSTMs.

A TCN residual block consists of several convolutional hidden layers with the same width as the input, which allows it to process long history sequences in parallel while encoding the temporal structure of the input time series in their output feature vector. It leverages three techniques: causal convolutions, which ensure outputs at time t depend only on current and past inputs; dilated convolutions, which expand the receptive field to capture long-range dependencies without increasing the number of parameters; and residual connections, which alleviate gradient vanishing by introducing shortcut paths [Van Den Oord et al., 2016, Bai et al., 2018].

The hidden-state vector extracted by the TCN is provided to the Bahdanau attention mechanism [Bahdanau et al., 2014]. We utilize its encoder-decoder architecture to exploit the most informative segments of the input sequence, irrespective of its length, thereby overcoming the traditional limitation of recurrent network models that rely on encoding the entire input. This architectural design leads to improved performance, particularly when dealing with long sequential data. The output layer is a multilayer perceptron (MLP), which is a dense network and better suited for making predictions from a compact hidden state representation.

3.3.3.3 Physics-inspired Loss Function

When it comes to model training, learning system dynamics solely from data bears the risk that the resulting network generalizes beyond the training distribution when the data are insufficient or unrepresentative. On the other hand, it is essential that the model remain consistent with underlying physical principles. Motivated by the previous works on quadcopters [Saviolo et al., 2022, Gu et al., 2024] that demonstrate the benefits of incorporating physical constraints into learning-based models, we incorporate physics-based constraints directly into the training process by integrating the state

transition in equation (3.28) into the loss function.

Specifically, at each training iteration, we minimize a composite loss function that enforces both data fidelity and adherence to physical principles

$$L = \lambda_1 L_{Fit} + \lambda_2 L_{Phy} \quad (3.30)$$

, where L_{Fit} corresponds to the mean squared error prediction loss between the training labels and the network predictions. L_{Phy} is the mean squared error physics-inspired loss between the physics laws' solution and the network's predictions. λ_1 and λ_2 are the hyperparameters that determine the relative weighting of each term. L_{Fit} and L_{Phy} are given by

$$L_{Fit} = \sum_{i=k}^{k+N} \left\| \bar{\mathcal{F}}(\mathbf{X}_i, \mathbf{u}_i) - \hat{\mathcal{F}}_{PIATCN}(\{\mathbf{X}_k, \mathbf{u}_k\}_M, \{\mathbf{u}_{k+1}\}_N; \Theta) \right\| \quad (3.31)$$

$$L_{Phy} = \sum_{i=k}^{k+N} \left\| \hat{\mathcal{F}}_{EK4}(\mathbf{x}_k, \mathbf{u}_k) - \hat{\mathcal{F}}_{PIATCN}(\{\mathbf{X}_k, \mathbf{u}_k\}_M, \{\mathbf{u}_{k+1}\}_N; \Theta) \right\| \quad (3.32)$$

where $\bar{\mathcal{F}}$ is the ground truth of the CSPQ system state given the past state and the proposed input. While L_{Fit} ensuring that the network learns the system dynamics directly from the ground truth, L_{Phy} constrains the predictions to be consistent with underlying equations derived from physics-based principles. Incorporating L_{Phy} imposes a physical interpretation on the network's internal states and serves as an unsupervised regularizer, hence enhancing generalization and stabilizing the training process.

3.3.3.4 Results

In this section, we describe the data acquisition from simulation experiments, model training procedures, and the prediction performance of the PIATCN model.

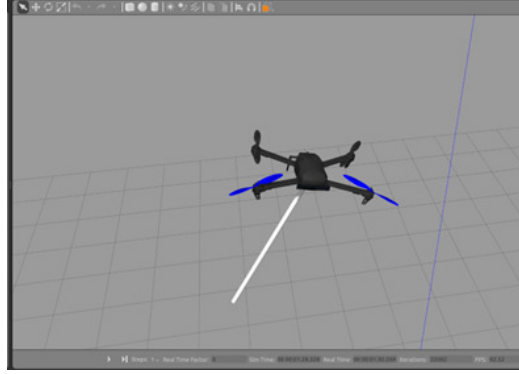


Figure 3.5: The modified Iris quadcopter model in Gazebo.

3.3.3.5 Data Acquisition

To acquire flight data, we conducted simulation experiments on a desktop computer equipped with an AMD Ryzen 7 5800X 8-Core Processor and 32 GB of RAM, running Ubuntu 20 OS. We adopted ROS SITL [Team, 2024] as the simulation platform. Specifically, we modified the built-in IRIS quadcopter model to include the tether and payload with a modification that it carries a payload, which is shown in Fig. 3.5. The payload is modeled as a point mass at the tip and the cable is modeled as a rigid rod with a ball joint connected to the quadcopter. The quadcopter itself weighs 1.35kg and the payload weighs 0.1kg. The tether is set to carry a negligible mass.

The system was commanded to perform aggressive flight maneuvers to excite different modes of the system, as depicted in Fig. 3.6. The control signal and ground truth state were obtained by MAVROS ROS messages. The acquired data was synchronized and sampled at $h = 30$ ms. A total of 40 different flights were conducted with a duration of 60 seconds. A sliding window approach with stride 1 was used to generate sequences from the flight data recorded, including $M = 50$ time steps (1.5s) of the past state and control inputs to the system and the future command of $N = 25$ time steps (0.75s). The model is trained to predict the state of the system for the next N time steps.

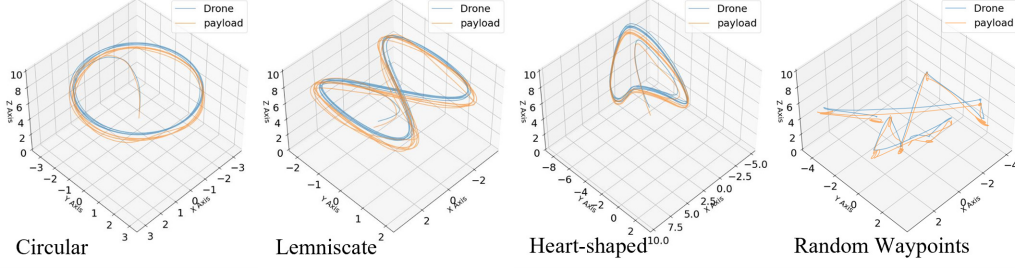


Figure 3.6: Families of the trajectories executed by the CSPQ system during data acquisition.

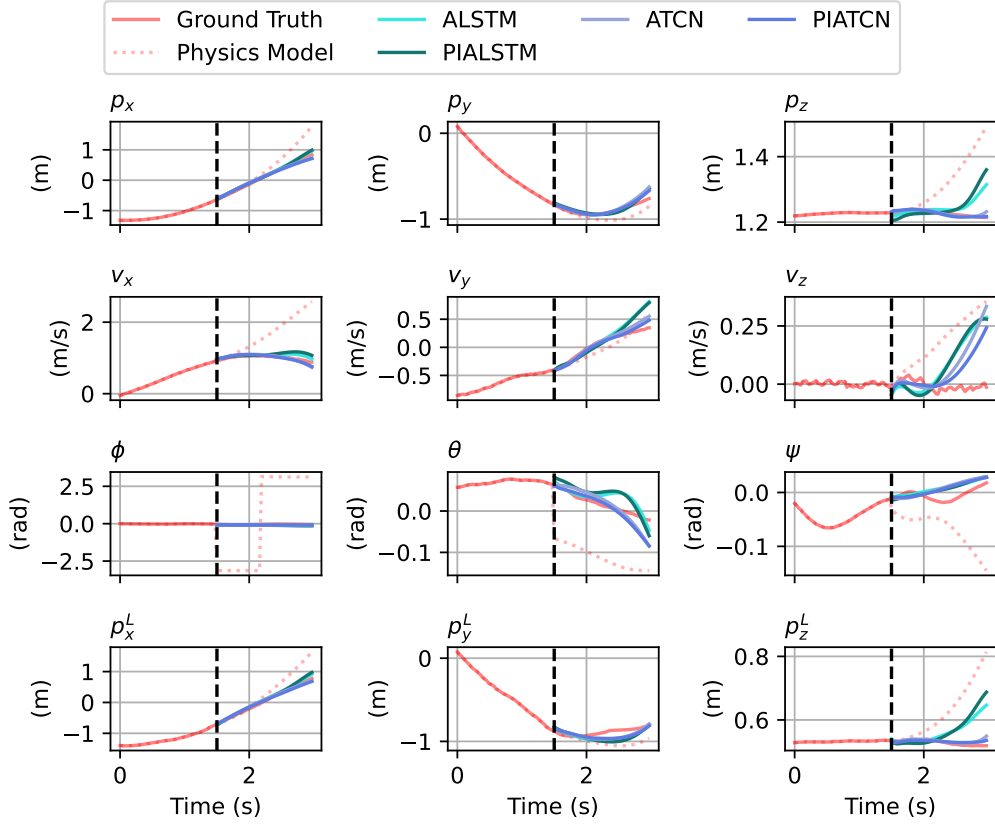


Figure 3.7: Prediction of the vehicle position, linear velocity, orientation, and the position of the payload as a function of time for a single sequence of the test set. The black dotted line indicates the separation between past state and state prediction

Table 3.1: Comparison of RMSE Errors Between PI-TCN and Baseline Models

	Physics model	ALSTM	PIALSTM	ATCN	PIATCN
Position [m]	0.458	0.531	0.427	0.250	0.151
Velocity [m]	0.924	0.415	0.207	0.423	0.077
Orientation [rad]	0.245	0.047	0.083	0.100	0.050
Payload Position [m]	0.424	0.953	0.490	0.346	0.319
Combined	2.051	1.946	1.207	1.119	0.597
Average Inference [ms]					
L40	0.020	18.37	19.23	17.14	19.62
Nvidia Jetson Nano	2.14	110.55	114.51	119.46	127.85

3.3.3.6 Model Training and Prediction Performance

The proposed models and the benchmarking models were trained in the PyTorch environment. This was accomplished on Ubuntu 22.04 OS with an Intel Xeon Gold 6348 Processor, NVIDIA L40 GPU and 64GB DDR5 RAM.

Regarding the PIATCN architecture, the TCN part comprises three hidden layers with filter sizes of 32, 64, and 128 respectively, each using a kernel size of 3 and dilation rates of 1, 2, and 4. Each TCN layer is followed by a ReLU activation, batch normalization, and a dropout regularizer with a dropout probability of 0.1. The hidden-state vectors are passed to the Bahdanau attention layer with a hidden size of 20, which computes a set of attention weights reflecting the relevance of each feature to the current decoder state and forms a context vector as a weighted sum of the encoder features. Each model is trained using the Adam optimizer with a learning rate of 1×10^{-4} and a l_2 weight decay of 0.05. The training procedure is performed for 3,000 epochs. The hyperparameters $\lambda_1 = 1$ and $\lambda_2 = 0.2$ were considered.

We compared the predictive performance of PIATCN against four baseline models on unseen trajectories collected in the real world. The first one is the nominal state transition model, as shown in equation (3.28). The second one is an LSTM encoder-decoder model with the same attention mechanism (ALSTM) with no physics loss. The third one is also attention LSTM but with physics loss included. The fourth one shares the same structure as PIATCN but with no physics loss.

For these experiments, we used the root mean squared error (RMSE) between ground truth and predicted accelerations. Table 3.1 reports the predictive performance in terms of accuracy and computational time of PIATCN and the baselines. The results indicate that, for modeling time series with complex dynamics, TCN-based approaches outperform LSTM-based models due to their ability to process the entire sequence simultaneously and effectively capture temporal dependencies.

Fig. 3.7 illustrates the predictive performance during a representative aggressive maneuver. All models demonstrate accurate predictions for low-speed trajectories, such as the ellipse and random waypoint scenarios. However, as speed increases, complex aerodynamic effects acting on the quadrotor platform significantly degrade prediction accuracy. In particular, the physics-based model is no longer able to guarantee accurate predictions under these conditions. In the figure, the time series predicted by the neural network models are shown as dashed lines, while the expected (ground truth) values are shown as solid lines. For comparison, the predictions from the discretized dynamics model are depicted with dotted lines.

The comparison of inference times in Table 3.1 highlights the computational efficiency of the physics-based model using the L40 GPU card. To check its real-time capability, we also tested the model in the Nvidia Jetson Nano onboard PC, in which the computation resource is limited. with an average inference time of just 0.02 ms, significantly faster than all learning-based baselines. While PIATCN achieves the best overall accuracy across metrics, its inference time (19.62 ms) is much higher. The situation is worsened when we use the Nvidia Jetson Nano platform, where the long inference time (around 100 ms) makes it not applicable for real-time applications. This substantial difference in computation time underscores why the physics-based model remains essential for real-time numerical optimization tasks, where rapid calculations are critical for system performance and stability.

3.4 Concluding Remarks

This chapter presented the comprehensive mathematical models for both quadcopter and CSPQ systems, using the standard Newton-Euler approach and Euler-Lagrange formulation. We also presented a Physics-Informed Attention Temporal Convolutional Network (PIATCN) for modeling and controlling Cable-Suspended Payload Quadcopters. We set up a TCN network with an attention mechanism and incorporate the physics-based nominal dynamics model derived from the Euler-Lagrange formulation into the loss function. Comprehensive simulations demonstrated the effectiveness of PIATCN in predicting the dynamics of CSPQs during various flight scenarios, including aggressive maneuvers. The results showed that PIATCN outperformed both purely physics-based models and learning-based baselines, achieving lower root mean squared error (RMSE) across key state variables such as position, velocity, orientation, and payload position.

Unfortunately, the trained model is not applicable for the real-time numerical optimization process due to its long inference time. For future work, we aim to reduce its inference time as well as design an efficient pipeline to integrate large networks into the optimization pipeline, such as model predictive control. In the following chapters, we are using the quadcopter model and the CSPQ model from the Newton-Euler approach for the optimization pipeline.

Chapter 4

Homotopy-based Trajectory Planner

This chapter addresses the problem of global path planning, which is crucial for optimizing the performance of UAV delivery systems by identifying routes that minimize energy consumption as well as other objectives like time. This not only extends the range and flight time of UAVs but also reduces operating costs, carbon emissions, and the risk of mid-flight battery failures. However, in complex environments, the existence of multiple homotopy paths poses a significant challenge, as these distinct trajectories belong to different topological classes and cannot be continuously deformed into one another without crossing obstacles. Optimizing within a single homotopy class risks being trapped in local optima, especially given the non-linear effects of wind dynamics and environmental constraints.

To address these challenges, we propose a homotopy-augmented lazy Theta* trajectory planner that generates topologically distinct paths. A fast optimization algorithm based on the timed elastic band (TEB) is used to evaluate and refine candidate paths, considering wind dynamics for energy-efficient UAV delivery. By reducing the problem to two dimensions for level flight, the approach ensures computational efficiency while preserving accuracy. Numerical simulations demonstrate the effectiveness of the proposed method in generating obstacle-free, energy-efficient trajectories for parcel delivery.

4.1 Introduction

Efficient path planning between waypoints is essential for optimizing the performance of UAV delivery systems. By identifying the most energy-efficient routes, it can reduce the amount of energy needed for a delivery mission [Hong et al., 2021, Zhang et al., 2023a]. Therefore, the range and flight time of the UAV could be improved, as well as reducing operating costs and carbon emissions. Moreover, energy-efficient path planning can help minimize

the risk of mid-flight battery failures, which can be particularly dangerous in UAV delivery systems. Additionally, optimizing paths can increase the accuracy and speed of delivery, thereby improving customer satisfaction.

In many applications, however, there are more metrics to rate a given route. For example, a path slightly longer than the shortest path but with fewer turns is sometimes more preferred by the user, as the total flight time and energy cost are reduced as the route can carry more speed. In other words, one cannot clearly say whether the shortest or the fastest route is “the best”.

One of the key reasons behind this is that multiple homotopy paths exist in navigating complex environments. A homotopy path refers to a trajectory that belongs to a distinct topological class, meaning it cannot be continuously deformed into another path without crossing obstacles. Exploring these distinct paths is crucial because optimized trajectories often get trapped in local optima due to highly non-linear dynamics, such as wind effects or environmental constraints. If the planner only considers a single homotopy class, it risks overlooking better solutions that may exist in other classes. By generating and evaluating topologically distinct paths, the planner can explore the full solution space and identify the global optimum. Our proposed approach ensures that the UAV achieves the most efficient trajectory, both in terms of energy usage and obstacle avoidance, while preventing the planner from being limited by local optima.

Another concern would be energy consumption under wind dynamics, a factor that has been largely absent from the literature. Traditional trajectory optimization mainly optimizes the mission time spent or the total distance traveled with consideration of UAV dynamics. Nonetheless, the force from the wind field experienced by the flying vehicle through a region is non-negligible. Intuitively, regions of favorable tailwinds could be exploited to reduce battery consumption [Jardin and Bryson Jr, 2001]. In this scenario, the optimization of trajectories in terms of time and thus energy consumption is an interesting and challenging problem.

Regarding UAV trajectory planning in the presence of wind, current publications mainly rely on optimal control formulation. One of the early ap-

proaches formulated the problem as an optimal Dubins path planning problem in wind [Techy and Woolsey, 2009]. A simple analytical solution can be derived. Ayhan and Kwan propose a numerical solution of a time-constrained bang–bang–bang-type external trajectory using optimal control-based methods [Ayhan and Kwan, 2018]. Pontryagin’s minimum principle can also be applied to determine the optimal heading angle for aircraft during cruise [Sridhar et al., 2014, Vinter and Vinter, 2010]. Pradeep et al. [Pradeep et al., 2020] used the Rapid Refresh (RR) operational weather prediction system from the National Center for Environmental Prediction in the US for the analysis, and the optimal control model with an energy consumption performance index for a VTOL is presented. Patrikar et al. [Patrikar et al., 2020] formulate a boundary value problem (BVP) connecting two states in the wind field as a nonlinear constrained optimization problem. It would accelerate the BVP solving time.

In the meantime, the Timed Elastic Band (TEB) algorithm, originating from the classical Elastic Band algorithm [Rösmann et al., 2017b], performs the optimization by deforming an initial trajectory subject to custom constraints. The TEB algorithm offers high flexibility and can incorporate several constraints based on different settings and needs. The multi-objective optimization of the TEB algorithm is dependent on only a few consecutive configurations, leading to a sparse structure for the underlying optimization problem, resulting in high speed performance [Rösmann et al., 2013]. It has shown wide robustness and applicability, and there is research extending its capabilities. Chung et al. [Chung et al., 2022] proposed a Distributed Timed Elastic Band (DTEB) Planner, which incorporates trajectory sharing between a fleet of robots to handle both reactive and predictive collision avoidance. Wu et al. extended TEB by adding constraints on the shortest local path to generate a more efficient path [Wu et al., 2021].

In this chapter, motivated by the aforementioned local optima path and wind dynamic concern, we propose a lazy Theta* trajectory planner with consideration of topologically distinct paths. To quickly evaluate and select the best homotopy class, we also proposed a fast algorithm based on TEB for generating wind-efficient UAV paths for parcel delivery as a preliminary

optimization. Considering level flight only, this method converts a three-dimensional AGL-based flight planning to a two-dimensional problem.

The rest of the chapter is organized as follows. Section 4.2 introduces the proposed drone path planning algorithm, including the search-graph construction, homotopy-augmented lazy Theta* search algorithm, and improved time-elastic band approach using wind dynamics. Section 4.3 presents numerical results and simulations of the path planning algorithm. Section 4.4 is the concluding remarks.

4.2 Methodology

This section presents the proposed trajectory planning algorithm by a two-stage approach. The first stage generates a subset of admissible candidate paths in different homotopy classes. It is performed by augmenting the lazy Theta* search algorithm [Nash et al., 2010] with the notion of edge penalty. The penalty method itself is quite straightforward and has been extensively implemented in transportation networks for exploring alternate routes [Kobitzsch et al., 2013, Bader et al., 2011]. The second stage involves a fast trajectory optimization for all candidate paths using the TEB algorithm augmented with wind consideration so that the global optimal energy-efficient trajectory can be revealed.

The terrain and wind data for the Hong Kong urban area are used as an illustrative example. Fig. 4.1(a) shows the area of interest of 2 km by 2 km. The terrain elevation data and building data are issued by the Hong Kong Lands Department. Fig. 4.1(b) shows the height map for the area of interest.

4.2.1 Homotopy Class Path Planning

4.2.1.1 Definition of Homotopy

We use the concept of homotopy to define the topologically distinct candidate trajectories; thus, a formal definition of homotopy is required. It can be illustrated more clearly in 2-D Euclidean space, even though the concept of

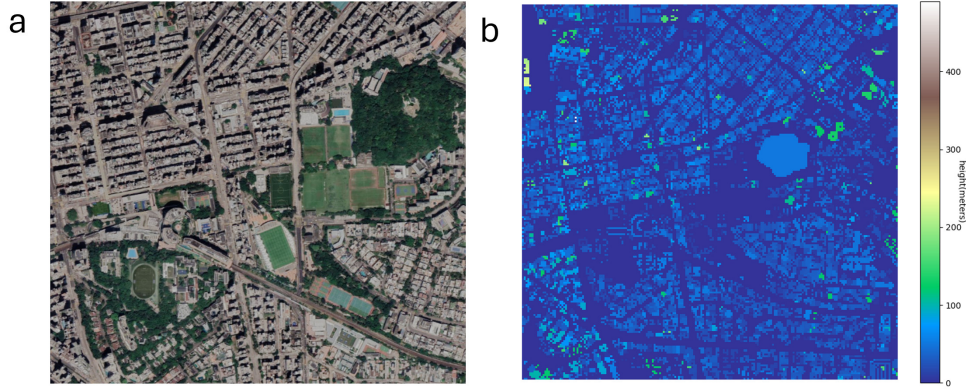


Figure 4.1: (a) Top view of the selected terrain for the study. (b) Heightmap of the selected terrain

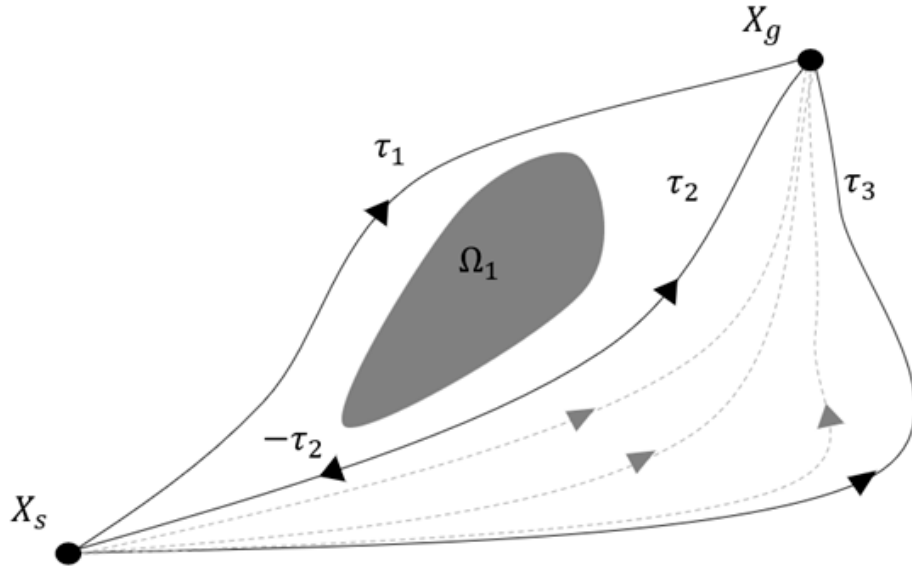


Figure 4.2: τ_1 and τ_2 are in different homotopy classes, where τ_2 and τ_3 are in the same homotopy class.

homotopy is a very general relation on topological spaces. The proposed work follows the definition of homotopy class and homologous trajectories stated in [Bhattacharya, 2010].

Definition 1 (Homotopy Trajectories). Two trajectories (or paths), τ_1 and τ_2 are in the same homotopy class if and only if one can be smoothly deformed into the other without intersecting obstacles. Otherwise, they belong to different homotopy classes. Consider two arbitrary parameterized paths parameterized by $t \in [0, 1]$, $\tau_1 : [0, 1] \rightarrow \mathbb{R}^2$ and $\tau_2 : [0, 1] \rightarrow \mathbb{R}^2$. They share the same start point $\tau_1(0) = \tau_2(0) = x_{\text{start}}$ and end point $\tau_1(1) = \tau_2(1) = x_{\text{goal}}$. τ_1 and τ_2 are homotopic iff there exist a continuous map γ

$$\gamma : [0, 1] \times [0, 1] \rightarrow \mathbb{R}^2 \quad (1)$$

Such that

$$\gamma(t, 0) = \tau_1(t) \quad (2)$$

$$\gamma(t, 1) = \tau_2(t) \quad (3)$$

$$\gamma(0, t) = \tau_1(0) = \tau_2(0) = x_{\text{start}} \quad (4)$$

$$\gamma(1, t) = \tau_1(1) = \tau_2(1) = x_{\text{goal}} \quad (5)$$

This map is the homotopy from τ_1 to τ_2 . The first argument encodes the parameter for the path image, where 0 represents the start point and 1 represents the end point. The second argument encodes the level of deformation between τ_1 and τ_2 , where 0 means the path image is τ_1 itself and 1 means the path image is τ_2 itself.

Definition 2 (Homotopy Class of Trajectories). Two trajectories τ_1 and τ_2 connecting the same start and end coordinates, x_{start} and x_{goal} respectively, are homologous if and only if $\tau_1 \cup -\tau_2$ form the complete boundary of a 2-D manifold in $\mathbb{R}^2$, not containing or intersecting any obstacles.

It is difficult to compute the closed form of homotopy classes, whereas

the computation of homology classes is much simpler. Bhattacharya, Kumar, and Likhachev state that for the context of most motion planning cases, definitions 1 and 2 can be treated as equivalent, although some exception cases exist [Bhattacharya, 2010]. A detailed mathematical explanation about homotopy and homology can refer to Hatcher [Hatcher, 2005] and Bhattacharya [Bhattacharya et al., 2012] which provide a detailed mathematical proof for the equivalence condition. Therefore, as the path planning and optimization are restricted to 2D Euclidean space, determining the homotopy relation of two paths is much simpler by homology checking.

Practically, we consider the polygon spanned by τ_1 concatenated with the $-\tau_2$, which is in reverse orientation. If the polygon is not containing obstacles, we can smoothly deform one bounding line of the polygon into the other one. It is worth noting that for the case of polygons with self-intersections, both sub-polygons should also be filled. A scanline filling algorithm should be adopted instead of other developed methods like flood-fill algorithms [Werner and Feld, 2014].

A graphical illustration is shown in Fig. 4.2. There are three paths τ_1, τ_2 and τ_3 , $-\tau_2$ represents an opposite direction. τ_1 and τ_2 are in different homotopy classes because the polygon spanned by τ_1 and $-\tau_2$ contains obstacle Ω_1 . On the other hand, τ_2 and τ_3 are homotopic as the concatenation of $-\tau_2$ and τ_3 contains no obstacles. The dotted lines inside the spanned polygon indicate the possible deformed paths.

4.2.2 Augmented Lazy Theta* with Penalty Algorithm

The lazy Theta* algorithm originates from the Theta* algorithm, which is a method to find any-angle paths [Akgün et al., 2000]. In A* the parent of any node is constrained by its neighbor, whereas in Theta* the parent of a node can be any predecessor node in between the start point and the node. Line-of-sight checks are needed to identify whether the paths between nodes do not intersect obstacles indeed. This allows Theta* to explore more potentially optimal paths than A*. In the meantime, for each node, Theta* performs line-of-sight checks for each unexpanded visible neighborhood. If some of the

neighbors are never expanded, the line-of-sight checks for the neighbor would be wasted. The notion of lazy Theta* is to reduce the number of line-of-sight checks.

Algorithm 1 LazyThetaStar

Input: Graph $G(V, E)$, s_{start} , s_{goal}
Output: Path $\{q\}$

```

1:  $open\_list \leftarrow \emptyset$ ,  $closed\_list \leftarrow \emptyset$ 
2:  $g(s_{\text{start}}) \leftarrow 0$ 
3:  $parent(s_{\text{start}}) \leftarrow s_{\text{start}}$ 
4:  $open\_list.Insert(\{s_{\text{start}} : g(s_{\text{start}}) + h(s_{\text{start}})\})$ 
5: while  $open\_list \neq \emptyset$  do
6:    $s \leftarrow open\_list.pop()$ 
7:    $SETVERTEX(s)$ 
8:   if  $s = s_{\text{goal}}$  then
9:     return Path( $s$ )
10:  end if
11:   $closed\_list \leftarrow closed\_list \cup \{s\}$ 
12:  for all  $s' \in neighbor(s)$  do
13:    if  $s' \in closed\_list$  then
14:      continue
15:    end if
16:    if  $s' \notin open\_list$  then
17:       $g(s') \leftarrow \infty$ 
18:       $parent(s') \leftarrow NULL$ 
19:    end if
20:     $EDGERELAX(s, s')$ 
21:  end for
22: end while
23: return  $\emptyset$ 

```

Algorithm 2 EdgeRelax(s, s')

```

1:  $g_{\text{old}} \leftarrow g(s')$  ▷ Store the old cost of  $s'$ 
2: // Path 2: Check if the new path is better
3: if  $g(parent(s)) + EdgeCost(parent(s), s') < g(s')$  then
4:    $parent(s') \leftarrow parent(s)$ 
5:    $g(s') \leftarrow g(parent(s)) + EdgeCost(parent(s), s')$ 
6: end if
7: if  $g(s') < g_{\text{old}}$  then
8:   if  $s' \in open\_list$  then
9:      $open\_list.Remove(s, s')$ 
10:  end if
11:   $open\_list.Insert(\{s' : g(s') + h(s')\})$ 
12: end if

```

The pseudocode of the lazy Theta* search algorithm is shown in Algorithm 1. Similar to A*, the algorithm keeps track of two data structures, the *open_list* and the *closed_list*. The G-value for the current node is the cost from the start node to the current node. The H-value is the heuristic cost from the current node to the goal. For each iteration, the node with

Algorithm 3 SetVertex(s)

```

1: if NOT lineofsight(parent( $s$ ),  $s$ ) then
2:   Path 1: Update parent and cost
3:   parent( $s$ )  $\leftarrow$  argmin( $s' \in \text{neighbor}(s) \cap \text{closed}(g(s') + \text{EdgeCost}(s', s))$ )
4:    $g(s) \leftarrow \min (s' \in \text{neighbor}(s) \cap \text{closed}(g(s') + \text{EdgeCost}(s', s))$ )
5: end if

```

Algorithm 4 Lineofsight(s, s')

```

1: // Bresenham computes all points  $s_{\text{in}}$  between  $s$  and  $s'$ 
2: for all  $s_{\text{in}} \in \text{Bresenham}(s, s')$  do
3:   if occupancy( $s_{\text{in}}$ ) = TRUE then
4:     return FALSE
5:   end if
6: end for
7: return TRUE

```

the lowest g-value added to the h-value is extracted from the open_list for expansion and then added to the closed_list. Lazy Theta* deviated from A* and Theta* for function *EdgeRelax* (line 20). It assumes all neighbors have line-of-sight with the current node and perform edge relaxation accordingly. The actual line-of-sight check is performed only when the current node is extracted from the open list, which is function *SetVertex* in line 7. If the line-of-sight check fails, *SetVertex* updates the parent and g-value of the current node by choosing the neighbor of the node that is in the close list with the lowest g-value. It reduces the number of line-of-sight checks because in Theta* it is performed for every discovered neighbor in the expansion stage.

The pseudocode of the augmented lazy Theta* search algorithm is shown in Algorithm 5. The path set is the subset of all paths in distinct homotopy. H_{max} represents the maximum number of homotopy classes to be found and K_{max} , represents the maximum number of iterations. For every iteration, the candidate path is found by the lazy Theta* algorithm (line 6). The homotopy check is done from lines 7 to 13. The candidate concatenates with every path representative in the path set. If the result polygon does not contain obstacles, it indicates that the candidate and path representative are in the same homotopy class, which means the candidate is not distinct and not included in the output set. At the end of the iteration, every edge in the generated path inflates the corresponding edge cost in the graph (lines 17 to 19), which encourages other edges to be found in the next iteration.

Algorithm 5 Augmented Lazy Theta***Input:** Graph $G(V, E)$, obstacles, s_{start} , s_{goal} , H_{max} , K_{max} **Output:** Path set $\pi = \{q_1, q_2, \dots, q_k\}$

```

1:  $\pi \leftarrow \emptyset$ 
2: for  $i = 1$  to  $K_{\text{max}}$  do
3:   if  $\pi.\text{size}() \geq H_{\text{max}}$  then
4:     return  $\pi$ 
5:   end if
6:    $path \leftarrow \text{LazyThetaStar}(G, s_{\text{start}}, s_{\text{goal}})$ 
   // Homotopy class check
7:    $newHomotopy \leftarrow \text{TRUE}$ 
8:   for all  $path' \in \text{optimal\_path\_set}$  do
9:      $polygon \leftarrow path \cup -path'$ 
10:    if  $polygon \cap \text{obstacles} = \emptyset$  then
11:       $newHomotopy \leftarrow \text{FALSE}$ 
12:    end if
13:  end for
14:  if  $newHomotopy$  then
15:     $\pi \leftarrow \pi \cup path$ 
16:  end if
  // Inflate edge cost
17:  for all  $edge \in path$  do
18:     $E.\text{edge} \leftarrow E.\text{edge} \times (1 + p)$ 
19:  end for
20: end for
21: return  $\pi$ 

```

4.2.3 TEB with Consideration of Wind Dynamics

In this chapter, the timed elastic band-based (TEB) approach [Roesmann et al., 2012] is adopted for the path optimization problem for each homotopic alternative path. It extends the conventional "elastic band" algorithm [Zeng et al., 2019] which deforms a predefined global path with consideration of the minimal path length while avoiding obstructions. TEB explicitly takes temporal motion characteristics in terms of dynamic constraints, such as limited velocities and accelerations, into account. The TEB's objective functions can be divided into two classes: Constraints such as velocity and acceleration limits are expressed in terms of penalty functions and objectives for trajectories, such as the shortest or quickest paths or obstacle avoidance.

The state space of a UAV system is explicitly defined by the position and linear velocity together with the orientation and the angular velocity:

$$X = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \Theta, \psi, \dot{\phi}, \dot{\Theta}, \dot{\psi}]^T \quad (4.1)$$

where $[x, y, z]^T$ is the position vector, and the orientation is parametrized

by Euler angles $[\phi, \Theta, \psi]^T$. Due to the fact that the dynamical model of a quadrotor with four inputs is differentially flat [Mellinger and Kumar, 2011], the inputs can be rewritten as functions of four flat outputs and their derivatives. As our UAV is constrained to level flights, the path of the UAV is at a constant height. Our choice of flat outputs is therefore reduced to three independent 1-D functions with respect to time:

$$s(t) = [x(t), y(t), \psi(t)]^T \quad (4.2)$$

In the Timed Elastic Band (TEB) representation, the trajectory is discretized into a set of N control poses $\{s_i = [x_i, y_i, \psi_i] \mid i = 1, 2, \dots, N\}$. The time component is explicitly represented by strictly positive time intervals

$$\{\Delta T_i \in \mathbb{R}^+ \mid i = 1, 2, \dots, N - 1\}$$

The trajectory is defined as the collection of pose and time sequences [Quinlan and Khatib, 1993]:

$$S = \{s_1, \Delta T_1, s_2, \Delta T_2, \dots, s_{n-1}, \Delta T_{n-1}, s_n\} \quad (4.3)$$

The TEB optimization problem is formulated in terms of both control poses and time differences as a multiobjective optimization problem [Quinlan and Khatib, 1993]:

$$S^* = \min_S \sum_{k,i} w_k e_{k,i} \quad (4.4)$$

where the problem is transformed into a non-linear least-squares optimization. e_k are the function components encapsulating the constraints and objectives, and w_k are the relative weights. Since our objective is to minimize the total energy consumption of the UAV mission instead of the total transition time, we replace the objective of TEB from the shortest path with that of the most energy-efficient path. The optimal energy cost function is defined as

$$e_{k,i} = E_{\text{level}}(S) \quad (4.5)$$

Herein, S denotes the optimized TEB, and E_{level} denotes the total energy consumption of the TEB. The formulation of E_{level} is based on the energy consumption model for UAV [Zeng et al., 2019]. For a rotary-wing UAV, the propulsion power of the UAV can be expressed as the function of the airspeed V_a :

$$P_{\text{level}}(v_a) = \overbrace{P_0 \left(1 + \frac{3v_a^2}{U_{\text{tip}}^2}\right)}^{\text{blade profile}} + \overbrace{P_i \frac{v_0}{v_a}}^{\text{induced}} + \overbrace{\frac{1}{2}d_0\rho sAv_a^2}^{\text{parasite}} \quad (4.6)$$

P_0 and P_i are UAV hardware-specific constants. U_{tip} denotes the tip speed of the rotor blade. v_0, d_0, s and ρ represents the tip speed, fuselage ratio, rotor solidity, and air density, respectively. The propulsion function consists of three terms. The first term is the blade profile power to overcome the profile drag of the blades. The second term is the induced power to overcome the induced drag inherent from the lift force, and the third term is the parasite power to overcome the parasite friction between the air and the aircraft surface. Airspeed is obtained by the vector difference between the ground speed and the wind speed:

$$\vec{v}_a = \vec{v}_g - \vec{v}_w \quad (4.7)$$

where $\vec{v}_g$ is the ground speed of the UAV and $\vec{v}_w$ is the wind speed. In this study, computational fluid dynamics (CFD) simulations with a given dominant wind direction were conducted. The resulting 3D wind profile with the resolution of $10 \text{ m} \times 10 \text{ m} \times 10 \text{ m}$ is adopted, where the wind vector is obtained through trilinear interpolation. The total energy consumption of the TEB is therefore obtained by integrating propulsion power over the trajectory:

$$E_{\text{level}}(T_t, \{q(t)\}) = \int_0^{T_t} P(\|v_a(t)\|) dt \quad (4.8)$$

Due to the existence of wind dynamics and the nonlinear nature of the propulsion power function, the energy consumption function is highly nonlinear. Meanwhile, assuming the velocity of UAV is strictly positive, the simplified power consumption function is a convex differentiable function of v_a with one term increasing with the square of v_a and the other decreasing inversely with v_a .

The constraint term is defined as a piecewise continuous and differentiable penalty function such that the error value increases with violation of the constraint:

$$e_k(x, x_b, \epsilon, s, n) = \begin{cases} \left(\frac{x - (x_b - \epsilon)}{s} \right)^n & \text{if } x > x_b - \epsilon \\ 0 & \text{if } x \leq x_b - \epsilon \end{cases} \quad (4.9)$$

where x_b denotes the error bound, s denotes the scaling term, n is the polynomial order, and ϵ represents a small deviation from the approximation. Kinematics and dynamic constraints such as robot velocity/acceleration limits and separation between UAV and obstacles are implemented in the format specified above.

4.3 Results

In this section, numerical simulations are carried out to evaluate the performance of the proposed path planning and optimization method. The simulation environment is a desktop computer with an AMD® Ryzen 7 5800x 8-core CPU and 32 GB RAM in Ubuntu 20.04 LTS. The lazy Theta* path planning is coded in C++ 11 and encapsulated with Python bindings. The path optimization part for the method is modified from the C++ ROS package from [Rösmann et al., 2017a].

The static urban environment in Fig. 4.1(a) is adopted as the testing scenario, which represents the building and terrain features of certain areas of Prince Edward, Hong Kong SAR with a dimension of $2 \text{ km} \times 2 \text{ km}$. Ratio 1 : 10 m is used to construct the extracted voxel world. The environment is eventually transformed into a voxel grid of dimension $\{200 \times 200 \times 30\}$.

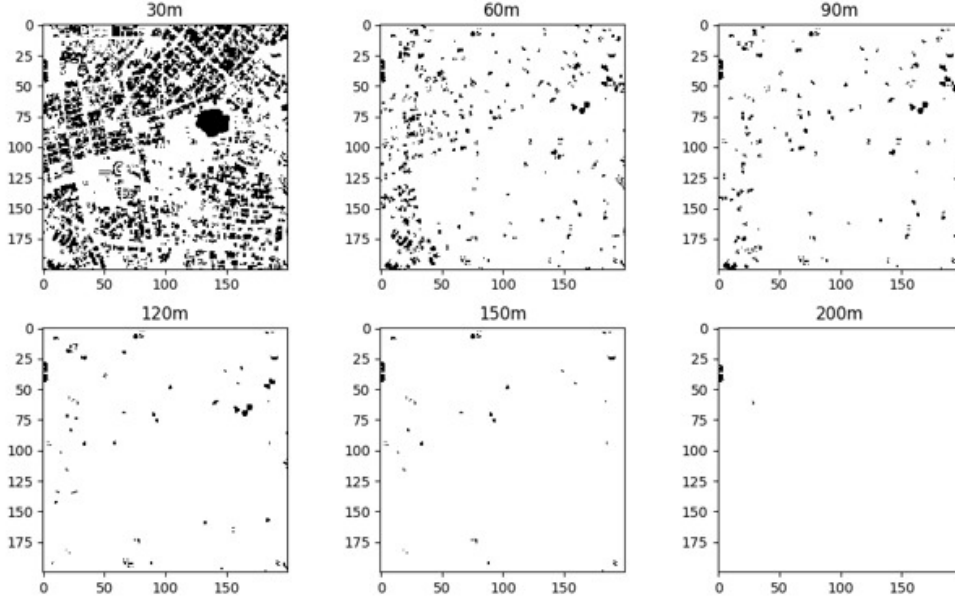


Figure 4.3: The 2D occupancy map at altitudes of 30m, 60m, 90m, 120m, 150 m, and 200 m

The path planning approach is performed on a set of discrete altitudes. Fig. 3 shows the projected 2D occupancy grid map at different altitudes. In the latter simulation, altitudes of 60 m and 150 m are adopted as the 2D occupancy grid for the augmented lazy Theta* for comparison, where 60 m represents the typical low altitude-high obstacle density scenario and 150 m represents the high altitude-low obstacle density scenario.

4.3.1 Augmented Lazy Theta* with Penalty Algorithm Performance

A random origin-destination pair $(10.5, 15.5)$, $(175.5, 189.5)$ is selected for the following analysis. Setting the maximum number of iterations for the penalty algorithm to 50, Fig. 4.4(a) shows that at altitude 60 m, 17 distinct homotopy classes of paths can be found by the augmented lazy Theta* with penalty algorithm. One observation is that the homotopy paths are near-optimal to the best path by lazy Theta* search (the path found at the first iteration), which indicates that they are also good candidates for path opti-

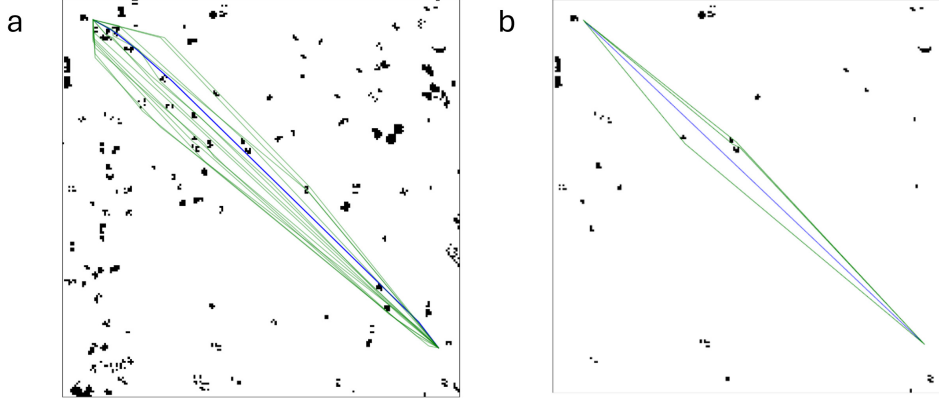


Figure 4.4: (a) Paths in 17 distinct homotopy classes. The blue path represents the shortest path. (b) Paths in 4 homotopy classes at altitude 150 m. The blue path represents the shortest path

Alt. (m)	Class	# Sub-paths	# Nodes	Init. Path Len. (unit)	Final Cost (10^6)
60	1	2	782	240.44	6.343
	2	5	4220	240.19	6.512
	3	3	2079	241.73	6.123
	4	16	11040	241.93	6.218
	5	1	696	243.02	6.033
	6	2	1376	240.23	6.552
	7	7	5089	243.46	6.552
	8	6	4410	244.14	6.086
	9	1	750	243.92	6.178
	10	1	716	244.92	6.262
150	1	19	11970	239.79	6.086
	2	12	7272	240.43	6.188
	3	18	11070	240.44	6.321
	4	1	606	240.71	6.319

Table 4.1: Statistics of the searched paths in different homotopy classes at altitudes of 60m and 150m.

mization. It is because at every iteration the small inflation of edge cost by the previously found path guides the search algorithm, deviating a bit from the found shortest path. Gradually the search algorithm converges towards a set of k shortest paths. At higher altitude with a less complicated obstacle field, Fig. 4.4(b) shows the case with the same OD-pair at altitude 150 m, whereas only 4 distinctive homotopy classes of paths are found within 50 iterations. It is because a lack of effective obstacles leads to less meaningful

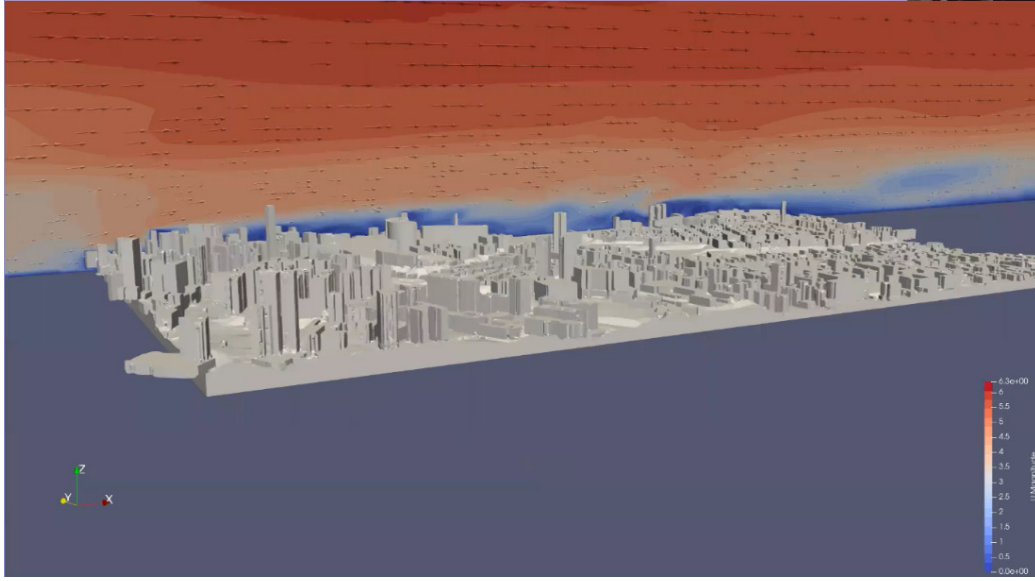


Figure 4.5: Wind velocity distributions over the area of interest

homotopy classes.

Table 4.1 presents the number of sub-optimal paths found and the number of nodes visited for every distinct homotopy class at altitudes of 60 m and 150 m. For brevity, the final cost column represents the final estimated energy consumption index in 4.8. We can observe that in both scenarios the required number of computations for the discovery of a new homotopy class varies a lot. In general, at an altitude of 60 m, it generally takes less than 3 iterations to discover a new homotopy class. However, for classes 2, 4, 7, and 8, it takes 5, 16, 7, and 6 iterations respectively. It is due to these classes spanning a larger free space compared to other classes. It requires more iterations to fully explore the class space. And on this basis, the number of iterations to discover new homotopy classes is even larger for altitude 150 m. The only exception is homotopy class 4, which is discovered at the last iteration.

4.3.2 TEB with Consideration of Wind Dynamics Performance

The wind field data in our work were generated using computational fluid dynamics (CFD)-based simulations with a given dominant wind coming from the west, with a magnitude of 6 m/s. The wind magnitude is smaller at lower

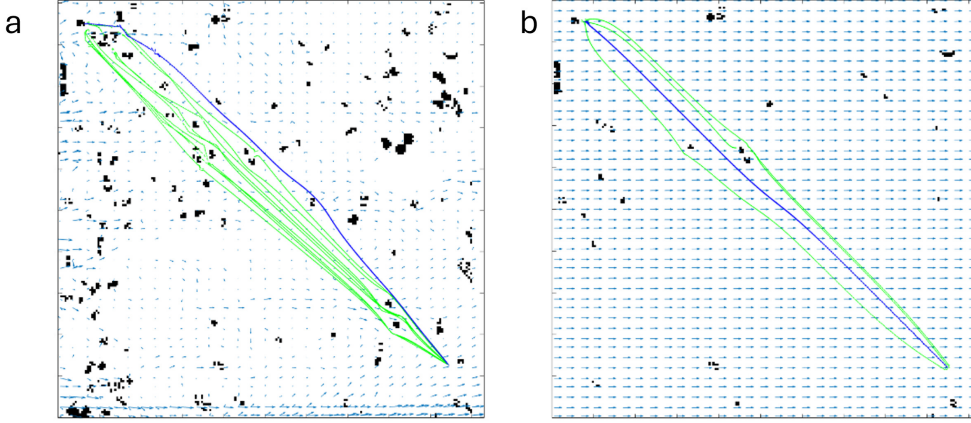


Figure 4.6: (a) TEB optimized trajectory in 17 distinct homotopy classes at altitude 60 m. The blue trajectory represents the shortest path, which belongs to different homotopy classes to the best path in Fig. 4.4(a). (b) TEB optimized trajectory in 4 distinct homotopy classes at altitude 150 m. The blue trajectory represents the shortest path, which is the same homotopy class to the best path in Fig. 4.4(b)

altitudes and more irregular in direction. At higher altitudes, the wind field is more uniform. Fig. 4.5 shows the cross-section view of the generated CFD data; the intensity is in a red-to-blue scale, where red represents 6 m/s and blue represents 0 m/s.

For the path optimization phase, the top 10 homotopy classes of paths are selected for the latter TEB optimization. The results for altitudes of 60 m and 150 m are shown in Fig. 4.6(a) and 4.6(b) respectively. The green paths are the TEB path candidates, and the blue path is the global optimal path in terms of the energy consumption cost, which is the objective function.

Table 4.1 summarizes the initial path length for every homotopy class and their final energy consumption cost optimized by TEB. It can be observed that at altitude 60 m, the global optimal path is in class 5, which is not the class with the shortest path found by lazy Theta* (homotopy class 1). The reason is that optimization of the best path by lazy Theta* gets stuck at local optimum due to the presence of the obstacle. Meanwhile, homotopy class 5 escapes from the dense obstacle field and takes a detour on the right. The

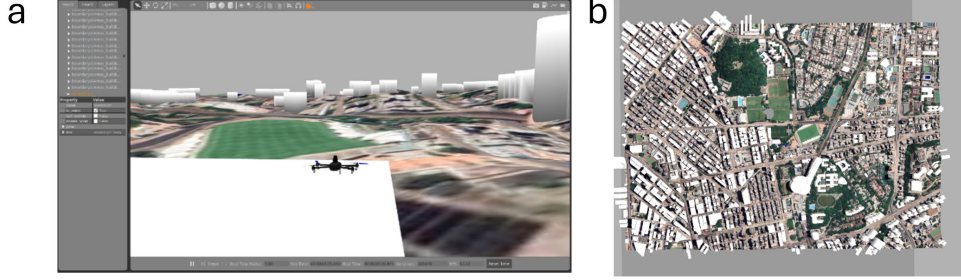


Figure 4.7: (a) Screenshot of the gazebo GUI showing the simulated environment of the area of interest. (b) The top view of the simulation terrain using the OpenStreetMap data

optimization process sacrifices the total distance traveled, and in return the drag and headwind experienced are reduced, and the path is smoother. It leads to a lower consumption of value $6.033e6$. On the contrary, the obstacle field is less dense at altitude 150 m. The best homotopy class path found in lazy Theta* search still outperforms other paths after TEB optimization, with the cost value $6.086e6$.

4.3.3 Experimental Validation under ROS and Gazebo Simulation

To conduct comprehensive simulations and experiments, we also built the test platform based on ROS (Robot Operating System) [Quigley et al., 2009b] and the Gazebo environment. ROS provides a flexible system architecture for the communication between different users, robot components, and simulation platforms. Gazebo is a robust physics-based simulator providing a realistic environment for testing and validating robotic systems. In our work, the environment is built upon OpenStreetMap data, which provides detailed and accurate representations of real-world urban settings as shown in Fig. 4.7. It also supports a custom wind plugin so that the wind data from CFD can be implemented in terms of external force acting on the UAV. Fig. 4.7 shows the overview of the urban environment. The UAV simulation module is implemented based on the XTDrone platform, which integrates PX4 flight control software and ROS [Xiao et al., 2020]. It also provides navigation

and path-tracking modules within the simulated environment for testing the paths generated by our proposed method.

To estimate the energy consumption of the UAV, we track the ROS topic `"/mavros/actuator_control"`. The message is published by the mavros node in ROS, which serves as the connection bridge between the PX4 control system and ROS. It is a feedback message about the input to the motors. We take the control values inside, where they are expressed as a normalized percentage ranging from 0 to 1. In the experiment, the cruising speed of the UAV is set to 12 m/s and the two paths are tested at each altitude. The first path is the best path generated by our method, and the second path is the original TEB path, which optimizes travel time and smoothness only. At an altitude of 60 m, numerical simulations show that two paths share roughly the same travel time, with only a slight difference of a few tens of seconds. However, the average throttle percentage exerted by the UAV for our best path is 0.7116, whereas the average throttle percentage in tracking the original TEB path is 0.7459. There is around 4.6% decrease in throttle. At altitude 150 m, the average throttle percentage for our best path is 0.7213, and that for the original TEB path is 0.7489, with around 3.6% decrease in throttle output.

4.4 Concluding Remarks

In conclusion, this chapter presents a path planning algorithm for generating wind-efficient paths for UAV trajectory optimization afterwards. By considering the issues of local optima paths and the challenges of wind dynamics, the algorithm aims to optimize the energy efficiency of UAV flights. A key contribution of the proposed algorithm is its consideration of topological distinctness. By generating paths that are significantly different from each other in structure and characteristics, the algorithm enhances the exploration of the solution space, increasing the likelihood of finding an optimal wind-efficient trajectory.

The results of this research highlight the effectiveness of the proposed algorithm in addressing the challenges faced in UAV path planning for parcel

delivery scenarios. By optimizing paths for wind efficiency and incorporating topological distinctness, the algorithm enables the UAVs to operate more efficiently, maximizing their flight time and enhancing the overall performance of the system.

Future research in this area could focus on further refining the algorithm, considering additional factors such as airspace regulations, dynamic obstacles, and real-time weather data. Additionally, conducting experimental validations and field tests would provide valuable insights into the practical implementation and performance of the algorithm in real-world scenarios.

Chapter 5

Path-Parametric Reformulation Method

The focus of this chapter is the representation of the state trajectory and the safe flight corridor (SFC) for time-optimal motion planning. The global planner described in chapter 4 is fair enough for the initial trajectory. Yet, its efficiency can be further optimized through trajectory optimization. One widely used approach is spatial reformation, as proposed by [Verschuere et al., 2016], where the robot’s kinematics and dynamics are represented in terms of longitudinal coordinates along the path and transverse coordinates perpendicular to it. This method offers two main advantages. First, it introduces the progress variable s , representing motion in the airspace, which simplifies the time-optimal OCP by transforming the infinite-horizon trajectory optimization problem into a finite-horizon one [Spedicato and Notarstefano, 2017]. Second, obstacles are handled in the transverse dynamics [Frasch et al., 2013], with spatial constraints enforced as simple bounds [Spedicato and Notarstefano, 2017].

Yet, conventional spatial reformulation methods have their drawbacks. In this chapter, a novel reformulation method using the so-called Gravity Normal frame is introduced, as well as its motivation and complete derivation. More importantly, a novel safety corridor generation method can be introduced afterwards.

5.1 Introduction

Conventional methods for corridor construction rely on convex approximations of the collision space. Seed points are selected along the initial path, and for each point, the maximum-volume shape, such as boxes [Lin et al., 2020], ellipsoids [Ren et al., 2022], or polytopes [Wu et al., 2025] that can be contained within the environment is computed. The final corridor is then constructed by overlapping all the geometric objects. However, it is difficult to determine how the geometric shapes and volumes interplay with the

dynamical feasibility of the trajectories computed by the optimization procedure [Wu et al., 2025]. Also, the resulting corridor is always irregular and not intuitive for end-users.

An emerging approach over the past decade is to reformulate the system dynamics using path-parameterized coordinates along the predefined reference trajectory [Verschuere et al., 2016, Spedicato and Notarstefano, 2017]. It projects system dynamics onto a moving frame along the path in free space, with the resulting states—path progress and orthogonal distance—combined with the path’s intrinsic properties (tangent, curvature, torsion) [Arrizabalaga and Ryll, 2022]. The spatial reformulation method was first introduced in autonomous driving, where vehicle dynamics were transformed into curvilinear coordinates along the road. This allowed nonlinear obstacle representations to be expressed as linear lateral constraints, enabling decoupled speed optimization along the tangential direction [Frasch et al., 2013]. Also, the notion of path progress and path properties is now embedded in the optimization problem. Verschuere *et al.* extended this approach to three-dimensional motion planning for robot manipulators [Verschuere et al., 2016], while Spedicato *et al.* applied a similar method to quadcopter systems, expressing the quadcopter pose in terms of decoupled tangential and transverse spatial states along the predefined path. This reformulation enabled the formulation of costs and constraints independent of the time variable [Spedicato and Notarstefano, 2017].

Among all these applications, the most common moving coordinate frame construction method along the given trajectory is the Frenet-Serret (FS) frame. It is well-defined for planar paths [Frasch et al., 2013, Kloeser et al., 2020, Li et al., 2024]. However, when it comes to 3D trajectories, the FS frame becomes undefined at inflection points, i.e., points where the curvature vanishes. Additionally, near those inflection points, the FS frame exhibits undesirable rotations (or distortions) about its tangential axis. Meanwhile, although the Parallel Transport (PT) frame presented in [Hanson and Ma, 1995] provides a smooth and singularity-free curve framing method, it contains undesired spinning and is highly dependent on the initial orientation [Ramírez and Nahon, 2021]. Also, the PT frame can only be solved numer-

ically, as it lacks a closed-form analytical solution. The model predictive contouring control (MPCC) method is closely related to spatial reformulation, which has shown its capability in trajectory tracking and autonomous drone racing [Romero et al., 2022b]. Yet, it requires high-accuracy arc-length parameterization for lag error and contour error construction, which also resorts to numerical methods only. Arrizabalaga *et al.* proposes the use of Pythagorean Hodograph (PH) splines with the advantage of analytical derivatives, but transforming from the original trajectory to PH spline is a computationally intense task in itself [Arrizabalaga and Ryll, 2022]. This motivates the search for a robust and efficient solution to the curve-framing method.

To account for those shortcomings, we adopted a new curve framing called the Gravity Normal (GN) frame proposed by Ramírez and Nahon [Ramírez and Nahon, 2021], which was originally used for tracking tracking and control of fixed-wing aircrafts. The intuition is that the attached frame should correlate with the inertial frame to the greatest extent. Specifically, the two transverse axes, which are orthogonal to the tangential axis, are constructed such that one axis is closely related to the lateral error (similar to the cross-track error used for planar curves), and the other one to the direction of gravity. It results in a steadier frame compared to the FS and PT frames and a clearer geometrical interpretation for subsequent flight corridor generation. Secondly, the exact GN frame and its derivatives can be computed analytically for closed-form spatial reformulation.

The rest of the chapter is structured as follows. Section 5.2 presents the motivation behind the GN frame and the spatial reformulation of the three-dimensional Euclidean coordinates into the path-parametric coordinates. Section 5.3 presents the complete flight corridor generation based on the GN frame, and Section 5.4 contains the concluding remarks.

5.2 Path-Parametric Reformulation

In this section, we derive the spatial reformulation of quadcopter kinematics and dynamics with respect to the path’s Gravity Normal (GN) frame. At

first, a recapitulation of the theory of the curve framing theory is presented. In the literature on tracking control theory, the path-parametric frame, such as the Frenet-Serret frame, is often adopted especially in the field of autonomous vehicles [Werling et al., 2010]. It is also applicable to trajectory optimization for UAVs, as path properties and variant spatial constraints are embedded effectively in using tangential and transverse dynamics [Arrizabalaga and Ryll, 2021]. On the other hand, we then illustrate that when it comes to the 3D case, singularity arises when the reference path is partially linear. Moreover, the choice of the Frenet-Serret (FS) frame introduces undesired twists in the tangential component. Also, some other methods, such as the Parallel Transport (PT) frame, experience some other drawbacks. As a result, the Gravity Normal frame is introduced to provide a singularity-free and smooth moving frame throughout the path for spatial reformulation of quadcopter kinematics and dynamics.

To distinguish derivatives with respect to different parameters, we use the dot operator $\dot{\mathbf{x}}$ to denote the first-order time derivative $\partial \mathbf{x} / \partial t$, and the prime operator $\mathbf{x}' = \partial \mathbf{x} / \partial s$ to represent the first-order derivative with respect to the path parameter s .

5.2.1 Preliminaries on Conventional Curve-Framing Methods

Let Γ be the initial reference trajectory resolved in the inertial frame $\{I\}$ depending on the path-parameter s :

$$\Gamma = \{ {}^I\gamma(s) \in \mathbb{R}^3 : s \in [s_0, s_f] \mapsto {}^I\gamma(s) \} \quad (5.1)$$

Throughout the chapter, it is assumed that Γ is provided by an arbitrary global planner, where there is already abundant research on the topic [Zhou et al., 2025]. Also, Γ must be at least twice differentiable with a non-vanishing first derivative. It is an essential assumption for the existence of the *adapted* orthonormal reference frame structured as follows:

$$\{ \mathbf{R}_\Gamma(s) \} = {}^I \begin{bmatrix} \mathbf{e}_1(s) & \mathbf{e}_2(s) & \mathbf{e}_3(s) \end{bmatrix} \in \mathbb{SO}^3 \quad (5.2)$$

for any $s \in [s_0, s_f]$, where the first component basis vector $\mathbf{e}_1(s)$ is always equivalent to the unit tangent at s , i.e. $\mathbf{e}_1(s) = \frac{\mathbf{\Gamma}'(s)}{\|\mathbf{\Gamma}'(s)\|}$. Hence any quantity referenced in $\{I\}$ can be expressed in the curve frame $\{\mathbf{R}_{\mathbf{\Gamma}}(s)\}$. Notice that this is a generic configuration for the adapted frame. In fact, Bishop [Bishop, 1975] showed that there exist infinitely many ways for moving frame configuration on any given space curve.

For $\mathbf{e}_2(s)$ and $\mathbf{e}_3(s)$, they constitute a plane orthonormal to the tangent, yet there exists no natural way to determine these basis vectors [Spivak, 1970]. Also, all basis vectors need to be differentiable to ensure smooth transitions along the curve.

5.2.1.1 Frenet-Serret Frame

In this section we are adopting the curve $\mathbf{\Gamma}$ defined in section 5.2.1. For defining FS, it must satisfy the following additional assumptions:

- $\mathbf{\Gamma}$ is thrice differentiable with a non-vanishing second derivative, and the motion is steady.
- $\mathbf{\Gamma}$ exhibits dynamic feasibility, allowing the vehicle to smoothly navigate without abrupt or impossible maneuvers, as well as excessive speeds that the vehicle cannot handle.
- The curve is regular, meaning the first derivative of the curve is always non-zero, i.e., the arc-length of $\mathbf{\Gamma}$ increases monotonically with time.
- The desired curve is constructed using planning algorithms such that the initial point of $\mathbf{\Gamma}$ coincides with the actual position of the quadcopter.

$\mathbf{\Gamma}$ is typically generated by arc-length reparameterization on the initial path produced by a global path planner, such as a Bézier curve or a B-spline. In the literature, s is also referred to as the progress variable of the curve, since geometrically it can be treated as the progress along the path.

Traditionally, the FS frame formulation [Spivak, 1970] is the most common choice for $\mathbf{R}_{\mathbf{\Gamma}}$. It leverages the geometric properties of the curve, namely

curvature and torsion, which are analytically defined. The three basis vectors at s are referred to as the tangent $\mathbf{t}$, normal $\mathbf{n}$ and bi-normal $\mathbf{b}$, and they are defined as follows:

$$\mathbf{t} = \frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|}, \quad \mathbf{b} = \frac{\mathbf{\Gamma}' \times \mathbf{\Gamma}''}{\|\mathbf{\Gamma}' \times \mathbf{\Gamma}''\|}, \quad \mathbf{n} = \mathbf{b} \times \mathbf{t} \quad (5.3)$$

Clearly, they are of unit length and orthogonal. Thus, the reference frame $\{\mathbf{R}_{FS}\} = [\mathbf{t} \ \mathbf{n} \ \mathbf{b}]$ can be constructed resembling equation (5.2). For readability and conciseness, the dependence on s is omitted for the time being. The differential form of equation (5.3) is given by

$$\begin{bmatrix} \mathbf{t}' \\ \mathbf{n}' \\ \mathbf{b}' \end{bmatrix} = \nu \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{n} \\ \mathbf{b} \end{bmatrix} \quad (5.4)$$

, where we additionally define $\nu(s) = \|\mathbf{\Gamma}'(s)\| \in \mathbb{R}$ as the scalar magnitude of the curve derivative at s for clarity. $\kappa(s) \in \mathbb{R}$ is defined as the curvature of $\mathbf{\Gamma}$ and $\tau(s) \in \mathbb{R}$ is defined as the torsion. They are given by

$$\kappa(s) = \frac{\|\mathbf{\Gamma}'(s) \times \mathbf{\Gamma}''(s)\|}{\|\mathbf{\Gamma}'(s)\|^3} \quad (5.5)$$

$$\tau(s) = \frac{(\mathbf{\Gamma}'(s) \times \mathbf{\Gamma}''(s)) \cdot \mathbf{\Gamma}'''(s)}{\|\mathbf{\Gamma}'(s) \times \mathbf{\Gamma}''(s)\|^2} \quad (5.6)$$

The complete derivation can be found in [Spivak, 1970]. For a curve to possess a well-defined FS frame, it must be at least thrice differentiable, with a non-vanishing second derivative. More importantly, it highlights that the normal vector becomes undefined when the curvature is zero, i.e., when the curve reduces to a straight line. Such cases, however, are often unavoidable, as they frequently appear as outputs from global planners.

Equations (5.5) and (5.6) reveal that singularity occurs when the curvature $\kappa(s)$ vanishes, making the normal and binormal vectors indeterminate. Intuitively, since the normal vector $\mathbf{n}(s)$ always points toward the center of the osculating sphere (or osculating circle in the 2D case), when the curvature diminishes to zero, the path is locally straight, and thus the normal vector

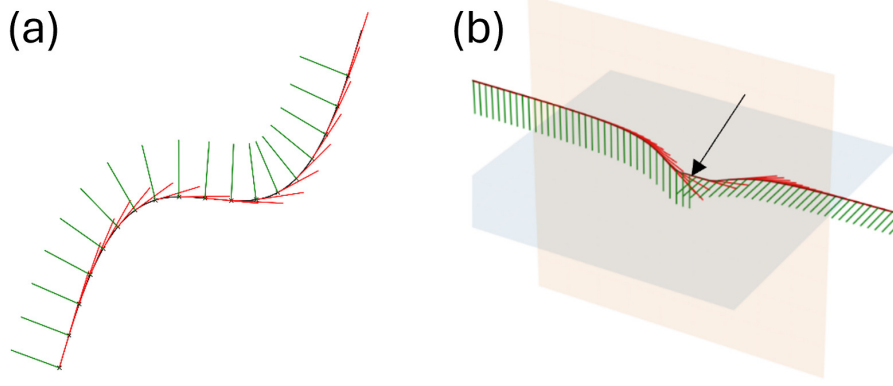


Figure 5.1: Illustration of Frenet-Serret curve framing. (a) Example Frenet frame in a 2D curve (b) Example Frenet frame in a 3D curve

cannot be defined. This happens at straight lines or inflection points, where the path changes from convex to concave or vice versa.

In the 2D case, this issue can be resolved by explicitly defining the binormal vector to always point normal to the plane occupied by the 2D curve, and the normal vector is determined as the cross product of the tangent and binormal vectors. A graphical illustration is shown in Fig. 5.1(a).

Another problem is that near the inflection points, the $\mathbf{n}(s)$ vector abruptly changes its direction, causing an unnecessary rotation in the normal plane. This behavior is shown in Fig. 5.1(b). The twisting behavior can be characterized by the Darboux vector:

$$\mathbf{d}_{FS}(s) = \kappa(s)\mathbf{b}(s) + \sigma(s)\mathbf{t}(s) \quad (5.7)$$

which satisfies $\mathbf{t}'(s) = \mathbf{d}_{FS}(s) \times \mathbf{t}(s)$, $\mathbf{n}'(s) = \mathbf{d}_{FS}(s) \times \mathbf{n}(s)$, $\mathbf{b}'(s) = \mathbf{d}_{FS}(s) \times \mathbf{b}(s)$. The Darboux vector $\mathbf{d}_{FS}(s)$ is the instantaneous angular velocity vector of $\{FS\}$, as derived in [Abbena et al., 2017]. In equation (5.7), the scalar component for $\mathbf{t}(s)$, $\sigma(s)$, corresponds to the rotation rate in the normal plane.

When the curve is partially straight, $\kappa(s)$ vanishes, $\sigma(s)$ diverges, and this causes the normal plane to flip drastically. This characteristic introduces challenges during trajectory optimization processes, as the optimization algorithms may encounter local optima when attempting to optimize the trajectory along these rapidly varying normal directions. Therefore, there is a need to define a moving frame along the reference path in a smooth and well-defined fashion.

5.2.1.2 Parallel Transport Rrame

To resolve these drawbacks, Bishop [Bishop, 1975] proposed what is now known as the Bishop frame or the PT frame. It is designed to ensure that the normal vectors vary smoothly along the path regardless of the curvature. This is achieved by constructing the frame such that the derivatives of the two normal vectors, $\mathbf{n}_1$ and $\mathbf{n}_2$, depend only on the tangent vector $\mathbf{t}$ and not on each other. Thus, the differential form of the PT frame is given by:

$$\begin{bmatrix} \mathbf{t}' \\ \mathbf{n}_1' \\ \mathbf{n}_2' \end{bmatrix} = \nu \begin{bmatrix} 0 & \kappa_1 & \kappa_2 \\ -\kappa_1 & 0 & 0 \\ -\kappa_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{n}_1 \\ \mathbf{n}_2 \end{bmatrix} \quad (5.8)$$

, where $\kappa_1(s)$ and $\kappa_2(s)$ are called the normal development of the curve to be addressed afterwards.

With the initial vectors $\mathbf{t}_0, \mathbf{n}_{1,0}$ and $\mathbf{n}_{2,0}$ chosen to be orthonormal, the rest of the PT frame remains of unit length and mutually orthogonal. This is because their derivatives are always orthogonal to themselves, as shown in equation (5.8). Together, they constitute the frame $\{\mathbf{R}_{PT}\} = \begin{bmatrix} \mathbf{t} & \mathbf{n}_1 & \mathbf{n}_2 \end{bmatrix}$ resemblance to equation (5.2). Notably, equation (5.8) provides the advantage that both k_1 and k_2 are well defined when curvature vanishes, ensuring the robustness of the PT frame.

Meanwhile, $\kappa_1(s)$ and $\kappa_2(s)$ need to be determined. Since both frames contain a tangent, by comparing the tangent component of the Cartan connection matrix of the Frenet-Serret frame $\mathbf{C}_{FS}(s)$ to that of the Parallel Transport frame $\mathbf{C}_{PT}(s)$, we can relate $\kappa_1(s)$ and $\kappa_2(s)$ to the curvature $\kappa(s)$

of the curve as

$$\kappa(s)^2 = \|\mathbf{t}'(s)\|^2 = \|\kappa_1(s)\mathbf{n}_1(s) + \kappa_2(s)\mathbf{n}_2(s)\|^2 = \kappa_1(s)^2 + \kappa_2(s)^2. \quad (5.9)$$

As a result, the magnitudes of κ_1 and κ_2 are independent of $\mathbf{n}_{1,0}$ and $\mathbf{n}_{2,0}$ and hence the $\{PT\}$ frame. To fully determine κ_1 and κ_2 , an extra condition is required, which can be solved by inspecting the parallel condition. Consider two normal fields $(\mathbf{n}_1, \mathbf{n}_2)$ and $(\mathbf{m}_1, \mathbf{m}_2)$ in the Parallel Transport frame. With a slight abuse of notation, define θ as the rotation angle between the two normal fields. We can relate the two normal fields by

$$\begin{bmatrix} \mathbf{n}_1(s) \\ \mathbf{n}_2(s) \end{bmatrix} = \begin{bmatrix} c_{\theta(s)} & -s_{\theta(s)} \\ s_{\theta(s)} & c_{\theta(s)} \end{bmatrix} \begin{bmatrix} \mathbf{m}_1(s) \\ \mathbf{m}_2(s) \end{bmatrix} \quad (5.10)$$

By differentiating equation (10) and ensuring that the derivatives of $\mathbf{n}_1$ and $\mathbf{n}_2$ only consist of the tangent component, θ can be written as

$$\theta(s) = \theta_0 + \int_{s_0}^s \sigma(u) du \quad (5.11)$$

as derived in [Bishop, 1975]. One common choice for κ_1 and κ_2 is

$$(\kappa_1(s), \kappa_2(s)) = (\kappa(s)c_{\theta(s)}, \kappa(s)s_{\theta(s)}). \quad (5.12)$$

Using equations (5.8) to (5.12), we can now construct the $\{PT\}$ frame along the curve using an arbitrary initial frame and numerically integrate equation (13), for example, using the classical fourth-order Runge-Kutta method to obtain θ and thus the $\{PT\}$ frame. Unlike the $\{FS\}$ frame, $\{PT\}$ is still well-defined when the second derivative of the curve vanishes.

Concerning the undesired twisting property, the Darboux vector, or the angular velocity vector corresponding to $\{PT\}$, was derived in studies [Bukcu and Karacan, 2008] as

$$\mathbf{d}_{PT} = -\kappa_2(s)\mathbf{n}_1(s) + \kappa_1(s)\mathbf{n}_2(s) \quad (5.13)$$

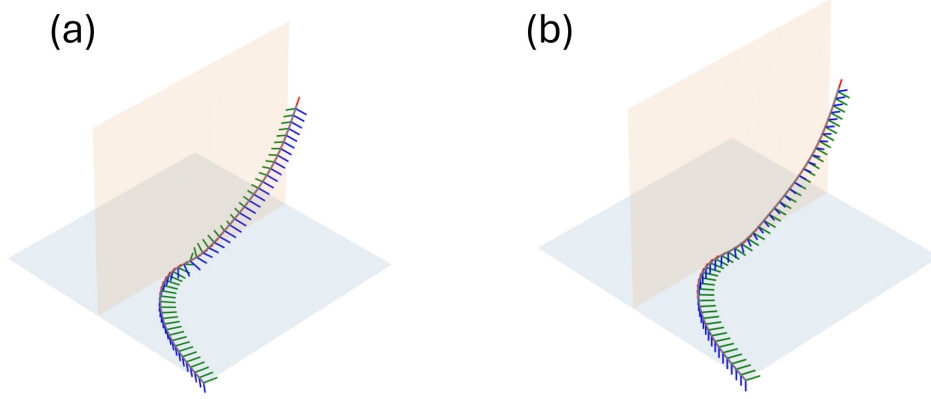


Figure 5.2: Illustration of Frenet-Serret curve framing. (a) The Frenet frame is undefined and changes abruptly at the straight line region. (b) The Parallel Transport frame is smooth throughout.

which satisfies

$$\mathbf{t}'(s) = \mathbf{d}_{PT}(s) \times \mathbf{t}(s), \quad \mathbf{n}'_1(s) = \mathbf{d}_{PT}(s) \times \mathbf{n}_1(s), \quad \mathbf{n}'_2(s) = \mathbf{d}_{PT}(s) \times \mathbf{n}_2(s). \quad (5.14)$$

It illustrates the rotation-minimizing property of $\{PT\}$, as its angular velocity has no component along the tangential direction, i.e., $\mathbf{d}_{PT} \cdot \mathbf{t} = 0$. As a result, there is no instantaneous rotation of the normal plane vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ about the tangent $\mathbf{t}$, avoiding the problem of sudden twisting of the normal plane.

The comparison between the properties of the Parallel Transport frame and the Frenet-Serret frame is illustrated in Fig 5.2. $\{FS\}$ and $\{PT\}$ frames are identical when the initial $\{PT\}$ frame is chosen to be the same as the $\{FS\}$ frame until the point of inflection occurs, as shown in Fig. 5.2b. The Frenet-Serret frame's normal components, depicted as the green axis, switch signs instantly, while the $\{PT\}$ frame has no such discontinuities.

On the contrary, there is no closed form for σ . It can only be determined using numerical integration, which depends on the initial condition σ_0 . When the torsion of the curve is nonzero, the osculating plane (the $\mathbf{n}_1\mathbf{n}_2$ plane) spins around the tangent vector. The notion of path-relative error direction is then

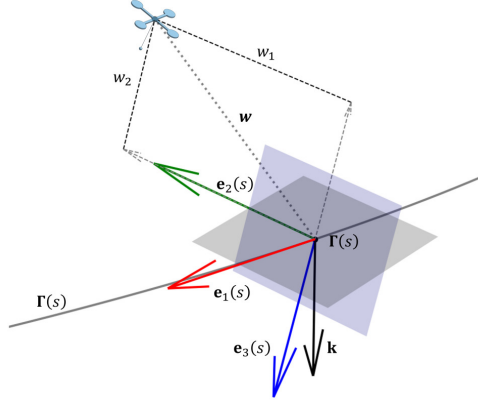


Figure 5.3: Spatial projection of the quadcopter's center onto the reference path $\Gamma(s)$. The resulting transverse distance w is expressed in the GN frame at s , especially along the two transverse axes $\mathbf{e}_2$ (green) and $\mathbf{e}_3$ (blue).

lost. This is because when the frame spins, any quantities expressed in the inertial frame will be distorted drastically [Ramírez and Nahon, 2021].

We argue that neither the FS frame nor the PT frame offers a meaningful interpretation of the physical space, which is essential for describing system states and obstacles. This limitation necessitates the development of an alternative spatial reformulation method.

5.2.2 Gravity-Normal Frame

Define $\mathbf{k}$ as the unit vector pointing at gravity in the inertial frame. The Gravity Normal (GN) frame is defined as follows:

$$\mathbf{e}_1(s) = \frac{\Gamma'(s)}{\|\Gamma'(s)\|} \quad (5.15a)$$

$$\mathbf{e}_2(s) = \frac{\mathbf{k} \times \Gamma'(s)}{\|\mathbf{k} \times \Gamma'(s)\|} \quad (5.15b)$$

$$\mathbf{e}_3(s) = \mathbf{e}_1(s) \times \mathbf{e}_2(s) \quad (5.15c)$$

The three basis vectors in equation (5.15) forms the frame basis $\{\mathbf{R}_{GN}(s)\} = {}^I \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}$ resembling equation (5.2). A graphical illustration is shown in Fig. 5.3. The red, green, and blue arrows represent the $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ axis respectively at $s = s^*$. $\mathbf{k}$ is represented as the black arrow, which always

points downwards in $\{I\}$. Note that this color scheme is used consistently throughout the chapter. The gray plane is the plane normal to $\mathbf{k}$ and the light purple plane is the osculating plane to $\mathbf{e}_1$. By observation, such formulation offers several advantages:

- With the help of $\mathbf{k}$, $\mathbf{e}_2$ represents the lateral direction relative to the curve while remaining in the osculating plane of $\mathbf{e}_1$. It is illustrated in Fig. 5.3, where $\mathbf{e}_2$ lies on both the horizontal plane of $\mathbf{k}$ and the osculating plane of $\mathbf{e}_1$. This consistency provides a clear and meaningful interpretation of lateral distances, which is essential for navigation and control systems. Similarly, $\mathbf{e}_3$ can represent the "altitude" direction relative to the curve as $\mathbf{e}_3$ is pointing downwards as much as possible while remaining normal to the tangent.
- It is well-defined when the curvature κ is zero and contains no inflection points, which outperforms the FS frame.
- It has a closed-form expression that does not rely on numerical integration methods, which outperforms the PT frame.

To obtain the differential form of the GN frame, we resort to the basic property of orthonormal matrices, i.e., $\mathbf{R}_{GN}^T \mathbf{R}_{GN} = \mathbf{I}$. Differentiating with respect to s on both sides, we have

$$\mathbf{R}_{GN}^T \mathbf{R}'_{GN} + \mathbf{R}_{GN}^{T'} \mathbf{R}_{GN} = 0$$

, which means $\mathbf{R}_{GN}^T \mathbf{R}'_{GN}$ is a skew-symmetric matrix. Denoting it as $\hat{\boldsymbol{\lambda}}$, the differential of $\mathbf{R}_{GN}$ can be characterized by

$$\mathbf{R}'_{GN}(s) = \mathbf{R}_{GN}(s) \hat{\boldsymbol{\lambda}}(s), \quad (5.16)$$

$$\hat{\boldsymbol{\lambda}}(s) = \begin{bmatrix} 0 & -\lambda_3(s) & \lambda_2(s) \\ \lambda_3(s) & 0 & -\lambda_1(s) \\ -\lambda_2(s) & \lambda_1(s) & 0 \end{bmatrix} \quad (5.17)$$

, where $\boldsymbol{\lambda}(s) = [\lambda_1(s) \ \lambda_2(s) \ \lambda_3(s)]^T \in \mathbb{R}^3$, which is also known as the angular velocity vector for the moving frame $\{\mathbf{R}_{GN}\}$. To derive the analytical

formula for $\boldsymbol{\lambda}(s)$, we differentiate equation (5.15a), (5.15b) and (5.15c) w.r.t. s as

$$\begin{aligned}\mathbf{e}'_1 &= \frac{\|\boldsymbol{\Gamma}'\|\boldsymbol{\Gamma}'' - \boldsymbol{\Gamma}'(\|\boldsymbol{\Gamma}'\|)'}{\|\boldsymbol{\Gamma}'\|^2} \\ &= \frac{\boldsymbol{\Gamma}''}{\|\boldsymbol{\Gamma}'\|} - \frac{\boldsymbol{\Gamma}'}{\|\boldsymbol{\Gamma}'\|^2} \frac{\boldsymbol{\Gamma}' \cdot \boldsymbol{\Gamma}''}{\|\boldsymbol{\Gamma}'\|} \\ &= \frac{\boldsymbol{\Gamma}''}{\|\boldsymbol{\Gamma}'\|} - \frac{\boldsymbol{\Gamma}'}{\|\boldsymbol{\Gamma}'\|^3} (\boldsymbol{\Gamma}' \cdot \boldsymbol{\Gamma}'')\end{aligned}\tag{5.18a}$$

$$\begin{aligned}\mathbf{e}'_2 &= \left(\frac{\mathbf{k} \times \boldsymbol{\Gamma}'}{\|\mathbf{k} \times \boldsymbol{\Gamma}'\|} \right)' \\ &= \frac{\|\mathbf{k} \times \boldsymbol{\Gamma}'\|(\mathbf{k} \times \boldsymbol{\Gamma}')' - (\mathbf{k} \times \boldsymbol{\Gamma}')\|\mathbf{k} \times \boldsymbol{\Gamma}'\|'}{\|\mathbf{k} \times \boldsymbol{\Gamma}'\|^2} \\ &= \frac{\mathbf{k} \times \boldsymbol{\Gamma}''}{\|\mathbf{k} \times \boldsymbol{\Gamma}'\|} - \frac{\mathbf{k} \times \boldsymbol{\Gamma}'}{\|\mathbf{k} \times \boldsymbol{\Gamma}'\|^3} (\mathbf{k} \times \boldsymbol{\Gamma}') \cdot (\mathbf{k} \times \boldsymbol{\Gamma}'')\end{aligned}\tag{5.18b}$$

$$\mathbf{e}'_3 = \mathbf{e}_1 \times \mathbf{e}'_2 + \mathbf{e}'_1 \times \mathbf{e}_2\tag{5.18c}$$

Multiplying both sides of equation (5.16) by $\mathbf{R}_{GN}^T$, the closed-form of $\widehat{\boldsymbol{\lambda}}(s)$ is expressed as

$$\widehat{\boldsymbol{\lambda}}(s) = {}^I\mathbf{R}_{GN}^T {}^I\mathbf{R}'_{GN} = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix} \begin{bmatrix} \mathbf{e}'_1 & \mathbf{e}'_2 & \mathbf{e}'_3 \end{bmatrix}\tag{5.19}$$

Therefore, $\widehat{\boldsymbol{\lambda}}$ is computed by comparing the coefficients of equations (5.17) and (5.19). For λ_1 , using the triple product identities, we obtain

$$\begin{aligned}
 \lambda_1 &= \mathbf{e}_3 \cdot \mathbf{e}'_2 \\
 &= \left(\frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|} \times \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \right) \cdot \left(\frac{\mathbf{k} \times \mathbf{\Gamma}''}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} - \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|^3} \cdot (\mathbf{k} \times \mathbf{\Gamma}') \cdot (\mathbf{k} \times \mathbf{\Gamma}'') \right) \\
 &= \left(\frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|} \times \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \right) \cdot \left(\frac{\mathbf{k} \times \mathbf{\Gamma}''}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \right) - (\mathbf{\Gamma}' \times (\mathbf{k} \times \mathbf{\Gamma}')) \cdot (\mathbf{k} \times \mathbf{\Gamma}') (\mathbf{O.T.}) \\
 &= \frac{(\mathbf{\Gamma}' \cdot \mathbf{\Gamma}') \mathbf{k} - (\mathbf{\Gamma}' \cdot \mathbf{k}) \mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\| \|\mathbf{k} \times \mathbf{\Gamma}'\|^2} (\mathbf{k} \times \mathbf{\Gamma}'') \\
 &= \frac{(\mathbf{\Gamma}' \cdot \mathbf{k}) \mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\| \|\mathbf{k} \times \mathbf{\Gamma}'\|^2} (\mathbf{\Gamma}'' \times \mathbf{k}) \\
 &= \frac{(\mathbf{k} \cdot \mathbf{\Gamma}') (\mathbf{k} \cdot (\mathbf{\Gamma}' \times \mathbf{\Gamma}''))}{\|\mathbf{\Gamma}'\| \|\mathbf{k} \times \mathbf{\Gamma}'\|^2}
 \end{aligned} \tag{5.20}$$

Similarly, for λ_2 ,

$$\begin{aligned}
 \lambda_2 &= \mathbf{e}_1 \cdot \mathbf{e}'_3 \\
 &= \mathbf{e}_1 \cdot (\mathbf{e}'_1 \times \mathbf{e}_2 + \mathbf{e}_1 \times \mathbf{e}'_2) \\
 &= \mathbf{e}_1 \cdot (\mathbf{e}'_1 \times \mathbf{e}_2) \\
 &= \frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|} \cdot \left(\left(\frac{\mathbf{\Gamma}''}{\|\mathbf{\Gamma}'\|} - \frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|^3} (\mathbf{\Gamma}' \cdot \mathbf{\Gamma}'') \right) \times \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \right) \\
 &= \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \cdot \left(\frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|} \times \left(\frac{\mathbf{\Gamma}''}{\|\mathbf{\Gamma}'\|} - \frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|^3} (\mathbf{\Gamma}' \cdot \mathbf{\Gamma}'') \right) \right) \\
 &= \frac{(\mathbf{k} \times \mathbf{\Gamma}') \cdot (\mathbf{\Gamma}' \times \mathbf{\Gamma}'')}{\|\mathbf{\Gamma}'\|^2 \|\mathbf{k} \times \mathbf{\Gamma}'\|}
 \end{aligned} \tag{5.21}$$

And for λ_3 ,

$$\begin{aligned}
 \lambda_3 &= \mathbf{e}_2 \cdot \mathbf{e}'_1 \\
 &= \frac{\mathbf{k} \times \mathbf{\Gamma}'}{\|\mathbf{k} \times \mathbf{\Gamma}'\|} \cdot \left(\frac{\mathbf{\Gamma}''}{\|\mathbf{\Gamma}'\|} - \frac{\mathbf{\Gamma}'}{\|\mathbf{\Gamma}'\|^3} (\mathbf{\Gamma}' \cdot \mathbf{\Gamma}'') \right) \\
 &= \frac{\mathbf{k} \cdot (\mathbf{\Gamma}' \times \mathbf{\Gamma}'')}{\|\mathbf{\Gamma}'\| \|\mathbf{k} \times \mathbf{\Gamma}'\|}
 \end{aligned} \tag{5.22}$$

It can also be verified that after all substitutions, the skew-symmetric condition at equation (5.17) holds. Eventually, for the differential form of the

GN frame shown in equation (5.17), the expression for $\boldsymbol{\lambda}$ is given by

$$\begin{cases} \lambda_1(s) = \mathbf{e}_3 \cdot \mathbf{e}'_2 = \frac{(\mathbf{k} \cdot \boldsymbol{\Gamma}')(\mathbf{k} \cdot (\widehat{\boldsymbol{\Gamma}'}\boldsymbol{\Gamma}''))}{\|\boldsymbol{\Gamma}'\| \|\widehat{\mathbf{k}}\boldsymbol{\Gamma}'\|^2} \\ \lambda_2(s) = \mathbf{e}_1 \cdot \mathbf{e}'_3 = \frac{(\widehat{\mathbf{k}}\boldsymbol{\Gamma}') \cdot (\widehat{\boldsymbol{\Gamma}'}\boldsymbol{\Gamma}'')}{\|\boldsymbol{\Gamma}'\|^2 \|\widehat{\mathbf{k}}\boldsymbol{\Gamma}'\|} \\ \lambda_3(s) = \mathbf{e}_2 \cdot \mathbf{e}'_1 = \frac{\mathbf{k} \cdot (\widehat{\boldsymbol{\Gamma}'}\boldsymbol{\Gamma}'')}{\|\boldsymbol{\Gamma}'\| \|\widehat{\mathbf{k}}\boldsymbol{\Gamma}'\|} \end{cases} \quad (5.23)$$

Fig. 5.4 provides a graphical illustration of the three curve-framing methods introduced in this chapter along a three-dimensional lemniscate. Fig. 5.4(a)-(c) depicts the frame axes along the lemniscate, while Fig. 5.4(d)-(f) compares the results for a simple flight corridor, defined as $\{(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \mid \mathbf{e}_1 \in [0, 1], -1 \leq \mathbf{e}_2 \leq 1, -1 \leq \mathbf{e}_3 \leq 1\}$, generated by the three curve-framing methods. In Fig. 5.4(a), the FS frame is stable around the two foci but undergoes rapid changes and distortions near the intersection regions, leading to twists in the flight corridor as shown in Fig. 5.4(d). The PT frames depicted in Fig. 5.4(b) are much smoother, however, the continuous torsion in the lemniscate causes $\mathbf{e}_2$ (green axis) and $\mathbf{e}_3$ (blue axis) to spin, preventing them from returning to their original orientation on a closed curve. To better illustrate this distortion, we follow the black line $\mathbf{r}(s) = (s, -1, -1)$ for $s \in [0, 1]$ along the corridor in Fig. 5.4(e). Initially, this line represents the top-left side of the corridor relative to the $\mathbf{e}_1$ direction, but it eventually shifts to the bottom-left side. In contrast, the GN frame shown in Fig. 5.4(c) remains closed and steady, with $\mathbf{e}_1$ always strictly planar. This is reflected in Fig. 5.4(f), where the black line consistently remains at the top-left side throughout the corridor relative to the $\mathbf{e}_1$ direction. This indicates that the flight corridor is always coherent with the flight direction, preserving the geometrical meaning of cross-track error along the corridor.

The numerical comparison is shown in Fig. 5.5. Fig. 5.5(a) compares the angle between $\mathbf{e}_2$ and the inertial ^Ixy plane, and Fig. 5.5(b) shows the absolute angle between $\mathbf{e}_3$ and $\mathbf{k}$ for the three curve framing methods in Fig. 5.4. In Fig. 5.5(a), the elevation angle plot for $\{FS\}$ shows two peaks

corresponding to the rapid changes near the two intersection regions. For $\{PT\}$, the angle does not return to zero, indicating a lack of closure. In contrast, the GN frame (GN) maintains a zero elevation angle, as it is always coplanar with the xy -plane. In Fig. 5.5(b), only the GN frame is able to maintain a low angle between $\mathbf{k}$ and $\mathbf{e}_3$, preserving the notion of pointing downward. Still, the angle is not exactly zero, as the adapted property is preserved.

The trade-off of $\{\mathbf{R}_{GN}\}$ is its inability to represent every curve in 3D space. Unlike restrictions imposed by curvature, this limitation arises from a constraint on the tangent direction. Specifically, the condition $\mathbf{k} \times \boldsymbol{\Gamma}' = 0$ implies that the curve must not have a completely vertical tangent at any value of $\boldsymbol{\Gamma}$. This constraint is a natural geometric consequence of the projected vector: it cannot possess a component parallel to $\mathbf{k}$ while remaining normal to a vertical tangent simultaneously. While this is a limitation, we consider it a reasonable trade-off for CSPQ systems when compared to constraints related to curvature and torsion. In most of the operations involving UAV systems, the takeoff phase and landing phase are handled separately with different navigation modules. In both cases we can redefine $\mathbf{k}$ (e.g., aligning it with the initial heading direction) for the trajectory, eliminating the singularity.

5.2.3 Spatial Reformulation

With the moving frame $\{\mathbf{R}_{GN}\}$ well-defined along the path, inspired by [Spedicato and Notarstefano, 2017], we opt to reformulate the equations of motion for position (3.13a) to embed the path characteristics into the system dynamics. Using $\mathbf{R}_{GN}$, the position of the quadcopter body can be expressed as the vector sum of the position vector of the path and the transverse vector from the path to the quadcopter as

$$\mathbf{p}_Q(t) = \boldsymbol{\Gamma}(s^*) + {}^I\mathbf{R}_{GN}(s^*)^{GN}\mathbf{w} \quad (5.24)$$

, where theoretically s^* can be any arbitrary value $\in [s_0, s_f]$ and ${}^{GN}\mathbf{w}$ is the displacement vector. The obvious choice for s^* is the one that can minimize

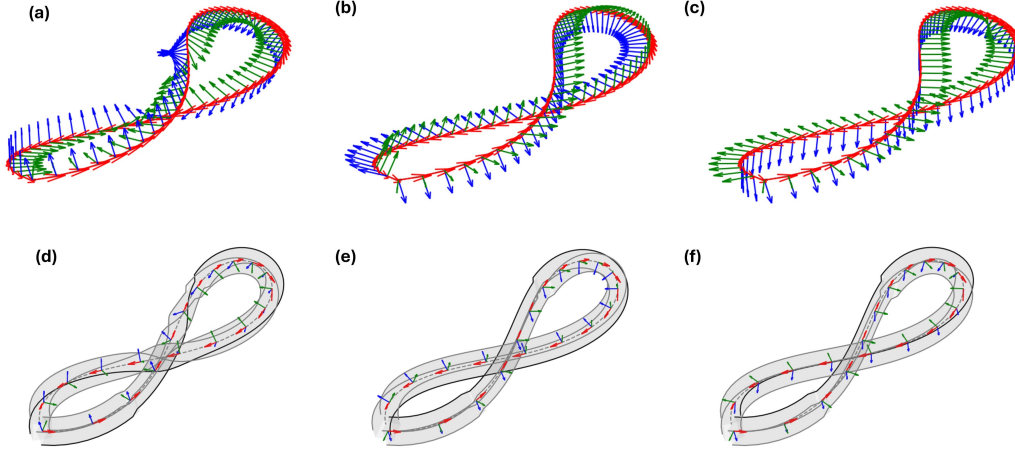


Figure 5.4: *Top*: comparison of the (a) FS, (b) PT and (c) GN frames on a lemniscate with height variation. *Bottom*: comparison of the sample flight corridor generated in (d) FS, (e) PT and (f) GN frames.

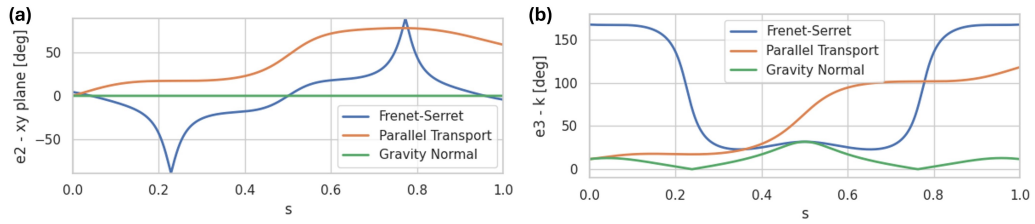


Figure 5.5: (a) The angle between $\mathbf{e}_1$ and xy plane in different frame construction methods. (b) The angle between $\mathbf{e}_1$ and the unit gravity vector $\mathbf{k}$.

${}^{GN}\mathbf{w}$, where

$$s^*(t) = \min_{s \in [s_0, s_f]} \|\mathbf{p}_Q(t) - \mathbf{\Gamma}(s)\|^2 \quad (5.25)$$

Another perspective is that minimizing ${}^{GN}\mathbf{w}$ is equivalent to taking the orthogonal projection of the quadcopter body onto $\mathbf{\Gamma}$. It is shown in Fig. 3.2 for a better illustration. Consequently, s^* can be viewed as the *progress* of the quadcopter body along the path. Under this configuration $\mathbf{w}$ is reduced to a two-dimensional vector, i.e. $\mathbf{w} = {}^{GN}[0 \ w_1 \ w_2]^T$ since the tangential component $\mathbf{e}_1$ is always zero.

Differentiating equation (5.24) w.r.t. t , we have

$$\mathbf{v}_Q(t) = \dot{s}^* \mathbf{\Gamma}'(s^*) + \dot{s}^* {}^I\mathbf{R}'_{GN}(s^*) {}^{GN}\mathbf{w}(t) + {}^I\mathbf{R}_{GN}(s^*) {}^{GN}\dot{\mathbf{w}}(t) \quad (5.26)$$

In equation (5.26), the matrices ${}^I\mathbf{R}_{GN}(s^*)$ and ${}^I\mathbf{R}'_{GN}(s^*)$ are determined using equation (5.15) and (5.17). For $\mathbf{\Gamma}'(s^*)$, recalling equation (5.15a) and since $\mathbf{v}_Q(t)$ is expressed in $\{I\}$, we have

$$\mathbf{\Gamma}'(s^*) = {}^I\mathbf{R}_{GN}(s^*) \nu(s^*) {}^{GN} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \quad (5.27)$$

Thus, multiplying both sides by ${}^I\mathbf{R}_{GN}^T(s^*)$ and rearranging the terms, we obtain

$$\dot{s}^*(t) = \frac{\mathbf{e}_1(s) \cdot \mathbf{v}_Q(t)}{\nu(s) - \lambda_3(s)w_1(t) + \lambda_2(s)w_2(t)} \quad (5.28a)$$

$$\dot{w}_1(t) = \mathbf{e}_2(s) \cdot \mathbf{v}_Q(t) + \dot{s}(t)\lambda_1(s)w_2(t) \quad (5.28b)$$

$$\dot{w}_2(t) = \mathbf{e}_3(s) \cdot \mathbf{v}_Q(t) - \dot{s}(t)\lambda_1(s)w_1(t) \quad (5.28c)$$

(Note: equation (5.28) is indeed a universal setup for any curve framing method. Undergoing the same approach, it can be shown that it applies to the FS frame when the angular velocity vector of $\{FS\}$ is adopted instead, i.e. $\boldsymbol{\lambda} = \nu \begin{bmatrix} -\kappa & 0 & -\tau \end{bmatrix}^T$. Similarly, for the PT frame, the choice of $\boldsymbol{\lambda}$ would be $\nu \begin{bmatrix} 0 & -\kappa c_\theta & -\kappa s_\theta \end{bmatrix}^T$.)

Together, the motion of the quadcopter body can be re-parameterized by spatial coordinates, s^* , w_1 , w_2 , as well as their derivatives. Similar to other

literature, s is also referred to as the *progress* variable, as it directly reflects the progress on the path parameterized by s itself. w_1 and w_2 are referred to as the transverse coordinates of the path.

Substituting s^* back to s for simplicity, we define the re-parameterization of the position of the quadcopter body as $\boldsymbol{\xi}_Q$, where

$$\boldsymbol{\xi}_Q = \begin{bmatrix} s & w_1 & w_2 \end{bmatrix}^T \quad (5.29)$$

and the new state vector and state-space ODE for CSPQ are

$$\mathbf{x}_\xi = \begin{bmatrix} \boldsymbol{\xi}_Q^T & \mathbf{v}_Q^T & \mathbf{q}^T & \boldsymbol{\omega}^T & \boldsymbol{\Phi}^T \end{bmatrix}^T \quad (5.30)$$

$$\dot{\mathbf{x}}_\xi = f(\mathbf{x}_\xi, \mathbf{u}) \quad (5.31)$$

Equation (5.31) comes from equation (3.13) in that $\mathbf{p}_Q$ in equation (3.13a) is converted to $\boldsymbol{\xi}_Q$, and their derivatives are given by equation (5.28).

The spatial reformulation of the system dynamics introduces additional nonlinearity to the system but is appealing for two main reasons. Firstly, the CSPQ system is now decoupled into tangential motion and two transverse motions along the original path, with the notion of progress formally defined. Secondly, geometric constraints, such as obstacle avoidance and tracking accuracy, which are inherently nonlinear, are transformed into simpler linear or norm-bound constraints dependent on s only. This transformation is beneficial for subsequent trajectory generation and tracking. Although it is not a complete spatial reformulation similar to [Spedicato and Notarstefano, 2017, Arrizabalaga and Ryll, 2023, Verschueren et al., 2016] where the system dynamics evolve with respect to s instead of the time t , it removes the singularity associated with $\dot{s} > 0$ [Kloeser et al., 2020].

5.3 Flight Corridor Formulation

In this section, based on the GN frame reformulation, we propose a method to construct a feasible flight corridor, which is treated as a free space for the quadcopter to optimize its trajectory while ensuring collision-free movement.

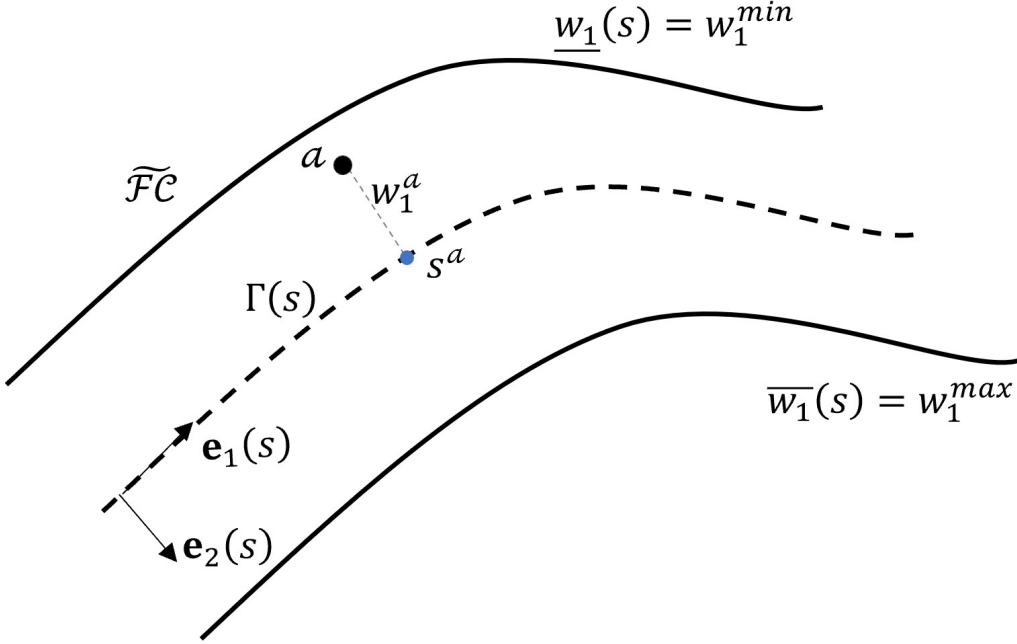


Figure 5.6: illustration of boundary update for a point

One major advantage of applying spatial reformulation to describe geometric constraints is that obstacles and path boundary constraints can be implemented as simple bounds on the state vector, in particular the bounds on transverse coordinates. It eliminates the need for complex non-linear inequalities. In addition, in conventional formulation, constraints are expressed in full 3D coordinates. Much literature comes up with convex approximations of the obstacle-free space, such as using voxel-grid-based decomposition [Chen et al., 2016], linear 3D convex polyhedrons [Liu et al., 2017], or by initializing a narrow safety corridor and incrementally expanding across all dimensions [Chen et al., 2023]. Meanwhile, in the spatial reformulation framework, the constraints are described in the transverse $\mathbf{e}_2\mathbf{e}_3$ plane, i.e. the obstacle dimension is reduced to 2D. Also, this method is suited for arbitrary paths in space.

5.3.1 Rationale

In this chapter, it is assumed that map information is known beforehand. For simplicity, all obstacles in the map are static and have been convexified, especially as convex polygonal prisms. This reflects an urban environment where the dominant obstacles are buildings and infrastructure.

Recall that in $\{GN\}$ frame, $\mathbf{e}_1$ stands for the unit tangent vector, and $\mathbf{e}_2$ and $\mathbf{e}_3$ are the osculating plane axis vectors representing the lateral and the downward unit vectors. For illustration purposes, as depicted in Fig. 5.6, a 2D flight corridor corresponding to the $\mathbf{e}_1\mathbf{e}_2$ plane is considered as an illustrative example. Denote $w_1^{\min}$ and $w_1^{\max}$ as the maximum safety distances along the negative and positive directions of $\mathbf{e}_2$, respectively. The path boundary functions $\underline{w}_1(s) = w_1^{\min}$ and $\bar{w}_1(s) = w_1^{\max}$ for $s \in [0, L]$ are proposed to generate the initial flight corridor $\tilde{FC}$. These are shown as two black border curves parallel to the gray dashed line, which represents the reference curve $\Gamma(s)$.

Consider a point ${}^I\mathbf{a}$ in the flight corridor. To express it as ${}^{GN}\mathbf{a}$ with reference to the moving frame $\{GN\}$ along $\Gamma(s)$, we compute its progress variable $s^{\mathbf{a}}$ along the curve, i.e., identify the point on $\Gamma(s)$ that is at the minimum distance from $\mathbf{a}$, as

$$s^{\mathbf{a}} = \arg \min_{s \in \mathbb{R}_0^+} \|\mathbf{a} - \Gamma(s)\|^2 \quad (5.32)$$

Using $s^{\mathbf{a}}$ in equation (5.32) and the rotation matrix ${}^I\mathbf{R}_{GN}$, the distance vector $\mathbf{a} - \Gamma(s^{\mathbf{a}})$ is remapped with frame components in $\{GN\}$ attached to the point $\Gamma(s^{\mathbf{a}})$, i.e., ${}^PT\mathbf{R}_I({}^I\mathbf{a} - \Gamma(s^{\mathbf{a}}))$. In Fig. 5.6, as only the $\mathbf{e}_1\mathbf{e}_2$ plane is considered, w_2 is known to be zero. For completeness, in the full case, w_1 and w_2 are computed as

$$w_1^{\mathbf{a}} = \mathbf{e}_2^T(s^{\mathbf{a}})({}^I\mathbf{a} - \Gamma(s^{\mathbf{a}})) \quad (5.33a)$$

$$w_2^{\mathbf{a}} = \mathbf{e}_3^T(s^{\mathbf{a}})({}^I\mathbf{a} - \Gamma(s^{\mathbf{a}})) \quad (5.33b)$$

According to Fig. 5.6, ${}^I\mathbf{a}$ is on the negative side of $\mathbf{e}_2$, and the magnitude of w_1^a is smaller than $w_1^{\min}$. Hence, the function $\underline{w}_1(s)$ at $s = s^a$ is updated as

$$\underline{w}_1(s^a) = w_1^a, \quad (5.34)$$

since $w_1^a < 0$.

As path boundary updates for one point can be computed, it is intuitive to update the path boundary with regard to a convex obstacle as follows: for each obstacle close to $\tilde{\text{FC}}$, discretize its boundary surface into evenly distributed points, and for each point on the surface, update path boundary functions $\underline{w}_1(s)$ and $\overline{w}_1(s)$ using equations (5.32) to (5.34). Yet, this approach has three disadvantages:

1. Although obstacles on the map are assumed to be convex, their convexity is not guaranteed when converted into the moving frame. This fact can be illustrated by the example described in [Reiter et al., 2022]: Consider a straight line, which is convex, and a circular path $\gamma(s)$ intersecting the line at two points $\gamma(a_1)$ and $\gamma(a_2)$. The normal component of the transformed coordinates is zero at both points. Yet, for any $\gamma(a_3) \in (\gamma(a_1), \gamma(a_2))$, the normal component of the transformed line is not zero, and thus the transformed set (straight line segment between $\gamma(a_1)$ and $\gamma(a_2)$) is not convex.
2. If the discretization of the boundary surface is not fine enough, it may not cover sharp edges on the obstacle present within the interval of surface points. The path boundary function cannot ensure safety in that region.
3. The number of sample points grows cubically in the case of a full 3D flight corridor.

To overcome the limitations associated with non-convexity and discretization resolution, we propose a more cautious approach for flight corridor generation without sacrificing efficiency. In short, we generate a safe flight corridor

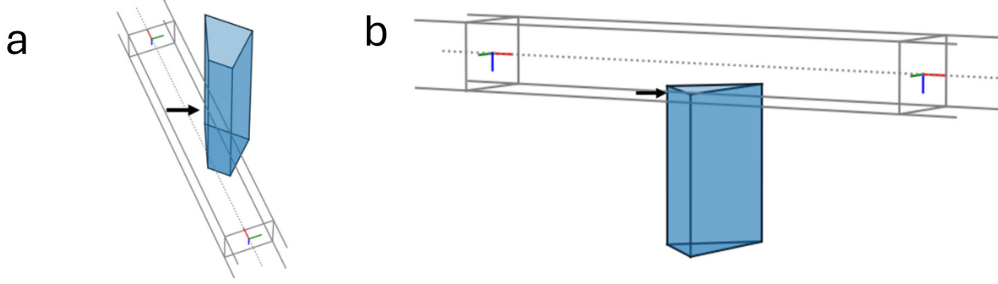


Figure 5.7: Illustration of the dominating side of the obstacle with respect to the flight corridor. (a) the dominant side at $\mathbf{e}_2$ (b) the dominant side at $\mathbf{e}_3$

by only considering a projected 2D case, followed by an adjustment algorithm to address any infringements in 3D. The details of the method are discussed as follows.

5.3.2 Determining the Dominant Side

To start off, generate the initial flight corridor $\tilde{FC}$, which consists of four boundary functions using the predefined maximum safety distances, i.e.,

$$\underline{w}_1(s) = w_1^{\min}, \quad \overline{w}_1(s) = w_1^{\max}, \quad \underline{w}_2(s) = w_2^{\min}, \quad \overline{w}_2(s) = w_2^{\max},$$

for $s \in [0, L]$, where $[0, L]$ can be any path parameterization.

Next, identify the set of active obstacles $\mathbf{O}$ that would potentially infringe upon the flight corridor from the obstacle set $\tilde{\mathbf{O}}$ on the map. Usually, revealing $\mathbf{O}$ efficiently relies on some properties that arise from the nature of the initial reference curve. For example, if the initial path is a set of cubic B-splines or a set of Bézier curves defined in the inertial frame $\{I\}$, we can make use of its convex hull property so that the curve is always within the convex hull of its control points. As a result, only obstacles inside the convex hull are considered.

For each obstacle $\mathbf{O}_i$ in $\mathbf{O}$, the dominant side of the obstacle with respect to the $\{GN\}$ frame is first determined, i.e., whether the obstacle restricts the flight corridor more on $\mathbf{e}_2$ or $\mathbf{e}_3$ direction. This is done to temporarily reduce the problem dimension to 2D, enabling a subsequent geometrical approach described in Section 5.3.4 to trim the corridor.

This is performed by a simple check: consider an arbitrary vertex v_i at the upper surface of the prism and transform it into the $\{GN\}$ frame. Given that $\mathbf{e}_3$ is pointing towards gravity, if the w_2 value of v_i is more negative than $w_2^{\min}$, it indicates that the height of the obstacle is higher than the flight corridor. This means the obstacle mainly appears in the $\mathbf{e}_2$ direction, and we therefore define that $\mathbf{e}_2$ is the dominant side for obstacle O_i . The idea is illustrated in Figure 5.7(a).

Notice that the red axis indicates the tangent vector $\mathbf{e}_1$, the green axis indicates $\mathbf{e}_2$, and the blue axis indicates $\mathbf{e}_3$. This setting is adopted for the latter part of the chapter. The rectangular tube is the flight corridor, and we can expect that limiting the corridor on the $\mathbf{e}_3$ side (the area where the arrow is pointing) prevents collision with the obstacle. Otherwise, the obstacle is intruding on both $\mathbf{e}_2$ and $\mathbf{e}_3$, which is the region indicated by the arrow as shown in Figure 5.7(b).

We call $\mathbf{e}_3$ the dominating side in this case, as we can observe that limiting the corridor in the $\mathbf{e}_3$ direction guarantees safety, but this is not the case for $\mathbf{e}_2$. Inevitably, any flight corridor generated by considering its 2D projection only would contain some remaining infringements, and these are subsequently addressed in the extension phase.

5.3.3 Line Update

Before explaining the workflow of the corridor generation, a basic operation is illustrated in Figure 5.8(a), which we name *line update*. Consider the $\mathbf{e}_1\mathbf{e}_2$ plane as an example. A line segment l is composed of two points $\mathbf{p}_1$ and $\mathbf{p}_2$, and some output variables are determined by the following three tests:

1. Transformation of Endpoints:

The two endpoints of the line segment are transformed into the $\{GN\}$ frame using the step proposed in equations (5.32) and (5.33). If the transverse coordinate on the dominant side falls within the range of the initial corridor, its s -coordinate is included in the output set. As shown in Figure 5.8(a), both endpoints of the black line segment l_1 lie within the corridor shown as a gray line, and the green dots represent

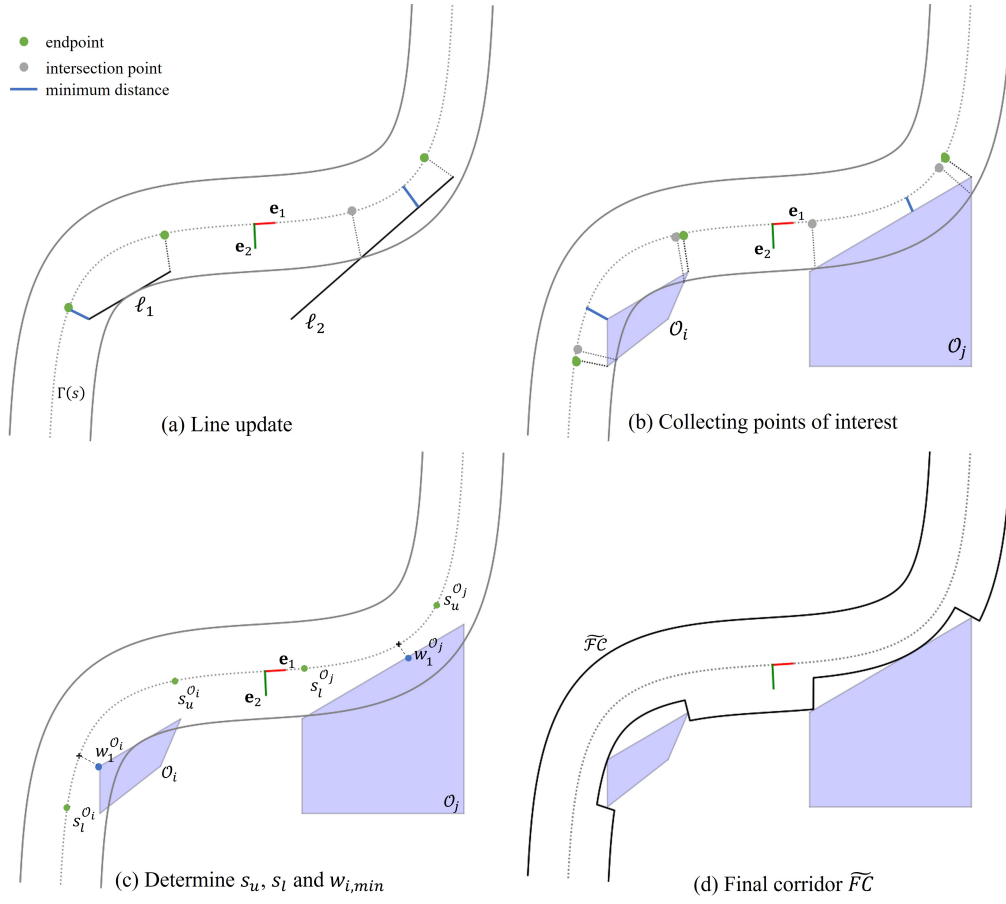


Figure 5.8: Illustration of the safety corridor generation process

the resulting output. For l_2 , an endpoint lies beyond the boundary and is therefore dropped.

2. Intersection with Flight Corridor Boundary:

If one endpoint lies within the boundary and the other exceeds the boundary, there must be a point intersecting the line segment positioned at the boundary of the flight corridor. The s -coordinate of the intersection point is included in the output set. For a line segment l with two endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$, the line segment can be parameterized by λ as:

$$\mathbf{p}(\lambda) = \lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2, \quad \lambda \in [0, 1].$$

In the case of l_2 , based on the transformation of its endpoints, we can determine that l_2 is on the positive side of $\mathbf{e}_2$. Knowing that the coordinate in the $\mathbf{e}_2$ direction is the boundary value $w_1^{\max}$, the boundary point can be determined by:

$$(s_b, \lambda_b) = \underset{s \in \mathbb{R}_0^+, \lambda \in [0, 1]}{\operatorname{argmin}} \left\| \mathbf{p}(\lambda) - {}^I \mathbf{R}_{GN} \begin{bmatrix} s \\ w_1^{\max} \\ 0 \end{bmatrix} \right\|^2, \quad (5.35)$$

where the s -coordinate of the intersection point is shown as the gray dots in Figure 5.8(a). Despite being an optimization process, a high-quality solution can be achieved with only 1 or 2 iterations using a standard optimizer. All intersection points must be considered if there is more than one intersection.

3. Minimum Distance to Reference Curve:

The minimum distance w_1^l between the line segment l and the reference curve $\Gamma(s)$ is determined by:

$$(s_{\min}, \lambda_{\min}) = \underset{s \in \mathbb{R}_0^+, \lambda \in [0, 1]}{\operatorname{argmin}} \left\| \mathbf{p}(\lambda) - {}^I \mathbf{R}_{PT} \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix} \right\|^2, \quad (5.36)$$

where the residual is the minimum distance value $\|w_1^l\|$. $s_{\min}$ and $\lambda_{\min}$ indicate the location of the point on the curve and the point on the line segment where the distance between them is minimized. In Fig. 5.8(a), since l_2 is on the positive side of $\mathbf{e}_2$, its sign value is also positive. These are represented by the blue lines in l_1 and l_2 .

The intuition of the line update is to distinguish the affected segment on the reference curve given the obstacles. The s -coordinates of both ends provide a reliable estimate of the entry and exit points of the line segment. Meanwhile, w_1^l indicates the maximum reduction in the flight corridor. By limiting the flight corridor within the specified range of the s -coordinates, the obstacle can be avoided easily.

5.3.4 Flight Corridor Generation on $\mathbf{e}_2$ Dominant Side

The aforementioned *line update* operation is applied to every edge of the polygon, as shown in Figure 5.8(b). $\mathbf{e}_2$ is the dominant side, and the two obstacles $\mathbf{O}_i$ and $\mathbf{O}_j$ are projected onto the $\mathbf{e}_1\mathbf{e}_2$ plane.

For each obstacle, the line update operation described in Section 5.3.3 is performed on every edge. Green dots represent s -coordinates of endpoints within the initial flight corridor, and gray dots represent any intersection points. The blue line represents the minimum distance $w_1^{(\mathbf{O}_i)}$ with the lowest magnitude across all edges, which is equivalent to the closest distance between the obstacle and the reference curve.

Given the set of s -coordinates from obstacle $\mathbf{O}_i$, select the minimum and maximum values from the set as $s_l^{(\mathbf{O}_i)}$ and $s_u^{(\mathbf{O}_i)}$, where l means lower bound and u means upper bound. As a result, the safe flight corridor across $\mathbf{O}_i$ can be updated as:

$$\overline{w}_1(s) = w_1^{(\mathbf{O}_i)}, \quad s_l^{(\mathbf{O}_i)} \leq s \leq s_u^{(\mathbf{O}_i)}, \quad (5.37)$$

for $\mathbf{e}_2$, where other parts remain unchanged.

By repeating the update process for each obstacle using Equation (5.37), the safe flight corridor $\tilde{\text{FC}}$ is constructed, as shown in Figure 5.8(d).

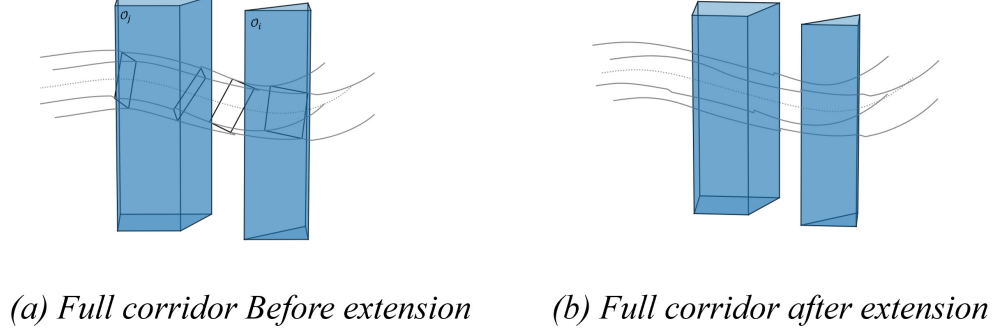


Figure 5.9: Illustration of a sample flight corridor in full 3D scenario with obstacles on $\mathbf{e}_2$ side

5.3.5 Flight Corridor Extension on $\mathbf{e}_2$ Dominant Side

The corridor in Section 5.3.4 prevents obstacles in 2D, but it cannot guarantee safety when extended to a full 3D corridor. For further explanation, consider the full scenario in Figure 5.9(a) and the corridor after extension in Figure 5.9(b), where its projection onto the $\mathbf{e}_1\mathbf{e}_2$ plane is the same as in Figure 5.8, except $\mathbf{O}_i$ is simplified to a triangular prism. Notice that the reference curve changes altitude when it comes across both obstacles.

As an illustrative example, we examine $\mathbf{O}_j$. Figure 5.10(a) shows the side view of $\mathbf{O}_j$, and Figure 5.10(b) shows the top view. There is an overlap between the obstacle and the corridor caused by the change in height, as indicated by the arrow, which is clearer in the top view. To address this issue, we propose a method to extend the safety zone to compensate for the overlap.

To determine if the flight corridor is intruding on the obstacle, at the two ends $s = s_l^{\mathbf{O}_j}$ and $s = s_u^{\mathbf{O}_j}$, we check whether any points among its four corner points:

$$[w_1^{\min}, w_2^{\min}], [w_1^{\max}, w_2^{\min}], [w_1^{\min}, w_2^{\max}], \text{ and } [w_1^{\max}, w_2^{\max}]$$

are located inside $\mathbf{O}_j$. In Figure 5.10(a), the corner points at both ends are connected to form a rectangle for better illustration. After checking, at the

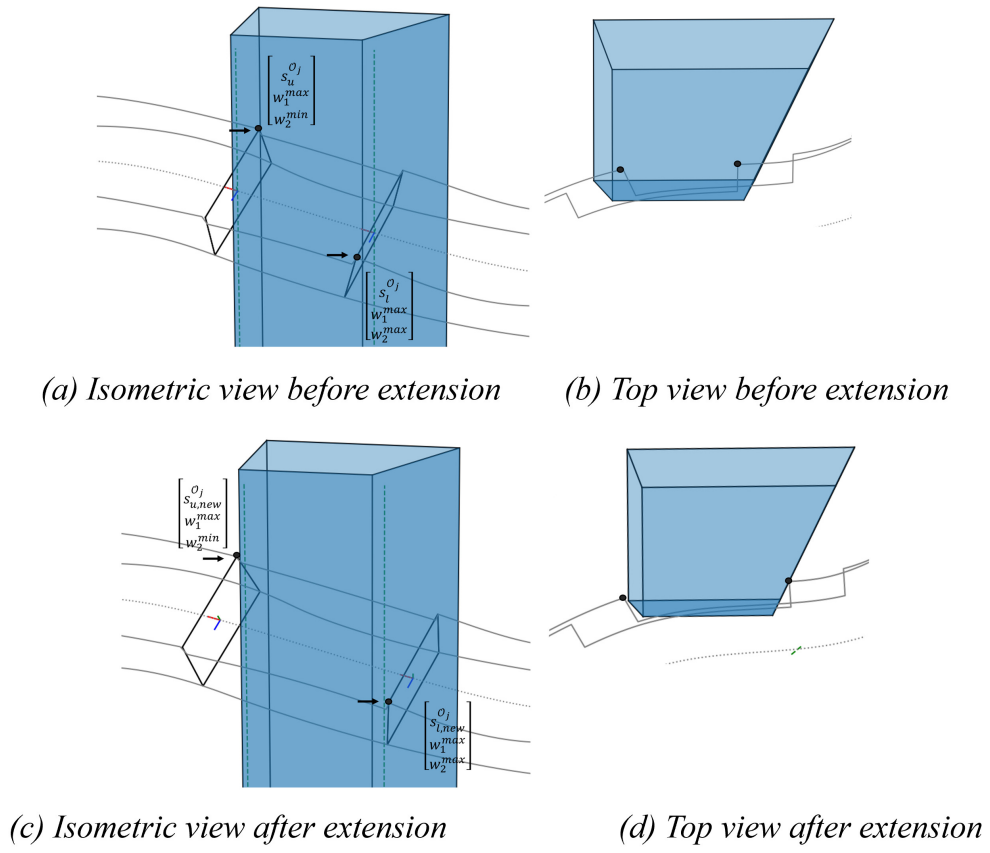


Figure 5.10: Close-up view on obstacle O_j during extension

lower end, the corner point $\begin{bmatrix} s_l^{O_j} & w_1^{\max} & w_2^{\max} \end{bmatrix}^T$ is intruding on O_j , and at the upper end, the corner point $\begin{bmatrix} s_u^{O_j} & w_1^{\max} & w_2^{\min} \end{bmatrix}^T$ is intruding on O_j .

To perform the safety zone extension for the obstacle O_j , we first set up two *anchor pillars* at points $s_l^{O_j}$ and $s_u^{O_j}$ on the curve. This is done by transforming both points back to the inertial frame $\{I\}$ and setting up two vertical straight lines located at $s_l^{O_j}$ and $s_u^{O_j}$, respectively. These are illustrated as the vertical dashed lines in Figure 5.10(a).

Consider $s_l^{O_j}$. Its anchor pillar is located at the position: $\begin{bmatrix} {}^I\mathbf{p}_x & {}^I\mathbf{p}_y \end{bmatrix}$, where ${}^I\mathbf{p}_x$ and ${}^I\mathbf{p}_y$ are the x and y coordinates of $\Gamma(s_l^{(O_j)})$ in the inertial frame. Regarding the intrusion point at the lower end $\begin{bmatrix} s_l^{(O_j)} & w_1^{\max} & w_2^{\max} \end{bmatrix}^T$, extending the safety zone means $s_l^{O_j}$ should be adjusted so that the intrusion point no longer lies within the obstacle. To do so, the new $s_{l,\text{new}}^{O_j}$ is found by minimizing the distance between $s_{l,\text{new}}^{(O_j)}$ and the anchor pillar:

$$(s_{l,\text{new}}^{O_j}, h) = \underset{s \in \mathbb{R}_0^+, h \in [0, H]}{\operatorname{argmin}} \left\| \begin{bmatrix} {}^I\mathbf{p}_x \\ {}^I\mathbf{p}_y \\ h \end{bmatrix} - {}^I\mathbf{R}_{GN} \begin{bmatrix} s \\ w_1^{\max} \\ w_2^{\max} \end{bmatrix} \right\|^2, \quad (5.38)$$

where h is the height of the pillar at minimum distance, and H is the height of the obstacle O_j .

Repeat this checking and extension process for every corner point until all four corner points stay outside O_j . Figure 5.10(c) and Figure 5.10(d) show the adjustments after the extension process, where the black dots at both sides are the adjusted corner points that no longer cause any collision. Figure 5.9(b) shows a sample of the final flight corridor after extension.

5.3.6 Flight Corridor Extension on $\mathbf{e}_3$ Dominant Side

Similar to the $\mathbf{e}_2$ dominant side, for $\mathbf{e}_3$ to be the dominant side, the obstacles are projected to the $\mathbf{e}_1\mathbf{e}_3$ plane, and the same line update described in Section 5.3.3 is also applied. In addition, other than the line update for all edges, there is an additional plane update for the top surface of the obstacle. This is because the minimum distance between the "roof" and the reference curve

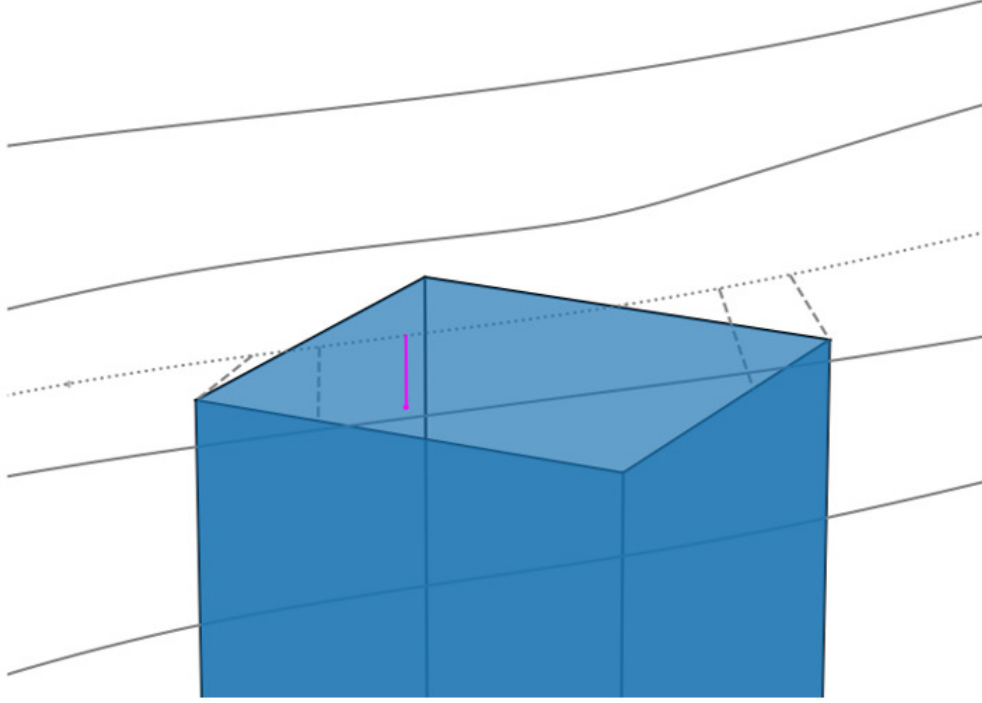


Figure 5.11: Minimum distance between the top surface and the reference curve (shown as magenta line)

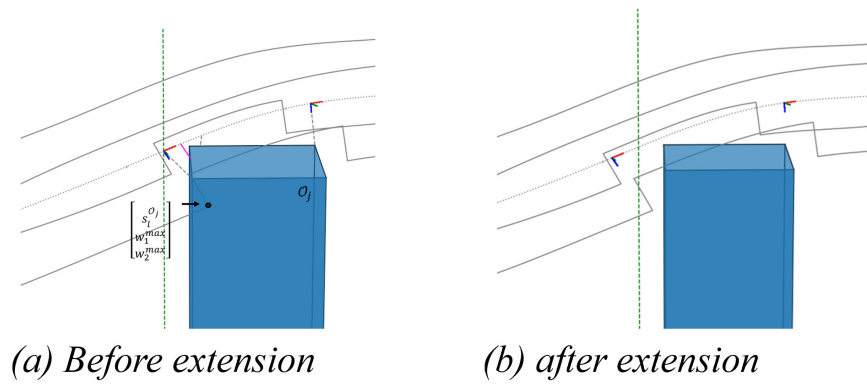


Figure 5.12: Close-up view on obstacle O_k during extension on $\mathbf{e}_2$ dominant side

should also be considered.

Consider the shape of the top surface as a rectangle, given 4 boundary points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ of a rectangle $\mathbf{r}$ in clockwise order. The rectangular plane is parameterized as

$$\mathbf{r}(u, v) = (1 - u)((1 - v)\mathbf{p}_1 + v\mathbf{p}_4) + u((1 - v)\mathbf{p}_2 + v\mathbf{p}_3), \quad u, v \in [0, 1] \quad (5.39)$$

The minimum distance $w_2^{\mathbf{r}}$ between a plane $\mathbf{r}$ and the reference curve $\Gamma(s)$ is determined by

$$(s_{\min}^{\mathbf{r}}, u_{\min}, v_{\min}) = \underset{s \in \mathbb{R}_0^+, u, v \in [0, 1]}{\operatorname{argmin}} \left\| \mathbf{r}(u, v) - {}^I \mathbf{R}_{PT} \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix} \right\|^2, \quad (5.40)$$

where the residual is the minimum distance value. It is illustrated in Fig. 5.11. The grey dashed lines represent the minimum distances from the line update, whereas the magenta line represents the minimum distance from the plane update, which is indeed the closest in this case.

For any arbitrary convex polygon shape of the top surface, we adopt a conservative approach: the polygon is to be covered by a rectangle, which is defined by the minimum and maximum x and y coordinates of the polygon's vertices.

The safety zone extension method at $\mathbf{e}_3$ side is the same as in $\mathbf{e}_2$ side. With reference to Figure 5.12(a), $\begin{bmatrix} s_l^{O_j} & w_1^{\max} & w_2^{\max} \end{bmatrix}^T$ is intruding the obstacle, and the same extension method described in Section 5.3.5 is applied. The *anchor pillar* at $s_l^{O_j}$ is shown as the green dotted line. Figure 5.12(b) shows the resulting flight corridor after extension.

5.4 Concluding Remarks

This chapter introduced a novel spatial reformulation framework based on the Gravity-Normal (GN) frame for time-optimal motion planning and flight cor-

ridor generation. Unlike conventional methods such as the Frenet-Serret (FS) and Parallel Transport (PT) frames, the GN frame provides a singularity-free, smooth, and analytically defined curve-framing method, addressing issues like twisting and undefined behavior at inflection points.

The reformulation decouples the system dynamics into tangential and transverse components, simplifying obstacle constraints into spatial bounds. This approach enhances computational efficiency and makes the optimization problem more intuitive. Using this framework, we developed a robust method for flight corridor generation by iteratively updating boundaries based on obstacle projections and addressing infringements in three dimensions. The result is a collision-free, coherent corridor that respects both lateral and vertical constraints.

Chapter 6

Time-Optimal Planning Framework

This chapter proposes a unified hierarchical framework for time-optimal motion planning and trajectory tracking of CSPQ systems, built upon a novel spatial reformulation method leveraging the Gravity-Normal (GN) frame. Both the quadcopter model and the CSPQ model are used for illustration.

Utilizing the system model in chapter 3, the global path in chapter 4 and the reformulation approaches described in chapter 5, the trajectory planning and optimization method is developed in terms of path-parametric coordinates with clear physical interpretation and no singularities, thereby simplifying the representation of spatial constraints. The hierarchical framework comprises an offline module for global time-optimal trajectory generation using the direct multiple shooting method and an online module for real-time trajectory tracking through nonlinear model predictive control (NMPC). Simulation experiments demonstrated the effectiveness of the proposed framework in achieving time-optimal motion while suppressing payload oscillations within acceptable limits.

6.1 Introduction

Our objective is to achieve fast and safe payload transport while suppressing payload vibrations within acceptable limits, utilizing a reference global path provided by any mature global planner [Zhou et al., 2025]. This problem can be better framed as an optimal control problem (OCP) with constraints on the swinging load within a safe flight corridor, as OCP outperforms the aforementioned methods by explicitly handling state and input constraints through its inherent formulation [Geisert and Mansard, 2016]. While OCP was traditionally an offline optimization problem, the mathematical framework for online trajectory generation using OCP (through *Model Predictive Control*) has significantly evolved [Mayne et al., 2000], with various applications already implemented in quadcopter systems [Wang et al., 2024, Sarvaiya

et al., 2025].

We developed a unified hierarchical framework for time-optimal motion planning framework based on the quadcopter and CSPQ system model described in chapter 3, the global planner described in chapter 4, and the GN frame spatial reformulation in chapter 5. This approach allows for a natural formulation of the time-optimal OCP, facilitating intuitive flight corridor generation, incorporating actuation constraints, and visualizing path progress. First, utilizing the spatial reformulation with respect to a reference curve defined in the inertial frame, we developed a natural formulation of the OCP. Second, the OCP is solved using a two-layer approach: the first layer solves the time-optimal trajectory generation problem offline using the direct multiple shooting method, while the second layer formulates a nonlinear model predictive control (NMPC) framework to track the reference trajectory online.

The rest of the chapter is organized as follows: Section 6.2 introduces the proposed UAV trajectory optimization framework, utilizing the system dynamics in chapter 3, the global planner in chapter 4, the spatial reformulation method in chapter 5, and the formulation of the optimization problem using the optimal control problem approach. Section 6.3 presents numerical results and simulations. Section 6.4 is the concluding remarks.

6.2 Hierarchical Framework

The overall framework is illustrated in Fig. 6.1 comprising the offline and online execution modules. In the offline module, based on the input reference path $\mathbf{\Gamma}$, the spatial reformulation sub-module calculates the necessary parameters, including ${}^I\mathbf{R}_{GN}(s)$ and $\boldsymbol{\lambda}(s)$, as defined in Section 5.2.3 for subsequent computations. The global trajectory optimization sub-module generates the time-optimal trajectory $\{\mathbf{x}_{\xi,ref}\}$ using the direct multiple shooting method, as described later in Section 6.2.1.1. During the optimization process, minimum payload swinging can be enforced by setting state bounds on the unit cable direction $\mathbf{q}$. It is worth noting that if real-time performance is required, the online module can utilize the unoptimized initial trajectory

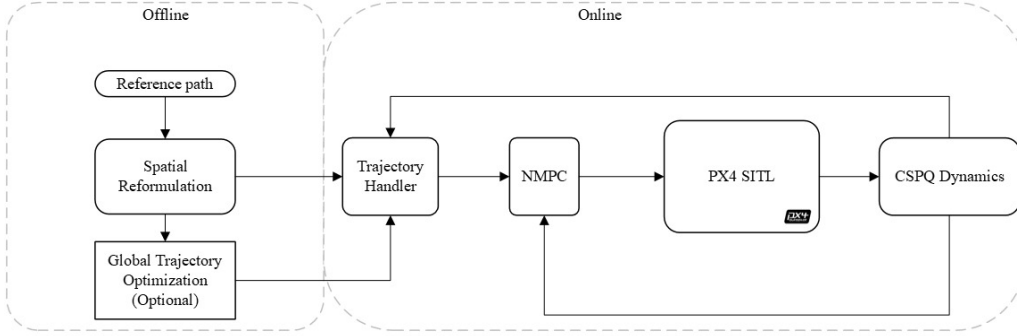


Figure 6.1: The overall framework for the time-optimal motion planning problem

instead of online trajectory tracking. It yields near-optimal motion, but the offline planning time is significantly reduced.

To generate the initial trajectory, unlike the approach in Section 4.2, we use the augmented Lazy Theta* planner combined with the original TEB (Timed-Elastic Band) planner. The TEB planner employs its original cost function, prioritizing smoothness while maximizing the distance from obstacles.

In the online module, based on the prediction horizon set in the NMPC sub-module, the trajectory handler sub-module calculates the target CSPQ states $\mathbf{x}_{ref,k}$ required by the NMPC. If the global trajectory is available, the full states are provided; otherwise, only the target progress state s serves as the input. The NMPC sub-module then solves the optimal inputs for online trajectory tracking in a real-time fashion, which is described in section 6.2.2. Subsequently, the computed total thrust and body angular rate commands are passed to the PX4 Software-in-the-Loop (PX4-SITL) package [Meier et al., 2015] and hence to the CSPQ system.

6.2.1 Time-Optimal Trajectory Generation

In this section, we describe the formulation of the time-optimal trajectory generation problem in an NMPC-based optimal control problem (OCP) approach. In Section 5.2.3, the spatial reformulation method can reshape the system dynamics into a reference path-dependent one. Using the flight corridor approach described in Section 5.3, obstacle constraints can be represented

as simple bounds on the state vector, especially transverse coordinates along the path. The time-optimal OCP in continuous space can then be defined as:

6.2.1.1 Direct Multiple Shooting for Global Time-Optimal Trajectory

To proceed directly, we adopt the time-optimal OCP problem formulation in [Betts, 2010] as

$$\min_{\mathbf{x}_\xi(\cdot), \mathbf{u}(\cdot), T} \quad s.t. \quad (6.1a)$$

$$\mathbf{x}_\xi(0) = \mathbf{x}_0, \quad \mathbf{x}_\xi(T) = \mathbf{x}_T, \quad (6.1b)$$

$$\dot{\mathbf{x}}_\xi(t) = f(\mathbf{x}_{\xi,k}(t), \mathbf{u}_k(t)), \quad t \in [0, T] \quad (6.1c)$$

$$\underline{\mathbf{x}}_{\xi,k}(t) \leq \mathbf{x}_{\xi,k}(t) \leq \overline{\mathbf{x}}_{\xi,k}(t), \quad t \in [0, T] \quad (6.1d)$$

$$\underline{\mathbf{u}}_k(t) \leq \mathbf{u}_k(t) \leq \overline{\mathbf{u}}_k(t), \quad t \in [0, T] \quad (6.1e)$$

$$H(\mathbf{x}_{\xi,k}(t)) < 0 \quad t \in [0, T] \quad (6.1f)$$

where the cost function is simply the flight time T and the decision variables are the state and the control trajectory $\mathbf{x}_\xi(t)$, $\mathbf{u}(t)$ as well as T . Equation (6.1b) represents the initial and desired terminal state constraints. Equation (6.1c) is the dynamic constraint, which is equivalent to equation (5.31) for CSPQ system or equation (3.4) for the quadcopter. Equation (6.1d) are the state bounds including limitations on $\mathbf{w}$, $\mathbf{v}$, $\mathbf{q}$, $\boldsymbol{\omega}$ and $\boldsymbol{\Phi}$. Equation (6.1e) denotes the control bounds and (6.1f) denotes the stage constraints.

To solve this infinite-dimension OCP, we use the multiple shooting method. The continuous-time trajectories $\mathbf{x}_\xi(t)$ and $\mathbf{u}(t)$ in equation (6.1a) are discretized with a fixed number of N shooting intervals. Since T is not a known priori, we introduce a time scaler $\tau := Nt/T$ [Rosmann et al., 2017] such that the state-space ODE in equation (6.1c) as

$$\frac{\partial \mathbf{x}_\xi}{\partial \tau} = f(\mathbf{x}_{\xi,k}(\tau), \mathbf{u}_k(\tau)) \frac{T}{N}, \quad \tau \in [0, N] \quad (6.2)$$

This time-scaling method reformulates the problem such that the time hori-

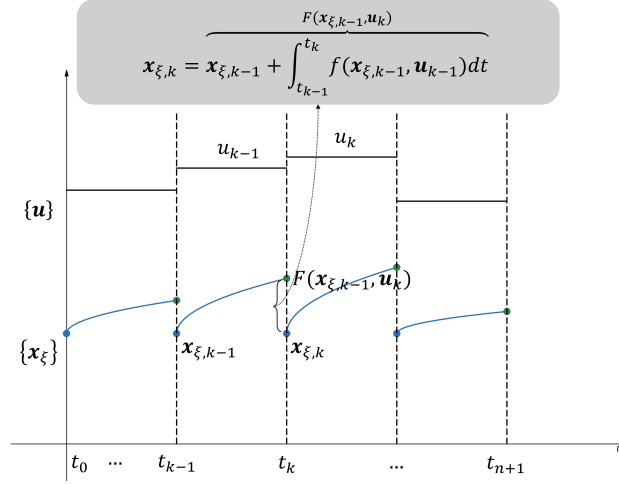


Figure 6.2: Illustration of the Direct Multiple Shooting method. The discretized state trajectory $\{\mathbf{x}\}$ contains $N + 1$ knot points and N intervals, and the control trajectory $\{\mathbf{u}\}$ is applied piecewise over these intervals. To ensure continuity, at each knot point, the state trajectory is computed by numerically integrating the state-space equation, and equality constraints are enforced between the integrated state and the corresponding next knot point.

zon is normalized to a fixed interval $[0, N]$, enabling a consistent discretization of the continuous-time system. Moreover, as explained in Section 5.2.3, the stage constraints in equation (6.1f) are already embedded in the state bounds after spatial reformulation. Hence, the problem (6.1) is transformed into a standard NLP formulation as follows:

$$\min_{\{\mathbf{x}_{\xi}\}, \{\mathbf{u}\}, \Delta t} T \quad s.t. \quad (6.3a)$$

$$\Delta t = T/N \quad (6.3b)$$

$$\mathbf{x}_{\xi,0} = \mathbf{x}_0, \quad \mathbf{x}_{\xi,N+1} = \mathbf{x}_T, \quad (6.3c)$$

$$\mathbf{x}_{\xi,k+1} = F(\mathbf{x}_{\xi,k}, \mathbf{u}_k, \Delta t), \quad k = 1, \dots, N \quad (6.3d)$$

$$\underline{\mathbf{x}_{\xi,k}} \leq \mathbf{x}_{\xi,k} \leq \overline{\mathbf{x}_{\xi,k}}, \quad k = 1, \dots, N + 1 \quad (6.3e)$$

$$\underline{\mathbf{w}_{\xi,k}} \leq \mathbf{w}_{\xi,k}(s) \leq \overline{\mathbf{w}_{\xi,k}}, \quad k = 1, \dots, N + 1 \quad (6.3f)$$

$$\underline{\mathbf{u}_k} \leq \mathbf{u}_k \leq \overline{\mathbf{u}_k}, \quad k = 1, \dots, N \quad (6.3g)$$

$\{\mathbf{x}_\xi\}$ and $\{\mathbf{u}\}$ are denoted as the discretized state and control trajectories. The time-scaled system ODE (6.2) is numerically solved on each interval as an initial value problem (IVP), indicated by the function F in equation (6.3d). The explicit 4th order Runge-Kutta (ERK4) integrators are adopted, which are fast and precise enough for most applications. Continuity constraints ensure that the end of one interval coincides with the beginning of the next. The graphical illustration for the method is shown in Fig. 6.2.

6.2.2 Trajectory Tracking

6.2.2.1 Trajectory Handler

With the reference trajectory provided by either the initial path or optimized path, the trajectory handler module generates the reference state $\mathbf{x}_{ref,k}$ for the NMPC tracking module in Fig. 6.1. We adopt the approach similar to [Li et al., 2024] when the globally optimized path is available. Given the current system states $\mathbf{x}_{ref,k}$ is obtained by shifting the previous reference tracking state by one additional step. In case only the initial path is available, we adopt the progress-maximization strategy proposed in [Kloeser et al., 2020, Verschueren et al., 2014]. Other state parameters are zero, and the reference progress is chosen to be slightly out of reach as $s_{k,ref} = s_k + \frac{s_{N,ref}}{N}k$. This has been proven to converge to optimality in [Verschueren et al., 2014].

6.2.2.2 Nonlinear Model Predictive Control

The NMPC serves as the online trajectory tracking controller of the hierarchical framework. Given the reference state computed in the trajectory handler module, its primary objective is to solve the optimal inputs and to control the system in following the time-optimal reference trajectories in real-time.

The NMPC problem structure is pretty much the same as equation (6.3), where the difference is the cost function and the optimization horizon. The NMPC problem is written as

$$\min_{\substack{\mathbf{x}_0, \dots, \mathbf{x}_N \\ \mathbf{u}_0, \dots, \mathbf{u}_N}} \sum_{k=0}^{N-1} \left((\mathbf{x}_k - \mathbf{x}_{k,\text{ref}})^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_{k,\text{ref}}) + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k \right) \quad (6.4a)$$

$$+ (\mathbf{x}_N - \mathbf{x}_{N,\text{ref}})^T \mathbf{Q}_N (\mathbf{x}_N - \mathbf{x}_{N,\text{ref}})$$

$$\mathbf{x}_0 = \mathbf{x}_c \quad (6.4b)$$

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k, \Delta t), \quad k \in 0, \dots, N-1 \quad (6.4c)$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}}, \quad k \in 0, \dots, N-1 \quad (6.4d)$$

$$\underline{w}_1 \leq w_1^k \leq \overline{w}_1, \quad k \in 0, \dots, N-1 \quad (6.4e)$$

$$\underline{w}_2 \leq w_2^k \leq \overline{w}_2, \quad k \in 0, \dots, N-1 \quad (6.4f)$$

$$\underline{\phi} \leq \phi_k \leq \overline{\phi}, \quad k \in 0, \dots, N-1 \quad (6.4g)$$

$$\underline{\theta} \leq \theta_k \leq \overline{\theta}, \quad k \in 0, \dots, N-1 \quad (6.4h)$$

$$\underline{\psi} \leq \psi_k \leq \overline{\psi}, \quad k \in 0, \dots, N-1 \quad (6.4i)$$

Here, $\mathbf{x}$ is the system state, and F is the fourth-order Runge-Kutta numerical integrator for the spatial quadrotor dynamics. $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q}_N$ are positive definite weighting matrices. Constraints on transverse coordinates w can be enforced by path boundary functions of the progress variable s in the state, i.e., $\underline{w}_1(s)$, $\underline{w}_2(s)$, $\overline{w}_1(s)$, and $\overline{w}_2(s)$ generated in Section 5.3. Upper and lower bounds of Euler angles are denoted by $\underline{\phi}$, $\overline{\phi}$, $\underline{\theta}$, $\overline{\theta}$, $\underline{\psi}$, and $\overline{\psi}$ in Equations (6.4g–i).

Note that the constraints are the same as in equation (6.3), and N represent the prediction sampling interval for the NMPC, in which the prediction horizon is shorter than the total flight time. $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q}_N$ are the weighting matrices penalizing tracking errors, control efforts, and terminal state errors. $\mathbf{R}$ must be fine-tuned to prevent oscillatory behavior.

6.3 Numerical Results

6.3.1 Simulation Setup

To evaluate our framework, simulation experiments were conducted on a desktop computer equipped with an AMD Ryzen 7 5800X 8-Core Processor and 32 GB of RAM, running Ubuntu 22 OS. The Global Trajectory Optimization module, Trajectory Handler module, and NMPC module, as shown in Fig. 6.1, were implemented in Python 3.8. We modify the built-in IRIS

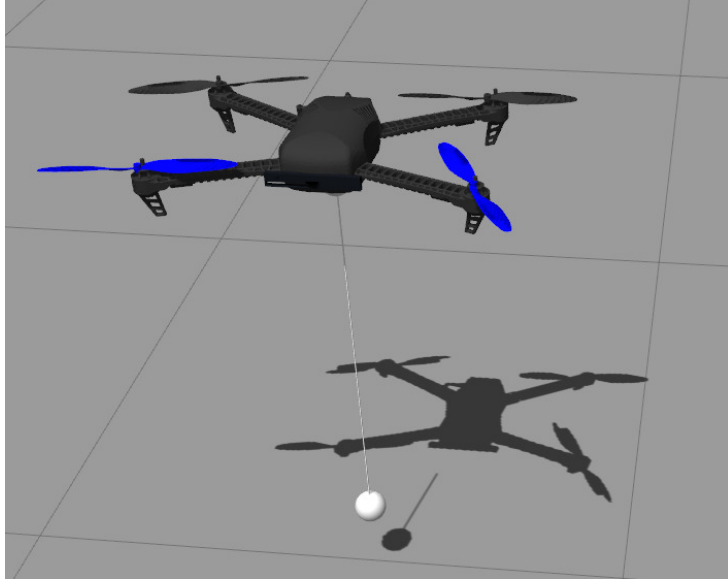


Figure 6.3: The simulated CSPQ system model. The system consists of an IRIS quadcopter, a massless tether, and a spherical payload.

quadcopter model in PX4 SITL to include the tether and payload with a modification that it carries a payload, which is shown in Fig. 6.3.

6.3.2 Global Optimization Results

We designed three scenarios to comprehensively validate the effectiveness of the global optimization module. The first and the second scenarios are analytical trajectories inspired by [Zhang et al., 2025], which are intended to validate the completeness of the GN frame method. Scenario 1 presents an analytical trajectory Γ_{E1} with a single zero-curvature singularity, consisting of two segments of Euler spirals expressed as follows:

$$\Gamma_{E1}(s) = \begin{bmatrix} 4 \operatorname{Fresnelc}(5(s - 0.5)) \\ 4 \operatorname{Fresnels}(5(s - 0.5)) \\ 6t - 3 \end{bmatrix}, \quad t \in [0, 1] \quad (6.5)$$

where $\operatorname{Fresnelc}(s) = \int_0^s \cos\left(\frac{\pi\tau^2}{2}\right) d\tau$ and $\operatorname{Fresnels}(s) = \int_0^s \sin\left(\frac{\pi\tau^2}{2}\right) d\tau$. Γ_{E1} is C^4 continuous with the inflection point at $s = 0.5$, illustrated as the dashed center line in Fig. 6.4(d). Fig. 6.4(a)-(c) depicts the FS, PT and GN

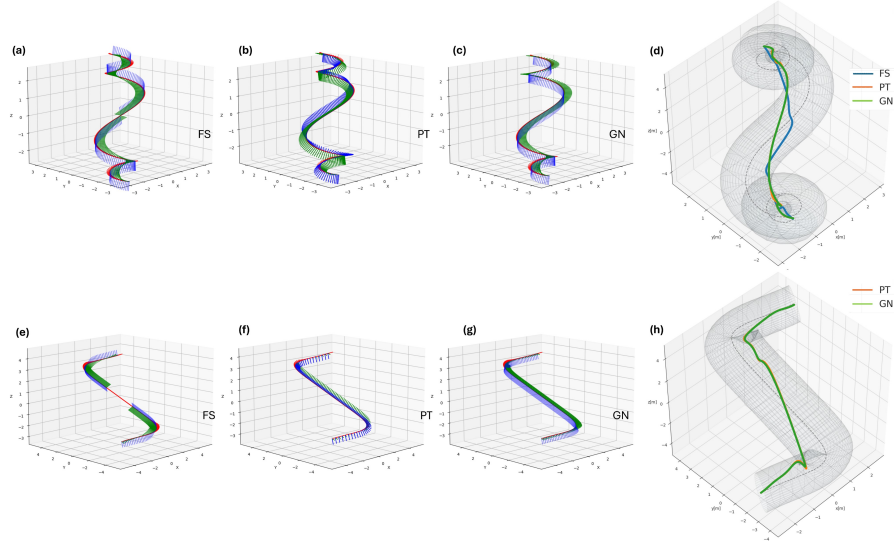


Figure 6.4: (a), (b) and (c): comparison of the FS, PT and GN frames on scenario 1 respectively. (d): Trajectory states plots resulting from the three frame constructions in scenario 1. (e), (f), (g): comparison of the FS, PT and GN frames on scenario 2 respectively. (h): Trajectory states plots resulting from the three frame constructions in scenario 2, where the FS frame fails to converge.

frame axes along scenario 1 respectively. Again, near the inflection point the FS frame just flips instantaneously. The PT frame remains continuous along the curve but undergoes constant torsion, resulting in continuous rotation.

Scenario 2 is an analytical trajectory Γ_{E2} composed of two Euler spiral

segments and a straight line, expressed as

$$\mathbf{\Gamma}_{E2}(s) = \begin{cases} \begin{bmatrix} 4 \sin(75(s - 0.4)^5) \\ 4 \text{Fresnelc}(4(s - 0.4)) - 1.6 \\ 6s - 2.4 \end{bmatrix}, & s \in [0, 0.4] \\ \begin{bmatrix} 0 \\ 16(s - 0.5) \\ 6s - 2.4 \end{bmatrix}, & s \in (0.4, 0.6) \\ \begin{bmatrix} 4 \sin(75(s - 0.6)^5) \\ 4 \text{Fresnelc}(4(s - 0.6)) + 1.6 \\ 6s - 2.4 \end{bmatrix}, & s \in [0.6, 1.1] \end{cases} \quad (6.6)$$

, which is also C^4 continuous but with continuous zero curvature for $s \in [0.4, 0.6]$. It is illustrated as the dashed center line in Fig. 6.4(h). Fig. 6.4(e)-(g) depicts the FS, PT and GN frame axes along $\mathbf{\Gamma}_{E2}$ respectively. In this case, the PT frame continues to rotate gradually, while the FS frame becomes undefined along the straight-line segments.

To better approximate real-world scenarios, the third scenario consists of a Bezier curve, which aligns with most mainstream UAV trajectory planners such as [Mellinger and Kumar, 2011, Zhang et al., 2023b, Zhou et al., 2019]. It is widely adopted due to the following advantages: (1) Its convex hull property ensures that the curve is confined within the convex hull spanned by its control points, which is useful for defining free spaces. (2) Its hodograph property guarantees a smooth and continuous derivative along the curve, which is essential for spatial reformulation. (3) It has a universal measure to regulate its progress along the path since the parameterization always ranges

from $[0,1]$. The representation is defined as

$$P(s) = \sum_{i=0}^n B_i^n(s) P_i \quad (6.7)$$

, where P_i are the corresponding $(i + 1)th$ control points for $i=0, \dots, n$. By definition, s always varies within the range $[0, 1]$. Denoting $\binom{n}{i}$ as the binomial coefficient, $B_i^n(s)$ is the Bernstein polynomial given by [Kamermans, 2011–2020]

$$B_i^n(s) = \binom{n}{i} (1-s)^{n-i} (s)^i, \quad i \in \{0, 1, \dots, n\} \quad (6.8)$$

The curve in scenario 3 is composed of three parts: a climbing helix, an arc-shaped motion, and a descending helix with a total length of 272.3 m. The trajectory is illustrated as the dashed center line in Fig. 6.5.

The OCP in the Global Trajectory Optimization module was formulated in the CasADi software framework [Andersson et al., 2019], and it is solved by the large-scale nonlinear optimization solver IPOPT with MUMPS as the linear solver back-end.

To validate the effectiveness of the proposed GN frame spatial reformulation in computing time-optimal trajectories, the performance of the resulting trajectory is compared against that of the two aforementioned curve-framing methods which are well-established and extensively studied. The constraint parameters as well as CSPQ system parameters used in all methods are summarized in Table 6.1.

For direct comparison, the free configuration space for the three framing methods are chosen to be in the circular tunnel with a radius of 1m surrounding the initial reference curve. i.e. $\sqrt{w_1(s)^2 + w_2(s)^2} < 1$, for $s \in [0, 1]$. w_1 and w_2 are the respective transverse coordinates. Also, we use $N_p = 100$ shooting nodes for numerical optimization.

Leveraging the aforementioned modular components in the offline stage (as shown in Fig. 6.1), the global time-optimal trajectories are obtained. To evaluate the reliability of our multiple-shooting method, we analyze the

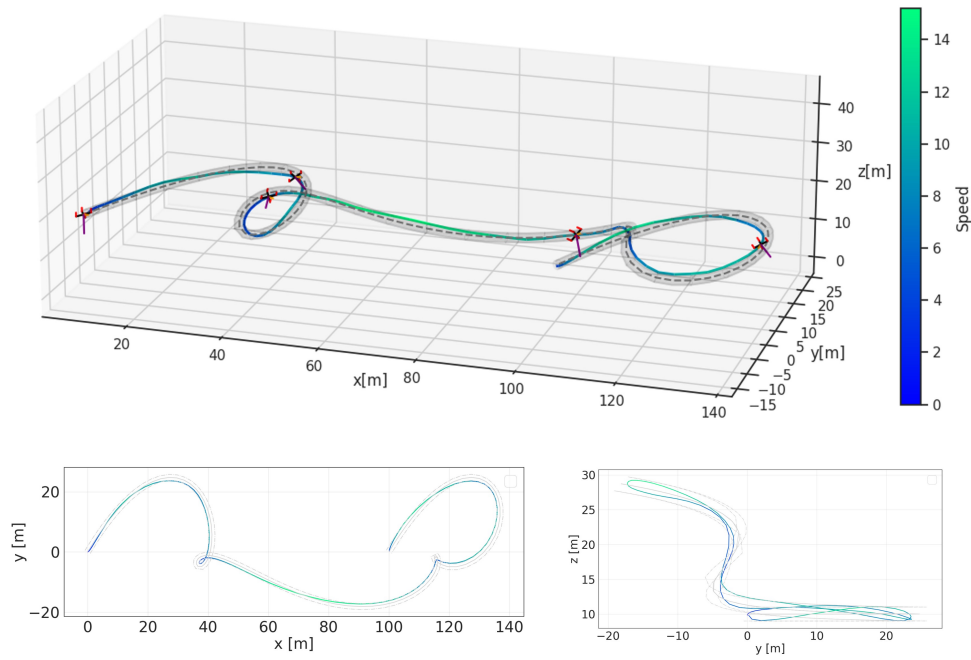


Figure 6.5: The resultant global time-optimal trajectory obtained for our proposed GN frame construction method in scenario 3. *left*: the isometric view showing the trajectory, with the gray tunnel visualizing the flight corridor generated in the GN frame. The color indicates speed in m/s. *Top right*: the top view of the trajectory. *Bottom right*: the side view of the trajectory.

Table 6.1: CSPQ Parameters and Constraints

Parameters	Value	Unit
m_Q	1.2	kg
m_L	0.1	kg
l	0.65	m
g	9.81	m/s ²
Constraints	Range	Unit
States $\mathbf{x}_\xi$:		
$\mathbf{w}$	$\{w_1^2 + w_2^2 \leq 1\}$	m
$\mathbf{q}$	$\{q_x, q_y\} \in [-0.3, 0.3], q_z \in [-1.0, -0.8]$	
$\mathbf{v}_Q$	$\{v_x, v_y\} \in [-10, 10], v_z \in [-5, 5]$	m/s ²
Φ	$\{\phi, \theta, \psi\} \in [-\pi/6, \pi/6]$	rad
Inputs $\mathbf{u}$:		
Ω	$\{\Omega_p, \Omega_q, \Omega_r\} \in [-8, 8]$	rad/s
T_Q	$T_Q \in [1.275, 51.012]$	N

Table 6.2: Performance comparison across all scenarios. The initial arc-length for scenario 1, 2 and 3 are 20.88, 19.60m and 272.3m respectively.

	Arc-length (m)	Flight Time (s)	Max Speed (m/s)	Time for Full Thrust (%)
Scenario 1				
FS	18.96	3.81	7.80	6.35
PT	18.90	3.80	7.90	6.30
GN	17.60	3.75	8.06	7.01
Scenario 2				
FS	—	—	—	—
PT	16.27	2.71	5.42	6.30
GN	16.35	2.69	5.75	7.30
Scenario 3				
FS	263.48	32.36	14.94	52.63
PT	261.54	32.17	15.27	51.12
GN	261.34	32.02	15.31	52.70

sparsity of the constraint Jacobian and constraint Hessian matrices in all scenarios, where for conciseness only the result for scenario 3 is as shown in Fig. 6.6. The result confirms that both matrices are sparse in all scenarios, with only around 15.0% of their elements being non-zero.

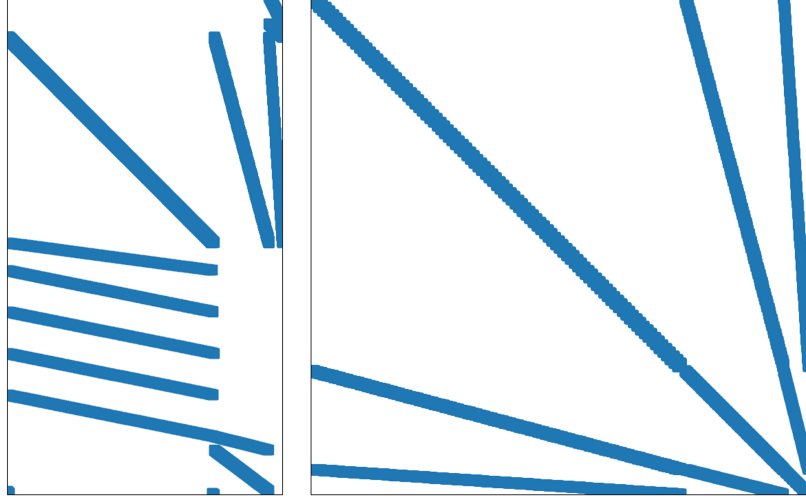


Figure 6.6: Jacobian (*left*) and Hessian (*right*) for scenario 3 for illustration

6.3.2.1 Analytical Trajectories

Fig. 6.4(d) and (h) show the resulting global trajectories for Scenario 1 and Scenario 2 respectively using the three frame construction methods, where both scenarios are analytical. The navigation times and additional performance metrics for the FS, PT, and GN frames are summarized in Table 6.2. In both cases, the GN frame slightly outperformed the other methods, achieving shorter total flight times, reduced path lengths, higher maximum speeds, and longer durations at full thrust. Notably, the FS frame performed poorly in both scenarios: in Scenario 1, it deviated from the optimal path, and in Scenario 2, the optimization failed to converge. These results further support the premise of this study, highlighting the deficiencies of the FS frame formulation for trajectory optimization in cases involving zero curvature.

6.3.2.2 Bezier Curve Trajectories

Indeed, analytical trajectories have limited applicability in real-world scenarios. In this chapter, we conducted a more comprehensive analysis of performance in Scenario 3, which utilized Bezier curve trajectories just like some state-of-the-art global planners. The results demonstrate that our pro-

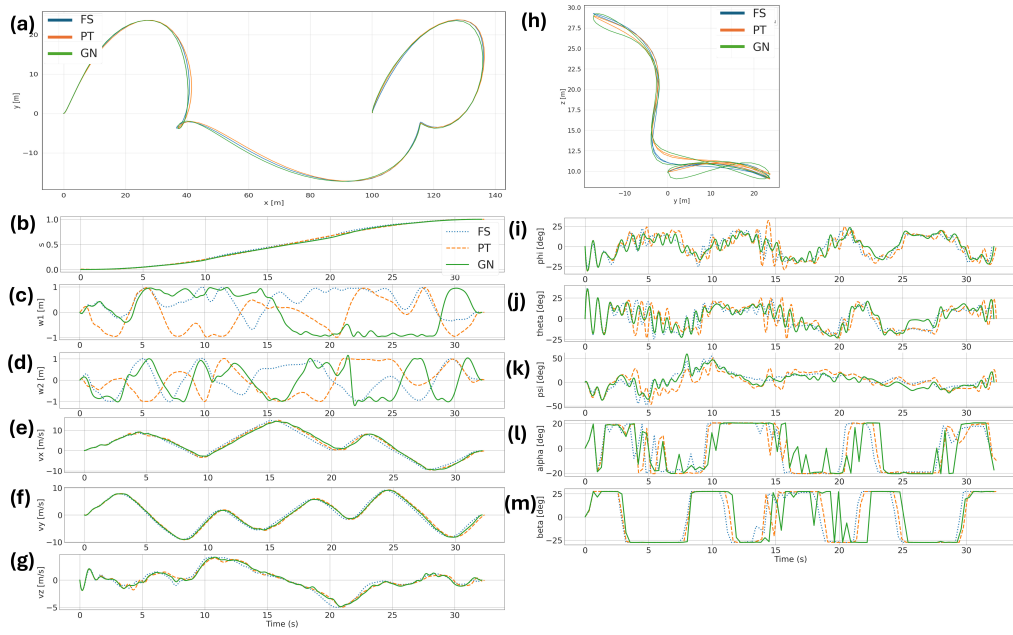


Figure 6.7: (a) and (h): the 2D projection of the computed trajectories on the xy -plane and the yz -projection in scenario 3. (b), (c) and (d): the plot for the progress variable and the two transverse coordinates over time. (e), (f) and (g): linear velocities over time. (i), (j) and (k): Euler angles over time. (l) and (m): α and β over time.

posed offline optimization module, employing the GN frame, is capable of generating time-optimal trajectories. Fig. 6.5 shows the resultant global time-optimal trajectory obtained for all frame construction methods in scenario 3.

Fig. 6.7 represents the xy projection of the resulting trajectories for scenario 3 generated by different framing methods, while Fig. 6.7(a) and (h) represents the xy and yz projection for scenario 3 correspondingly. The optimized results among different methods are more correlated, but still the deviation in z direction is large. On the other hand, the total flight time across all cases is similar, although the proposed GN frame slightly outperformed the other benchmarks.

The state plots for the trajectories generated by all frame construction methods in scenario 3 are presented in Fig. 6.7. Firstly, the plot of path progress versus time in Fig. 6.7(b) clearly illustrates the percentage of trajectory completion, which is one of the motivations of our method. Then, in all cases, although the two transverse coordinates w_1 and w_2 in Fig. 6.7(c) and (d) are different because of different frame representations, they all satisfy the state bounds $\sqrt{w_1^2 + w_2^2} < 1$. Referring to in Fig. 6.7(e)-(g), the velocities in ${}^I x$, ${}^I y$, and ${}^I z$ directions satisfy the limit of $[-10, 10]$, and they all share the same characteristics. In terms of the Euler angles Fig. 6.7(i)-(k), they behave differently across different frame construction methods. Aggressive maneuvers are observed as the roll and pitch angles frequently approach their limits of -30 to 30 degrees. It is believed that the oscillations originate from the quadcopter's efforts to suppress the swinging motion of the slung load. Still, the overall trajectory remains feasible, and the differences highlight the influence of the chosen framing method on the vehicle's orientation dynamics.

One reason for the discrepancies, despite sharing the same configuration space, can be attributed to numerical implementation. Moreover, in the case of the Frenet-Serret (FS) frame, extreme curvature and torsion values in regions near inflection points can cause numerical instability or reduced smoothness in the optimization process. This is because the FS frame is highly sensitive to rapid changes in curvature and torsion, leading to abrupt

shifts in reference frame orientation and complicating the numerical handling of the optimization problem. In contrast, both the Parallel Transport (PT) and GN frames mitigate these issues by maintaining smooth orientation transitions, resulting in slightly different optimized trajectories. However, as the PT frame is derived through approximations, minor discrepancies may still arise and be reflected in the optimization problem.

To better illustrate whether the payload oscillation is bounded in all cases, the Euler angles of the payload position are also computed. An additional payload frame $\{L\}$ is also defined whose origin coincides with the quadcopter body center of mass, while its orientation always coincides $\{I\}$. By doing so, we also express the position of the payload as

$$\mathbf{p}_L = \mathbf{p}_Q + {}^I\mathbf{R}_L {}^L\mathbf{p}_{L,0} \quad (6.9)$$

such that

$${}^I\mathbf{R}_L = \mathbf{R}(\alpha)\mathbf{R}(\beta) = \begin{bmatrix} c_\alpha & s_\alpha & s_\alpha c_\beta \\ 0 & c_\beta & -s_\beta \\ -s_\alpha & s_\beta c_\alpha & c_\alpha s_\beta \end{bmatrix} \quad (6.10)$$

In equation (6.9), $\mathbf{p}_{L,0} = {}^L\begin{bmatrix} 0 & 0 & -l \end{bmatrix}$ is the natural position when the CSPQ system is stationary, which is pointing downward. α is the rotation angle along ${}^Ly$ axis and β is the rotation angle along ${}^Lx$ axis. The relation between α , β and $\mathbf{q}$ is given by

$$\alpha = \text{atan2}(\mathbf{q}_x, \mathbf{q}_z) \quad (6.11a)$$

$$\beta = \text{asin}\left(-\frac{\mathbf{q}_y}{l}\right) \quad (6.11b)$$

The results are shown in the bottom right plot in Fig. 6.7(l) and (m). It is evident that α is bounded between -20 to 20 degrees and β is bounded between -25 to 25 degrees. While further restricting the payload is possible, such adjustments inevitably come at the expense of reduced flight time.

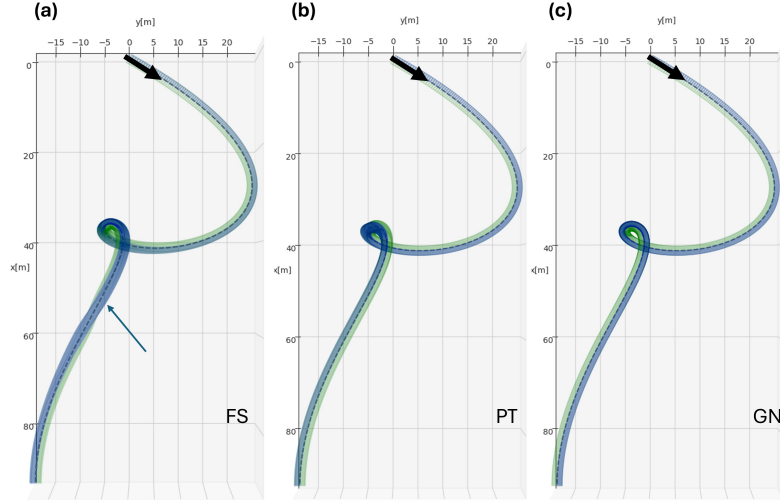


Figure 6.8: Comparison of the sample flight corridor generated in (a) FS, (b) PT and (c) GN frames along scenario 3.

6.3.2.3 Flight Corridor Analysis

Meanwhile, the effectiveness of the GN frame representation arises from its ability to accurately describe the flight corridor. Fig. 6.8(a)–(c) compare the results generated by the three curve-framing methods for a simple flight corridor, defined as $\{(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \mid \mathbf{e}_1 \in [0, 1], -1 \leq \mathbf{e}_2 \leq 1, -1 \leq \mathbf{e}_3 \leq 1\}$, in Scenario 3. For clarity, the “positive” region where $\mathbf{e}_2 > 0$ is shown in blue, while the “negative” region where $\mathbf{e}_2 < 0$ is shown in green. The black arrow indicates the initial position and orientation of the CSPQ system.

In Fig. 6.8(a), the flight corridor represented in the FS frame flips at the location indicated by the blue arrow. For the PT frame corridor shown in Fig. 6.8(b), the blue corridor initially corresponds to the left side of the system, but later shifts to the right. In contrast, as shown in Fig. 6.8(c), the GN frame corridor consistently maintains the blue region on the left side throughout the trajectory, thereby enabling a clear and unambiguous description of the path.

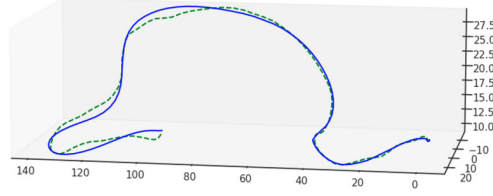


Figure 6.9: NMPC Trajectory tracking response for Scenario 3

6.3.3 NMPC Results

6.3.3.1 Bezier Curve Trajectoryies

For the NMPC module, the optimal control framework *acados* [Verschuere et al., 2022] was employed as the NMPC solver. By default, it adopts the direct multiple shooting approach and minimizes equation (6.4) using the inbuilt sequential quadratic programming (SQP) method. The explicit 4-th order Runge–Kutta method is adopted for numerical integration. To address real-time applicability, we used its real-time iteration variant (SQP-RTI) [Diehl et al., 2005], combined with HPIPM to solve the SQP efficiently.

The prediction horizon is set to be 1s with 20 shooting nodes, equivalent to a sampling time of 50Hz. The total simulation time is 50s. The weighting matrices are selected as

$$\begin{aligned} \mathbf{Q} &= \text{diag}(1e1\mathbf{I}_1, \mathbf{I}_2, 1e-1\mathbf{I}_3, 1e-8\mathbf{I}_6, 1e-4\mathbf{I}_3) \\ \mathbf{R} &= \text{diag}(1e-3, 5e-3\mathbf{I}_3) \\ \mathbf{Q}_N &= \text{diag}(1e2\mathbf{I}_3, 1e-1\mathbf{I}_3, 1e-8\mathbf{I}_6, 1e-2\mathbf{I}_3) \end{aligned}$$

, with $\mathbf{I}_n$ referring to a $n \times n$ identity matrix. The state constraints are the same as in Table 6.1.

All simulated tests are conducted in the Gazebo simulator environment with ROS [Quigley et al., 2009a] as the interfacing middleware. The position controller inbuilt in PX4-SITL is bypassed and the computed input is directly forwarded to the angular rate controller. The NMPC module update the

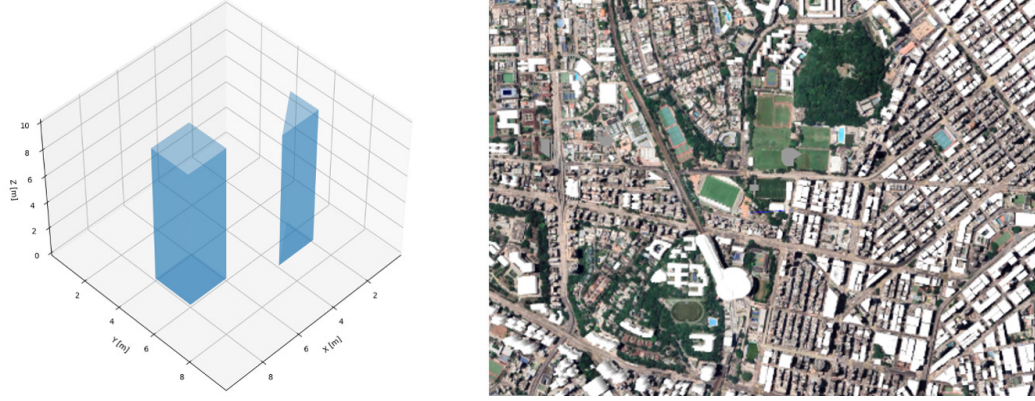


Figure 6.10: Graphical illustration of two test areas)

quadcopter state and payload state before every NMPC iteration.

Fig. 6.9 shows the comparison between the reference time-optimal trajectory and the executed trajectory for scenario 3. The flight time achieved by the NMPC was $33.62s$ which was $1.6s$ more. One of the main reason lies on the cost function. In equation (6.4), the $\mathbf{R}$ matrix is necessary for suppressing the oscillatory behaviour in the input, but at the expense of flight performance. Also, in offline trajectory module it assumes an instant, ideal response from controllers. Regarding computation time, the average computation time for each NMPC iteration is $3.52ms$ with a maximum of $4.70ms$. These timings allow for the deployment of the online module in our proposed framework in a real-time fashion.

6.3.4 Urban Environment

In this section, to further validate the effectiveness of the proposed trajectory optimization framework and the flight corridor in real life situations, we conducted computer simulation experiments in two simulated environments. This time the direct attachment quadcopter is used. The test results were obtained on a desktop PC equipped with an AMD® Ryzen 7 5800x 8-core CPU, and 32 GB of RAM running Ubuntu 20.04 LTS.

First, we consider a simplified scenario as depicted in Figure 6.10(a) for the sake of a clearer visualization of the resulting flight corridor subject to surrounding obstacles, as well as the details of the flight trajectory. The full

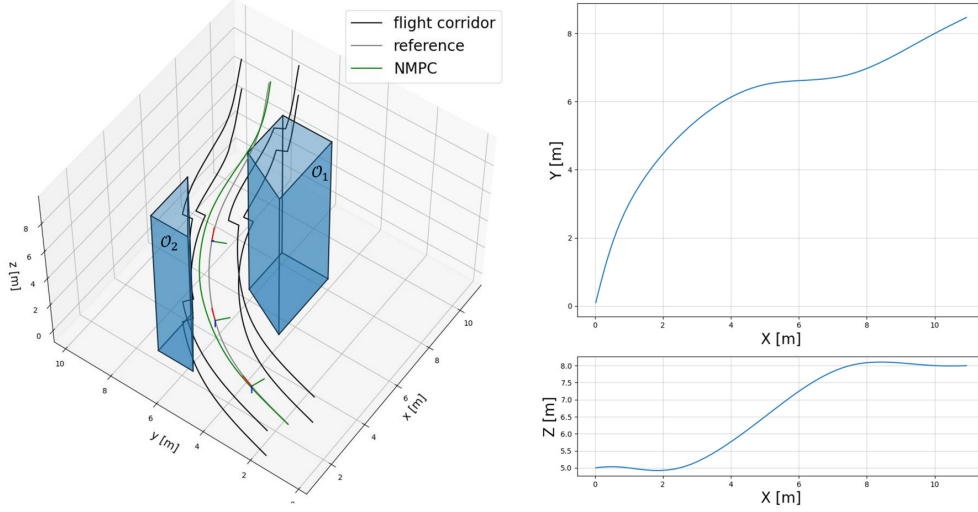


Figure 6.11: Visualisation of the resulting flight corridor

dynamical behaviour of the trajectory is also presented.

Second, to fully explore the capabilities of the framework in drone parcel delivery, we consider a dense urban area depicted in Figure 6.10(b), which contains the buildings and terrain features of certain areas of Prince Edward, Hong Kong.

In both environments, we follow the ideal specification of an Amazon Prime air delivery UAV provided in Sunghun and Hyunsu [2017], considering battery capacity, weight, and motor dynamics, with a total mass of 6.1kg, maximum velocity of 22.36m/s, and arm length of 1.15m.

Regarding the NMPC parameters, we select the prediction horizon $N = 50$ shooting nodes corresponding to $t_f = 1$ CPU second, with a sampling time of 20ms.

6.3.4.1 A Simplified Scenario

For better illustration of the flight corridor generation, the first scenario involves a drone navigating through two polygonal prisms, mimicking the behavior of a parcel delivery drone operating in an urban area with dense

buildings. Each unit on the grid represents 1 meter throughout the chapter. The origin-destination (OD) pair is chosen from $(0, 0, 5)$ to $(11.0, 8.5, 8.0)$, where two prisms as obstacles O_1 and O_2 are placed at the scene with centers at $(7, 5)$ and $(2.67, 6.5)$ with a height of 10 m, respectively.

We adopt the approach proposed in chapter 4 for generating the initial reference path $\Gamma(t)$. The GN frame components are calculated accordingly by the approach presented in 5.2.3.

The initial flight corridor parameters are chosen as $w_1^{\min} = -1$, $w_1^{\max} = 1$, $w_2^{\min} = -1$, and $w_2^{\max} = 1$. Both obstacles O_1 and O_2 are intruding on the flight corridor, triggering the corridor update. For both obstacles, their dominant sides are at $\mathbf{e}_2$. For obstacle O_1 , it is at the positive $\mathbf{e}_2$ direction and O_2 is at the negative $\mathbf{e}_3$ direction. The resulting flight corridor is indicated as the tunnel bounded by solid black lines in Figure 6.11(a).

The weighting matrices in Equation (6.4a) remain constant throughout both simulation environments, as $\mathbf{R} = \mathbf{I}_4$, $\mathbf{Q} = \text{diag}(10, 10^{-10}\mathbf{I}_2, \mathbf{I}_3, 10^{-10}\mathbf{I}_6)$, and $\mathbf{Q}_N = \text{diag}(10, 10^{-10}\mathbf{I}_2, 10\mathbf{I}_3, 10^{-10}\mathbf{I}_6)$. To enforce time-optimality, only elements in state x with respect to the progress and their derivatives, along with rotor thrusts, are penalized, and the remaining diagonal values are kept small to ensure numerical stability. This ensures the maneuverability of the quadcopter is undisrupted as long as the quadcopter is within the flight corridor. The resulting trajectory is depicted in Figure 6.11(a), where the trajectory is projected onto x - y and x - z planes in Figure 6.11(b).

The total flight time of the minimum-time trajectory is 5.87 s. The state transition of the quadcopter over time is shown in Figure 6.12(a). The velocity plot in Figure 6.12(b) has been converted to x , y , and z directions in the inertial frame for clearer interpretation. The maximum speed achieved is 8.89 m/s, which is below the nominal maximum speed of the quadcopter. This is attributed to the limitation on the maximum roll and pitch angles during the flight, which led to less aggressive dynamics and reduced forward acceleration. While it hinders the performance, it is critical for stability for both the quadcopter itself and the parcel. It is reflected in Figure 6.12(c), where the roll angle and pitch angle are limited to the range from -45° to 45° . Referring to Figure 6.12(e), the quadcopter was near full throttle at the

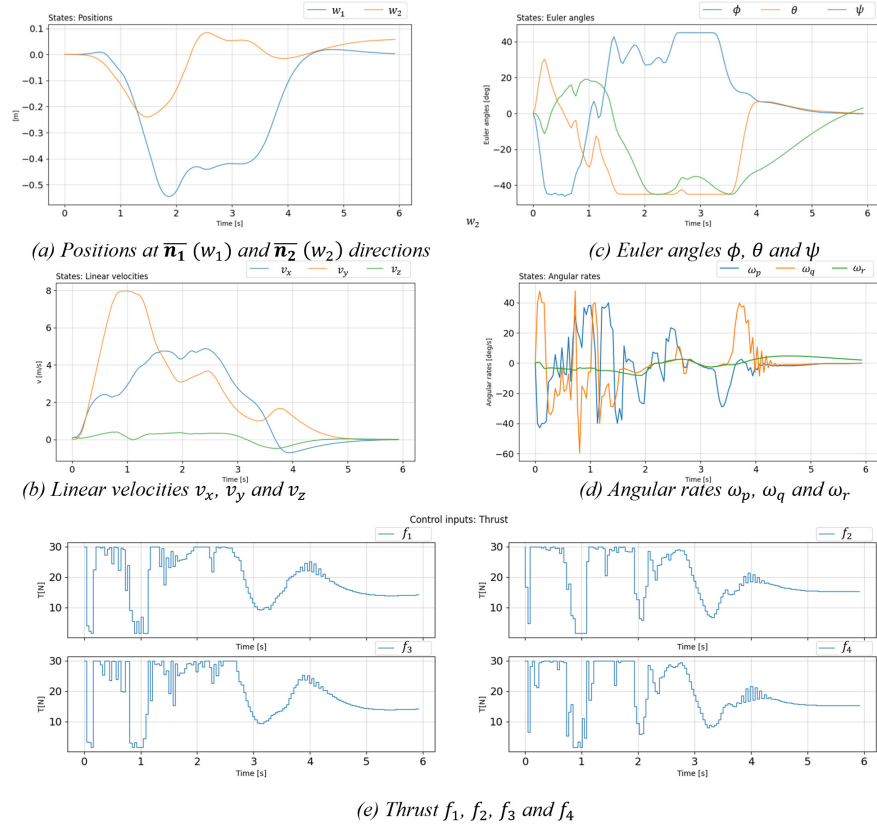


Figure 6.12: Velocities and angles plots

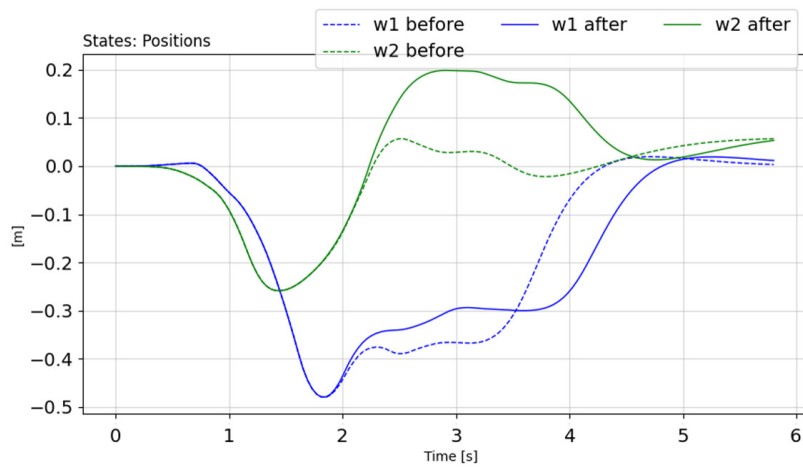


Figure 6.13: Obstacle avoidance

interval of $[0.2\text{ s}, 0.79\text{ s}]$ and $[1.1\text{ s}, 2.67\text{ s}]$ for rapid changes of flight attitude at that moment, which corresponds to changing flight directions.

The total flight time is compared to the time-optimal trajectory in the global optimization module. The corresponding solution is treated as the lower bound as it exhibits a bang-bang control behavior with more aggressive maneuvers. The optimal flight time is 5.60 s, which is 4.60% better than our proposed approach. On the other hand, the computation time of the optimal approach takes 7.31 s to compute, in contrast to our NMPC approach of 52 ms. This enables the capability for real-time trajectory optimization and also real-time adjustments.

The agility of NMPC trajectory computation enables real-time collision avoidance. In the same experimental setup, after the prediction time of 1 s, we introduced a sudden abrupt change in $\mathbf{e}_2$ direction where $w_1^{\min}$ is decreased in magnitude to -0.3 m . It is to mimic the sudden detection of obstacles besides buildings, such as flocks of birds or other drones. For any other incoming obstacles b , their transverse coordinates w_1^b and w_2^b can also be computed using the corridor generation method described in Section 5.3. Figure 6.13 shows the resulting trajectory, in which w_1 is now reaching the bound at -0.3 and w_2 is deviating from the original trajectory to compensate for the quadcopter dynamics. Also, we tried to decrease the prediction horizon to less than 1 s, but the optimization process failed for an abrupt change. In this setup and quadcopter configuration, the minimum prediction horizon would be around 1 s.

Table 6.3: Comparison of Time-Optimal and NMPC Approaches for Two OD Pairs

Expected time for Time-Optimal Trajectory			
OD Pair	Distance (m)	Flying Time (s)	Compute Time (s)
1	255.31	15.80	13.55
2	287.93	18.77	13.78
NMPC Tracking results			
OD Pair	Distance (m)	Flying Time (s)	Corridor Gen. Time (ms)
1	260.70	16.20	0.41
2	298.98	19.05	0.43

We chose certain areas of Prince Edward, Hong Kong, as the testbed

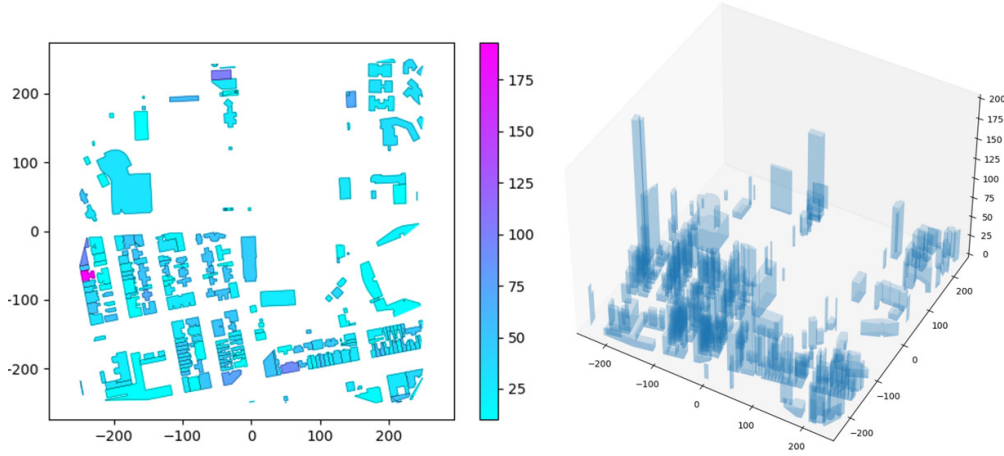


Figure 6.14: Graphical illustration of the urban environment

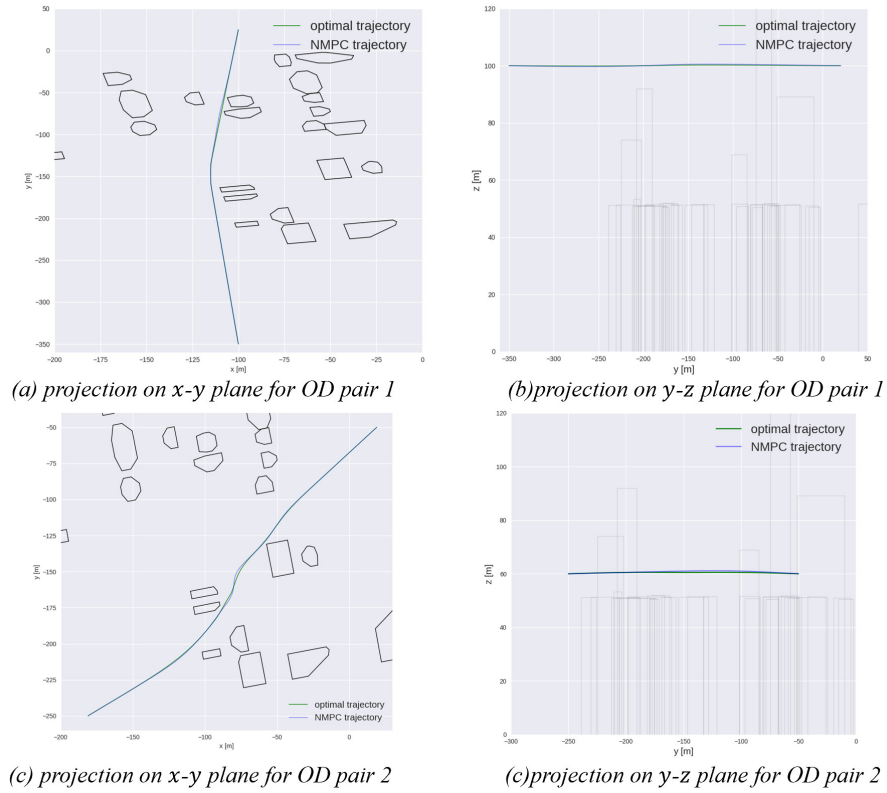


Figure 6.15: Graphical illustration of the paths generated for the 2 OD pairs

for simulating an urban environment, which is shown in Figure 6.14(a) and 6.14(b). The dimensions of the area is $500\text{ m} \times 500\text{ m}$ with a total of 239 buildings inside and a maximum height of 180 m. The 3D map of the environment is extracted from the Geo-Reference Database (iG1000) provided by the Lands Department, the Government of the Hong Kong Special Administrative Region. It is a public dataset including building names and addresses, site polygons, and road centerlines. We placed the center of the map at latitude and longitude (22.32676, 114.17041).

6.3.4.2 Full Urban Scenario

We extended our framework into the full urban scenario. We are following the concept of operation (ConOps) proposed by [Jang et al., 2017], in which the airspace is divided into multiple 2D layers by altitude. Each layer is designed to have a specific airspace structure that guides and constrains the UAV, such as skylane, sky-tube, and sky-corridor systems. The delivery drone takes off at a vertiport to the designated layer, navigates with a minimum spacing requirement within the layer, and lands at the designated vertiport. We chose two origin-destination (OD) pairs, each at altitudes of 60 m and 100 m, to illustrate our results. All buildings in each layer are replaced by their 2D convex hulls for the corridor generation method.

Similar to the simplified scenario, the flight corridor width is still chosen as 1 m at both the positive and negative sides of $\mathbf{e}_2$ and $\mathbf{e}_3$ directions. Our NMPC trajectories are compared to the time-optimal results, and the results are summarized in Table 6.3. The discussions of the planned paths in the city for these two OD pairs are presented as follows:

1. The planned path for OD pair 1 (from $[-100.0, -350.0, 100]$ to $[-100, 25, 100]$) is shown in Figure 6.15(a) and 6.15(b). At an altitude of 100 m, only a few obstacles exist. The acquired path experiences only a gradual turn around a building. The flying time delta is +0.40 s. It shows that the NMPC tracking is comparable to the true-optimal trajectory in less complex scenarios.

2. The planned path for OD pair 2 (from $[-180, -250, 60]$ to $[20, -50, 60]$) is shown in Figure 6.15(c) and 6.15(d). The acquired path has to navigate over some dense low-rise buildings with various turns while passing through a large free airspace. The flying time obtained by the proposed approach is 19.05 s. Compared to the true time-optimal approach, the path is only slightly longer with a time delta of +0.28 s.

On average, the flying time of NMPC tracking is 3.10% longer than the true time-optimal one. Concerning the computation time, the time required for the NMPC approaches is less than 1 s, while the time-optimal approach requires around 13 s, which is 92.5% faster. Also, in both cases, the computation time for the flight corridor costs less than 1 ms. This is because instead of sampling lots of points for every obstacle, our corridor generation method only requires a finite number of checks concerning vertices and planes.

These results suggest the effectiveness of our method for UAV trajectory generation, especially in complex, real-life metropolitan areas. The computational burden of our method is very subtle while maintaining a reasonable flying time. This test demonstrates the applicability of the proposed approach in highly complicated working environments.

6.4 Concluding Remarks

This chapter presented a unified hierarchical framework for time-optimal motion planning and trajectory tracking of CSPQ systems, built upon the GN frame spatial reformulation. The GN frame effectively addresses issues like singularities and twisting in traditional curve-framing methods, enabling a smooth and robust spatial reformulation. By simplifying complex geometric constraints into linear or norm-bound forms, the framework facilitates efficient trajectory optimization and flight corridor generation.

Simulation results demonstrated the GN frame's superiority over traditional methods, achieving smoother trajectories and better handling of obstacle-dense environments. In urban scenarios, the framework generated feasible flight corridors and near-optimal trajectories with significantly reduced computation times, showcasing its practicality for real-world UAV

applications like parcel delivery.

Chapter 7

Conclusion

7.1 Conclusion of the thesis

In this thesis, to enhance the autonomous capabilities of drone delivery systems, we identified specific gaps in the current research and proposed a framework for time-optimal UAV trajectory optimization tailored to urban parcel delivery scenarios.

To explore the possibility of more accurate system models, apart from deriving the model through traditional approaches, we presented a Physics-Informed Attention Temporal Convolutional Network (PIATCN) for modeling and controlling Cable-Suspended Payload Quadcopters (CSPQs). We set up a TCN network with an attention mechanism and incorporate the physics-based nominal dynamics model derived from the Euler-Lagrange formulation into the loss function. This hybrid modeling approach balances the strengths of physics-based models and data-driven methods, enabling accurate predictions of the system's state while adhering to physical principles. Comprehensive simulations demonstrated the effectiveness of PIATCN in predicting the dynamics of CSPQs during various flight scenarios, including aggressive maneuvers. The results showed that PIATCN outperformed both purely physics-based models and learning-based baselines, achieving lower root mean squared error (RMSE) across key state variables such as position, velocity, orientation, and payload position.

This thesis also places particular emphasis on the use of homotopic paths in UAV trajectory planning. A novel topology-aware path planning algorithm is proposed to generate wind-efficient trajectories for UAVs operating in parcel delivery scenarios. By addressing the issues of local optima and the challenges posed by complex wind dynamics, the algorithm aims to enhance the energy efficiency of UAV flights. A key contribution of the proposed approach lies in its explicit consideration of topological distinctness. By generating paths that are structurally and spatially diverse, the algorithm enables

more comprehensive exploration of the solution space, thereby increasing the likelihood of identifying an optimal, wind-efficient trajectory.

Inspired by the path-parametric method in the field of autonomous vehicles, the thesis describes a spatial reformulation of system dynamics based on our reference path. This results in the notion of progress variable and transverse coordinates along the reference path. The emerging GN frame was introduced to address the limitations of conventional curve-framing techniques, providing a singularity-free and geometrically meaningful representation of system dynamics. Based on that, we developed a novel, accurate, and intuitive flight corridor generation method, which is critical for collision avoidance.

Integrating all the methods, this thesis proposed a unified hierarchical framework for time-optimal motion planning and trajectory tracking of UAV delivery systems. Using the predefined global path, the hierarchical framework combines an offline global optimization module, which uses direct multiple shooting to generate time-optimal trajectories, and an online NMPC module for real-time trajectory tracking. Simulation results demonstrate the advantages of the proposed framework in terms of robustness, physical interpretability, and accurate flight corridor representation while achieving the same level of flight performance, with limited payload oscillations and adherence to physical constraints. Under the same configuration, our global optimization module achieves shorter flight times compared to other benchmark methods. The online NMPC module guarantees real-time trajectory tracking with computation times in the order of milliseconds.

7.2 Areas of future research

This thesis studies various tasks related to motion planning of UAV systems and provides solutions accordingly. These studies also reveal the cutting-edge topics that may be worth researching. Here, we briefly state our research plan for the future:

1. **Real-time Capabilities of PIATCN** Our proposed PIATCN model has demonstrated a strong capability in accurately capturing highly

nonlinear dynamical effects. However, onboard testing revealed that its computational complexity poses challenges for integration into embedded MPC pipelines, particularly due to the stringent real-time requirements of high-frequency control loops. Future research will explore two potential directions: (1) training a smaller model or reducing the model’s computational burden through methods such as knowledge distillation [Hinton et al., 2015] and (2) adopting control strategies that differ from online-optimized MPC, including non-predictive approaches, eliminating online optimization, or learning the controller end-to-end [Salzmann et al., 2023].

2. **Effect of UAV Localization** In the numerical experiments, there is an implicit assumption that the exact UAV location is known during the whole flight. In actual flight, the localization error may cause a deviation between the UAV and the generated trajectory. Therefore, one direction is to account for the effect of localization error in the trajectory planning and optimization phase.
3. **Larger-scale Area** For a larger-scale area, the trajectory optimization problem is still computationally burdensome with an increased number of constraints and path length. This issue could be alleviated by appropriate partitioning of the environment at first, followed by optimizing the divided path inside partitions and combining them for the final trajectory. Still, it is worthy of investigation.
4. **Real-world Experiments** Conducting experimental validations and field tests would provide valuable insights into the practical implementation and performance of the algorithm in real-world scenarios. For example, the capability of the flight corridor and MPC tracking algorithm to compensate for moving obstacles, such as a flock of birds, should be validated. In our simulation experiment, we consider buildings as the dominant obstacles. The UAV no-fly zones can also be treated in the same fashion; however, they are time-dependent, and the proposed method has to be extended to accommodate that.

References

- E. Abbena, S. Salamon, and A. Gray. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 2017.
- V. Akgün, E. Erkut, and R. Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121(2):232–246, mar 1 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(99)00214-3.
- K. AL-Dosari, Z. Hunaiti, and W. Balachandran. Systematic review on civilian drones in safety and security applications. *Drones*, 7(3):210, 2023.
- J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.
- J. Arrizabalaga and M. Ryll. Towards time-optimal tunnel-following for quadrotors. *2022 International Conference on Robotics and Automation (ICRA)*, pages 4044–4050, 2021.
- J. Arrizabalaga and M. Ryll. Spatial motion planning with pythagorean hodograph curves. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2047–2053. IEEE, 2022.
- J. Arrizabalaga and M. Ryll. Sctomp: Spatially constrained time-optimal motion planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4827–4834, 2023. doi: 10.1109/IROS55552.2023.10341500.
- B. Ayhan and C. Kwan. Time-constrained extremal trajectory design for fixed-wing unmanned aerial vehicles in steady wind. *Journal of Guidance, Control, and Dynamics*, 41(7):1569–1576, 2018.

- R. Bader, J. Dees, R. Geisberger, and P. Sanders. Alternative route graphs in road networks. In *International Conference on Theory and Practice of Algorithms in (Computer) Systems*, pages 21–32. Springer, 2011.
- S. A. Bagherzadeh. Nonlinear aircraft system identification using artificial neural networks enhanced by empirical mode decomposition. *Aerospace Science and Technology*, 75:155–171, 2018. doi: <https://doi.org/10.1016/j.ast.2018.01.004>.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.
- D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. Direct trajectory optimization and costate estimation via an orthogonal collocation method. *Journal of Guidance Control and Dynamics*, 29:1435–1440, 2006.
- J. T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998. ISSN 0731-5090. doi: 10.2514/2.4231. publisher: American Institute of Aeronautics and Astronautics _eprint: <https://doi.org/10.2514/2.4231>.
- J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.
- S. Bhattacharya. Search-based path planning with homotopy class constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1230–1237, 2010.
- S. Bhattacharya, M. Likhachev, and V. Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 10 2012. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-012-9304-1.

- R. L. Bishop. There is more than one way to frame a curve. *The American Mathematical Monthly*, 82(3):246–251, 1975.
- R. Bordalba, T. Schoels, L. Ros, J. M. Porta, and M. Diehl. Direct collocation methods for trajectory optimization in constrained robotic systems. *IEEE Transactions on Robotics*, 39:183–202, 2023.
- B. Bukcu and M. K. Karacan. Special bishop motion and bishop darboux rotation axis of the space curve. *Journal of Dynamical Systems and Geometric Theories*, 6:27 – 34, 2008.
- G. Chen, S. Wu, M. Shi, W. Dong, H. Zhu, and J. Alonso-Mora. Rast: Risk-aware spatio-temporal safety corridors for mav navigation in dynamic uncertain environments. *IEEE Robotics and Automation Letters*, 8:808–815, 2023.
- J. Chen, T. Liu, and S. Shen. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1476–1483, 2016.
- Y. M. Chung, H. Youssef, and M. Roidl. Distributed Timed Elastic Band (DTEB) Planner: Trajectory Sharing and Collision Prediction for Multi-Robot Systems. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10702–10708, may 23 2022. doi: 10.1109/ICRA46639.2022.9811762. URL <http://arxiv.org/abs/2303.11279>. arXiv:2303.11279 [cs].
- A. P. Cohen, S. A. Shaheen, and E. M. Farrar. Urban air mobility: History, ecosystem, market potential, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):6074–6087, 2021.
- O. Cokorilo. Urban air mobility: safety challenges. *Transportation research procedia*, 45:21–29, 2020.
- J. J. Craig. *Introduction to Robotics*. Pearson Educacion, 2006.

- R. Dai and J. Cochran Jr. Path planning and state estimation for unmanned aerial vehicles in hostile environments. *Journal of guidance, control, and dynamics*, 33(2):595–601, 2010.
- M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005. doi: 10.1137/S0363012902400713. URL <https://doi.org/10.1137/S0363012902400713>.
- S. Fortune. Voronoi diagrams and delaunay triangulations. In *Handbook of discrete and computational geometry*, pages 705–721. Chapman and Hall/CRC, 2017.
- J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles. In *2013 European Control Conference (ECC)*, pages 4136–4141. IEEE, 2013.
- M. Fu, R. Rothfeld, and C. Antoniou. Exploring preferences for transportation modes in an urban air mobility environment: Munich case study. *Transportation Research Record*, 2673(10):427–442, 2019.
- S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks. Ros based autonomous mobile robot navigation using 2d lidar and rgb-d camera. In *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*, pages 151–154. IEEE, 2019.
- M. Geisert and N. Mansard. Trajectory generation for quadrotor based systems using numerical optimal control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2958–2964, 2016. doi: 10.1109/ICRA.2016.7487460.
- S. Gowroju and N. Santhosh Ramchander. Applications of drones—a review. *Drone Technology: Future Trends and Practical Applications*, pages 183–206, 2023.

- S. K. I. Graham. *Development of a quadrotor slung payload system*. University of Toronto (Canada), 2019.
- W. Gu, S. Primatesta, and A. Rizzo. Physics-informed neural network for quadrotor dynamical modeling. *Robotics and Autonomous Systems*, 171: 104569, 2024. doi: <https://doi.org/10.1016/j.robot.2023.104569>.
- M. E. Guerrero, D. A. Mercado, R. Lozano, and C. D. García. Passivity based control for a quadrotor uav transporting a cable-suspended payload with minimum swing. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6718–6723, 2015. doi: 10.1109/CDC.2015.7403277.
- G. Guban and A. Haque. Path planning for autonomous drones: Challenges and future directions. *Drones*, 7(3):169, 2023.
- A. J. Hanson. *Visualizing quaternions*, pages 1–es. Elsevier, 2005.
- A. J. Hanson and H. Ma. Parallel transport approach to curve framing. *Indiana University, Techreports-TR425*, 11:3–7, 1995.
- C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987. ISSN 0731-5090. doi: 10.2514/3.20223. publisher: American Institute of Aeronautics and Astronautics _eprint: <https://doi.org/10.2514/3.20223>.
- J. Harris, F. Arthurs, J. V. Henrickson, and J. Valasek. Aircraft system identification using artificial neural networks with flight test data. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 679–688. IEEE, 2016.
- S. Hasan. Urban air mobility (uam) market study. Technical report, NASA, 2019.
- A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2005. ISBN 7-302-10588-X.

- G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- N. V. Hoffer, C. Coopmans, A. M. Jensen, and Y. Chen. A survey and categorization of small low-cost unmanned aerial vehicle system identification. *Journal of Intelligent & Robotic Systems*, 74(1):129–145, 2014. doi: 10.1007/s10846-013-9931-6.
- D. Hong, S. Lee, Y. H. Cho, D. Baek, J. Kim, and N. Chang. Energy-Efficient Online Path Planning of Multiple Drones Using Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, 70(10):9725–9740, 10 2021. ISSN 1939-9359. doi: 10.1109/TVT.2021.3102589. event-title: IEEE Transactions on Vehicular Technology.
- C. Huang, B. Du, and M. Chen. Multi-uav cooperative online searching based on voronoi diagrams. *IEEE Transactions on Aerospace and Electronic Systems*, 60(3):3038–3049, 2024.
- D.-S. Jang, C. A. Ippolito, S. Sankararaman, and V. Stepanyan. Concepts of airspace structures and system analysis for uas traffic flows for urban areas. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0449. American Institute of Aeronautics and Astronautics (AIAA), 2017.
- M. R. Jardin and A. E. Bryson Jr. Neighboring optimal aircraft guidance in winds. *Journal of Guidance, Control, and Dynamics*, 24(4):710–715, 2001.
- 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*, pages 4569–4574, 5 2011. doi: 10.1109/ICRA.2011.5980280. ISSN: 1050-4729.
- M. P. Kamermans. A primer on bézier curves, 2011–2020. URL <https://pomax.github.io/bezierinfo/>.
- I. I. Kaminer, A. M. S. Pascoal, E. Xargay, N. Hovakimyan, C. Cao, and V. N. Dobrokhodov. Path following for unmanned aerial vehicles using

- l1 adaptive augmentation of commercial autopilots. *Journal of Guidance Control and Dynamics*, 33:550–564, 2010.
- M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM review*, 59(4):849–904, 2017.
- S.-H. Kim, G. E. G. Padilla, K.-J. Kim, and K.-H. Yu. Flight path planning for a solar powered uav in wind fields using direct collocation. *IEEE Transactions on Aerospace and Electronic Systems*, 56:1094–1105, 2020.
- D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl. Nmpc for racing using a singularity-free path-parametric model with obstacle avoidance. *IFAC-PapersOnLine*, 53(2):14324–14329, 2020. ISSN 2405-8963. doi: <https://doi.org/10.1016/j.ifacol.2020.12.1376>. URL <https://www.sciencedirect.com/science/article/pii/S2405896320317845>. 21st IFAC World Congress.
- M. Kobitzsch, M. Radermacher, and D. Schieferdecker. Evolution and evaluation of the penalty method for alternative graphs. In *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (2013)*, pages 94–107. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013.
- D. Kouzoupis, I. Pendharkar, J. Frey, M. Diehl, and F. Corman. Direct multiple shooting for computationally efficient train trajectory optimization. *Transportation Research Part C: Emerging Technologies*, 152:104170, 2023. ISSN 0968-090X. doi: <https://doi.org/10.1016/j.trc.2023.104170>.
- P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme. An experimental review on deep learning architectures for time series forecasting. *International journal of neural systems*, 31(03):2130001, 2021.
- J.-T. Li, C.-K. Chen, and H. Ren. Time-optimal trajectory planning and tracking for autonomous vehicles. *Sensors*, 24(11), 2024. ISSN 1424-8220. doi: 10.3390/s24113281. URL <https://www.mdpi.com/1424-8220/24/11/3281>.

- Q. Li, C. Y. Yiu, C. Simon, and K. K. Ng. Situational awareness and flight approach phase event recognition based on psychophysiological measurements. In *2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1308–1312. IEEE, 2021.
- Y. Lim, A. Gardi, R. Sabatini, K. Ranasinghe, N. Ezer, K. Rodgers, and D. Salluce. Optimal energy-based 4d guidance and control for terminal descent operations. *Aerospace Science and Technology*, 95:105436, 2019.
- J. Lin, H. Zhu, and J. Alonso-Mora. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2682–2688, 2020. doi: 10.1109/ICRA40945.2020.9197481.
- H. Liu, Y. P. Tsang, and C. Lee. A cyber-physical social system for autonomous drone trajectory planning in last-mile superchilling delivery. *Transportation Research Part C: Emerging Technologies*, 2024.
- S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. R. Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2:1688–1695, 2017.
- I. Lluvia, E. Lazkano, and A. Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445, 2021.
- W. Luo, J. Chen, H. Ebel, and P. Eberhard. Time-optimal handover trajectory planning for aerial manipulators based on discrete mechanics and complementarity constraints. *IEEE Transactions on Robotics*, 39:4332–4349, 2022.
- T. Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo 22.22*, 2011.
- Z. Ma and J. Chen. Adaptive path planning method for uavs in complex environments. *International Journal of Applied Earth Observation and Geoinformation*, 115:103133, 2022.

- J. A. Marshall, W. Sun, and A. L’Afflitto. A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. *Annual Reviews in control*, 52:390–427, 2021. ISSN 1367-5788.
- D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6): 789–814, 2000.
- S. McCammon, D. Jones, and G. A. Hollinger. Topology-Aware Self-Organizing Maps for Robotic Information Gathering. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1717–1724, 10 2020. doi: 10.1109/IROS45743.2020.9341040. ISSN: 2153-0866.
- L. Meier, D. Honegger, and M. Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, 2015. doi: 10.1109/ICRA.2015.7140074.
- D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.
- J. Men and J. Requena Carrión. A Generalization of the CHOMP Algorithm for UAV Collision-Free Trajectory Generation in Unknown Dynamic Environments. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 96–101, 11 2020. doi: 10.1109/SSRR50563.2020.9292599. ISSN: 2475-8426.
- I. Mir, F. Gul, S. Mir, M. A. Khan, N. Saeed, L. Abualigah, B. Abuhaija, and A. H. Gandomi. A Survey of Trajectory Planning Techniques for Autonomous Systems. *Electronics*, 11(18):2801, 1 2022. ISSN 2079-9292. doi: 10.3390/electronics11182801. number: 18 publisher: Multidisciplinary Digital Publishing Institute.

- M. Moshref-Javadi and M. Winkenbach. Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, 177:114854, 2021.
- M. C. R. Murça. Identification and prediction of urban airspace availability for emerging air mobility operations. *Transportation Research Part C: Emerging Technologies*, 131:103274, 2021. ISSN 0968-090X. doi: <https://doi.org/10.1016/j.trc.2021.103274>. URL <https://www.sciencedirect.com/science/article/pii/S0968090X21002862>.
- A. Nash, S. Koenig, and C. Tovey. Lazy theta*: Any-angle path planning and path length analysis in 3d. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 147–154, 2010.
- J. Patrikar, V. Dugar, V. Arcot, and S. Scherer. Real-time Motion Planning of Curvature Continuous Trajectories for Urban UAV Operations in Wind. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 854–861, 9 2020. doi: 10.1109/ICUAS48674.2020.9213837. ISSN: 2575-7296.
- Y. V. Pehlivanoglu. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. *Aerospace Science and Technology*, 16(1):47–55, 2012.
- P. Pradeep, T. A. Lauderdale, G. B. Chatterji, K. Sheth, C. F. Lai, B. Sridhar, K.-M. Edholm, and H. Erzberger. Wind-optimal trajectories for multirotor evtol aircraft on uam missions. In *Aiaa Aviation 2020 Forum*, page 3271, 2020.
- Z. Qadir, F. Ullah, H. S. Munawar, and F. Al-Turjman. Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review. *Computer Communications*, 168:114–135, 2021.
- M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009a.

- M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source Robot Operating System. *ICRA Workshop on Open Source Software*, 2009b.
- S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2, 5 1993. doi: 10.1109/ROBOT.1993.291936.
- M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: <https://doi.org/10.1016/j.jcp.2018.10.045>.
- A. M. Raivi, S. A. Huda, M. M. Alam, and S. Moh. Drone routing for drone-based delivery systems: A review of trajectory planning, charging, and security. *Sensors*, 23(3):1463, 2023.
- J. C. H. Ramírez and M. Nahon. A gravity-referenced moving frame for vehicle path following applications in 3d. *IEEE Robotics and Automation Letters*, 6(3):4393–4400, 2021.
- A. Rao. A Survey of Numerical Methods for Optimal Control. *Advances in the Astronautical Sciences*, 135, jan 1 2010.
- J. Rao et al. Path planning for dual uavs cooperative suspension transport based on artificial potential field-a* algorithm. *Knowledge-Based Systems*, 277:110797, 2023. doi: <https://doi.org/10.1016/j.knosys.2023.110797>.
- N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, Kobe, 5 2009. IEEE. ISBN 978-1-4244-2788-8. doi: 10.1109/ROBOT.2009.5152817. URL <http://ieeexplore.ieee.org/document/5152817/>. [Online; accessed 2023-05-12].

- R. Reiter, A. Nurkanović, J. Frey, and M. Diehl. Frenet-cartesian model representations for automotive obstacle avoidance within nonlinear mpc. *Eur. J. Control*, 74:100847, 2022.
- Y. Ren, F. Zhu, W. Liu, Z. Wang, Y. Lin, F. Gao, and F. Zhang. Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6332–6339, 2022. doi: 10.1109/IROS47612.2022.9981518.
- H. Ritchie, V. Samborska, and M. Roser. Urbanization. *Our world in data*, 2024.
- C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6, 5 2012.
- S. Roga, S. Bardhan, Y. Kumar, and S. K. Dubey. Recent technology and challenges of wind energy generation: A review. *Sustainable Energy Technologies and Assessments*, 52:102239, 2022.
- A. Romero, R. Pěnička, and D. Scaramuzza. Time-optimal online replanning for agile quadrotor flight. *IEEE Robotics and Automation Letters*, PP:1–8, 2022a.
- A. Romero, S. Sun, P. Foehn, and D. Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022b.
- C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143, 9 2013. doi: 10.1109/ECMR.2013.6698833.
- C. Rösmann, F. Hoffmann, and T. Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and*

- Autonomous Systems*, 88:142–153, feb 1 2017a. ISSN 0921-8890. doi: 10.1016/j.robot.2016.11.007.
- C. Rösmann, F. Hoffmann, and T. Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686, 9 2017b. doi: 10.1109/IROS.2017.8206458. ISSN: 2153-0866.
- C. Rosmann, A. Makarow, F. Hoffmann, and T. Bertram. Time-optimal nonlinear model predictive control with minimal control interventions. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 19–24, 2017. doi: 10.1109/CCTA.2017.8062434.
- T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.
- M. Sarvaiya, G. Li, and G. Loianno. Hpa-mpc: Hybrid perception-aware nonlinear model predictive control for quadrotors with suspended loads. *IEEE Robotics and Automation Letters*, 10(1):358–365, 2025. doi: 10.1109/LRA.2024.3505816.
- A. Saviolo, G. Li, and G. Loianno. Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking. *IEEE Robotics and Automation Letters*, 7(4):10256–10263, 2022. doi: 10.1109/LRA.2022.3192609.
- D. Shapovalov and G. A. Pereira. Tangle-free exploration with a tethered mobile robot. *Remote Sensing*, 12(23):3858, 2020.
- G. Shi, X. Shi, M. O’Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pages 9784–9790. IEEE, 2019.

- V. Silva, A. Amaral, and T. Fontes. Sustainable urban last-mile logistics: A systematic literature review. *Sustainability*, 15(3):2285, 2023.
- Y. Song, B. Pan, Q. Fan, and B. Xu. A computationally efficient sequential convex programming using chebyshev collocation method. *Aerospace Science and Technology*, 141:108584, 2023.
- S. Spedicato and G. Notarstefano. Minimum-time trajectory generation for quadrotors in constrained environments. *IEEE Transactions on Control Systems Technology*, 26(4):1335–1344, 2017.
- M. D. Spivak. A comprehensive introduction to differential geometry. (*No Title*), 1970.
- B. Sridhar, K. S. Sheth, and S. Grabbe. Airspace complexity and its application in air traffic management. In *2nd USA/Europe Air Traffic Management R&D Seminar*, pages 1–6. Federal Aviation Administration Washington, DC, USA, 1998.
- B. Sridhar, H. K. Ng, F. Linke, and N. Y. Chen. Benefits analysis of wind-optimal operations for trans-atlantic flights. In *14th AIAA Aviation Technology, Integration, and Operations Conference*, page 2583, 2014.
- J. Sunghun and K. Hyunsu. Analysis of amazon prime air uav delivery service. *Journal of Knowledge Information Technology and Systems*, 12: 253–266, 2017.
- P. D. Team. *PX4 Autopilot User Guide: Software In the Loop (SITL) Simulation*, 2024. <https://docs.px4.io/main/en/simulation/>.
- L. Techy and C. A. Woolsey. Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of guidance, control, and dynamics*, 32(6):1736–1746, 2009.
- G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.

- J. P. Udani, K. Mall, M. J. Grant, and D. Sun. Optimal flight trajectory to minimize noise during landing. In *55th AIAA Aerospace Sciences Meeting*, page 1180, 2017.
- A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.
- R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE Conference on Decision and Control*, pages 2505–2510, 2014. doi: 10.1109/CDC.2014.7039771.
- R. Verschueren, N. van Duijkeren, J. Swevers, and M. Diehl. Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation. In *2016 American Control Conference (ACC)*, pages 2092–2097. IEEE, 2016.
- R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.
- D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho. A survey on load transportation using multicopter uavs. *Journal of Intelligent & Robotic Systems*, 98(2):267–296, 2020. doi: 10.1007/s10846-019-01088-w.
- R. Vinter and R. Vinter. The maximum principle. *Optimal Control*, pages 201–231, 2010.
- H. Wang, J. Hu, Y. Feng, and X. Li. Optimal control-based highway pilot motion planner with stochastic traffic consideration. *IEEE Intelligent Transportation Systems Magazine*, 15:421–436, 2023.
- H. Wang, H. Li, B. Zhou, F. Gao, and S. Shen. Impact-aware planning and control for aerial robots with suspended payloads. *IEEE Transactions on Robotics*, 40:2478–2497, 2024. doi: 10.1109/TRO.2024.3381555.

- Z. Wang, D. Delahaye, J.-L. Farges, and S. Alam. Complexity optimal air traffic assignment in multi-layer transport network for urban air mobility operations. *Transportation Research Part C: Emerging Technologies*, 142: 103776, 2022.
- M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. *2010 IEEE International Conference on Robotics and Automation*, pages 987–993, 2010.
- M. Werner and S. Feld. Homotopy and alternative routes in indoor navigation scenarios. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 230–238, 10 2014. doi: 10.1109/IPIN.2014.7275488.
- J. Wu, X. Ma, T. Peng, and H. Wang. An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environment. *Sensors (Basel, Switzerland)*, 21(24):8312, dec 12 2021. ISSN 1424-8220. doi: 10.3390/s21248312. PMID: 34960406 PMCID: PMC8707625.
- Y. Wu, I. Spasojevic, P. Chaudhari, and V. Kumar. Towards optimizing a convex cover of collision-free space for trajectory generation. *IEEE Robotics and Automation Letters*, 10(5):4762–4769, 2025. doi: 10.1109/LRA.2025.3553416.
- K. Xiao, S. Tan, G. Wang, X. An, X. Wang, and X. Wang. Xtdrone: A Customizable Multi-rotor UAVs Simulation Platform. In *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pages 55–61, Wuhan, China, 6 2020. IEEE. ISBN 978-1-72819-857-6. doi: 10.1109/ICRAS49812.2020.9134922. URL <https://ieeexplore.ieee.org/document/9134922/>. [Online; accessed 2023-02-13].
- H. S. Yahia and A. S. Mohammed. Path planning optimization in unmanned aerial vehicles using meta-heuristic algorithms: A systematic review. *Environmental Monitoring and Assessment*, 195(1):30, 2023. ISSN 0167-6369.

- D. Yi, M. A. Goodrich, and K. D. Seppi. Homotopy-aware rrt*: Toward human-robot topological path-planning. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 279–286. IEEE, 2016.
- H. Yu, Z. Zhang, T. Pei, J. Han, Y. Fang, and X. Liang. Visual servoing-based anti-swing control of cable-suspended aerial transportation systems with variable-length cable. *IEEE Transactions on Automation Science and Engineering*, pages 1–11, 2024. doi: 10.1109/TASE.2024.3434637.
- W. Yu. A new concept using lstm neural networks for dynamic system identification. In *2017 American Control Conference (ACC)*, pages 5324–5329, 2017. doi: 10.23919/ACC.2017.7963782.
- Y. Zeng, J. Xu, and R. Zhang. Energy Minimization for Wireless Communication With Rotary-Wing UAV. *IEEE Transactions on Wireless Communications*, 18(4):2329–2345, 4 2019. ISSN 1558-2248. doi: 10.1109/TWC.2019.2902559. number: 4 event-title: IEEE Transactions on Wireless Communications.
- C. Zhang, H. Liu, and Y. Tang. Quantitative evaluation of voronoi graph search algorithm in uav path planning. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 563–567. IEEE, 2018.
- D. Zhang, Z. Xuan, Y. Zhang, J. Yao, X. Li, and X. Li. Path Planning of Unmanned Aerial Vehicle in Complex Environments Based on State-Detection Twin Delayed Deep Deterministic Policy Gradient. *Machines*, 11(1):108, 1 2023a. ISSN 2075-1702. doi: 10.3390/machines11010108. number: 1 publisher: Multidisciplinary Digital Publishing Institute.
- N. Zhang, M. Zhang, and K. H. Low. 3d path planning and real-time collision resolution of multirotor drone operations in complex urban low-altitude airspace. *Transportation Research Part C-emerging Technologies*, 129: 103123, 2021.

- Q. Zhang, Y. Gao, S. Meng, H. Dong, C. Ru, and W. Dong. A novel contouring control method based on optimal vector-referenced moving frame for 3d trajectory with zero curvature. *IEEE Transactions on Automation Science and Engineering*, 22:7798–7809, 2025. doi: 10.1109/TASE.2024.3470799.
- Z. Zhang, X. Liu, and B. Feng. Research on obstacle avoidance path planning of uav in complex environments based on improved b ezier curve. *Scientific Reports*, 13(1):16453, 2023b.
- Z. Zhang, D. Zhang, D. Kong, and Y. Wang. Real-time local obstacle avoidance and trajectory tracking control of quadrotor uavs with suspended payload in complex environments. *IEEE Access*, 11:144017–144029, 2023c.
- 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*, 4(4):3529–3536, 10 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2927938. event-title: IEEE Robotics and Automation Letters.
- B. Zhou, F. Gao, J. Pan, and S. Shen. Robust real-time uav replanning using guided gradient-based optimization and topological paths. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1208–1214. IEEE, 2020.
- Y. Zhou, L. Yan, Y. Han, H. Xie, and Y. Zhao. A survey on the key technologies of uav motion planning. *Drones*, 9(3), 2025. ISSN 2504-446X. doi: 10.3390/drones9030194.