공학박사학위논문

Trajectory generation and control of multi-rotors with a suspended load using nonlinear optimization

비선형 최적화를 이용한 멀티로터 현수 운송의 경로 계획 및 제어 기법

2021년 8월

서울대학교 대학원

기계항공공학부

손 영 동

Trajectory generation and control of multi-rotors with a suspended load using nonlinear optimization

비선형 최적화를 이용한 멀티로터 현수 운송의 경로 계획 및 제어 기법

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2021년 6월

서울대학교 대학원

기계항공공학부

손 영 동

손영동의 공학박사 학위논문을 인준함

2021년 7월

위 원 장 : ____________________

부위원장 : ____________________

위　　원 : ____________________

위　　원 : ____________________

위　　원 : ____________________

Trajectory generation and control of multi-rotors with a suspended load using nonlinear optimization

A Dissertation

by

SON Youngdong

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2021

Trajectory generation and control of multi-rotors with a suspended load using nonlinear optimization

YOUNGDONG SON

Department of Mechanical and Aerospace Engineering

Seoul National University

to my

GRANDMOTHER, FATHER, MOTHER, BROTHER, and NAKYEONG

with love

Abstract

Trajectory generation and control of multi-rotors with a suspended load using nonlinear optimization

SON Youngdong
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

Trajectory generation and control are fundamental requirements for safe and stable operation of multi-rotors. The dynamic model should be considered to generate efficient and collision-free trajectories with feasibility. While the dynamic model of a bare multi-rotor is expressed non-linearly with high dimensions which results in computational loads, the suspended load increases the complexity further. This dissertation presents efficient algorithms for trajectory generation and control of multi-rotors with a suspended load.

A single multi-rotor with a suspended load is addressed first. Since the load is suspended through a cable without any actuator, movement of the load must be controlled via maneuvers of the multi-rotor. However, the highly non-linear dynamics of the system results in difficulties. To relive them, the rotational dynamics is simplified to reduce the non-linearity and consider the delay in attitude control. For trajectory generation, the vehicle, cable, and load are considered as ellipsoids with different sizes and shapes, and collision-free constraints are expressed in an efficient and less-conservative way. The augmented Lagrangian method is applied to solve a nonlinear optimization problem with nonlinear constraints in real-time. Model predictive control with the sequential linear quadratic solver is used to track the generated trajectories. The proposed algorithm is validated with several simulations and experiment.

A system with multiple multi-rotors for cooperative transportation of a suspended load is addressed next. As the system has more state variables and coupling terms in the dynamic equation than the system with a single multi-rotor, optimization takes a long time without an efficient method. The differential flatness of the system is used to reduce the complexity of the highly non-linear dynamic equation. The trajectories are also parameterized using piece-wise Bernstein polynomials to decrease the number of optimization variables. By decomposing an optimization problem and performing convexification, convex sub-problems are formulated for the load and the tension trajectories optimization, respectively. In each sub-problem, a light-weight sampling method is used to find a feasible and low-cost trajectory as initialization. In the first sub-problem, the load trajectory is optimized with safe flight corridor (SFC) and clearance constraints for collision avoidance and security of space for the multi-rotors. Then, the tension histories are optimized with safe flight sector (SFS) and relative safe flight sector (RSFS) constraints for obstacle and inter-agent collision avoidance. Simulations and experiments are conducted to demonstrate efficient trajectory generation in a cluttered environment and validate the proposed algorithms.

Keywords: Aerial manipulation, Motion and path planning, Optimization and optimal control, Multi-agent.

Student Number: 2015-22735

Table of Contents

List of Tables

List of Figures

1 Introduction

Unmanned aerial vehicles (UAVs) have been widely used in various areas such as photography, surveillance, inspection, farming, and transportation with the advancements of high computing power and efficient algorithms. Among the areas, many freight delivery companies have been actively investigating aerial transportation since it can substantially reduce the time consumed in land transportation.

Various manipulators have been attached to UAVs to expand the coverage of the application in aerial transportation. For instance, a robotic gripper is equipped, and the vehicle gains an ability to grasp or release variously shaped objects [1]. A robotic arm with multi degrees of freedom is used to pick or push an object [2, 3]. While the two mentioned types require actuators to control the manipulators actively, only a cable is required for the suspension type. Some studies taking into account the movement of both the suspended load and the multi-rotor carry the load safely [4], [5], [6].

The system composed of multi-rotors and an object suspended by a cable, also called a slung load system, has various advantages in transportation. Since no additional actuators are required, there is little increase in the total mass of the system, and users can utilize the maximum payload. Moreover, by attaching one end of the cable close to the center of the mass of the vehicle, the attitude dynamics of the multi-rotor is scarcely affected.

Although the slung load system has several merits in transportation, it is very difficult to control the overall system stably. The difficulty comes from under-actuation and non-linearity of the system. Multi-rotor is originally an under-actuated system with twelve states and four inputs [7], and it should tilt itself to generate horizontal acceleration. The slung load system has six more states for each multi-rotor due to the cable while no input is added [8], which increases the level of under-actuation and non-linearity from the dynamic coupling between the vehicles and the suspended load.

Stabilizing controllers and efficient trajectory generation algorithms are required to operate this under-actuated nonlinear system. Efficient trajectory generation algorithms enable the vehicle and the load to fly in cluttered environments while avoiding obstacles. Meanwhile, stabilizing controllers need to stabilize the swing of the suspended object while tracking the desired trajectory.

The first aimed system is composed of a single multi-rotor and a suspended load which is depicted in Fig. 1.1(a). To compensate the attitude control delay and reduce complexity of the dynamic model, an original attitude dynamics is simplified. After conducting simple trajectory tracking experiments, model identification is performed with two candidates: a time-delay model and a first-order model. The more exact attitude dynamics is used for the trajectory generation and control. By enclosing the multi-rotor, cable, and load with three ellipsoids, collision avoidance for obstacles is considered in trajectory generation optimization as done in [9]. For trajectory tracking control, model predictive control is used to consider full states and generate optimal control inputs in real-time.

The second targeted model, the multiple multi-rotors with a suspended load, has a higher-dimensional dynamic model as shown in Fig. 1.1(b). Therefore, the proposed methods for a single multi-rotor cannot be directly applied. To reduce the computational load, the differential flatness property is used to formulate an optimization problem with the smaller number of variables called flat output which is composed of the load position and cable tension vectors. The problem is also parameterized using piece-wise Bernstein polynomials and decomposed into two sub-problems: load trajectory generation and tension history generation. For each sub-problem, convexification methods are proposed to convexify non-convex constraints, which formulates a convex optimization sub-problem. A robust controller is implemented to track the generated trajectory in a decentralized manner.

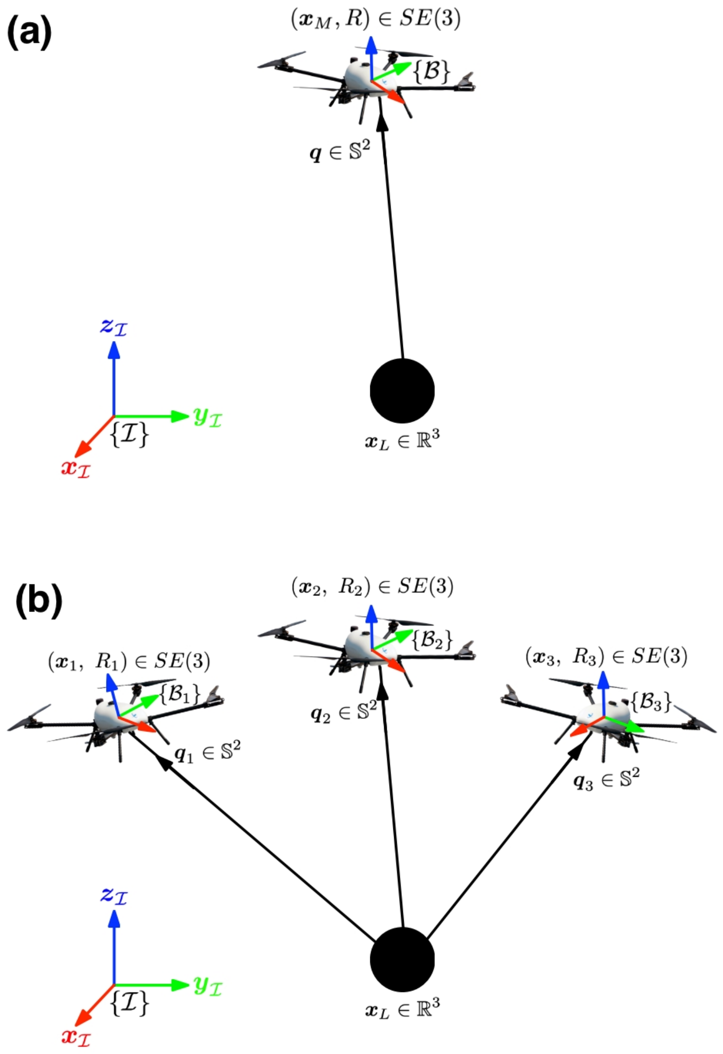

(a)
$(\boldsymbol{x}_M, R) \in SE(3)$
$\{\mathcal{B}\}$
$\boldsymbol{q} \in \mathbb{S}^2$
$\boldsymbol{z}_\mathcal{I}$
$\boldsymbol{y}_\mathcal{I}$
$\{\mathcal{I}\}$
$\boldsymbol{x}_\mathcal{I}$
$\boldsymbol{x}_L \in \mathbb{R}^3$
(b)
$(\boldsymbol{x}_2,\ R_2) \in SE(3)$
$\{\mathcal{B}_2\}$
$(\boldsymbol{x}_1,\ R_1) \in SE(3)$
$\{\mathcal{B}_1\}$
$(\boldsymbol{x}_3,\ R_3) \in SE(3)$
$\{\mathcal{B}_3\}$
$\boldsymbol{q}_2 \in \mathbb{S}^2$
$\boldsymbol{q}_1 \in \mathbb{S}^2$
$\boldsymbol{q}_3 \in \mathbb{S}^2$
$\boldsymbol{z}_\mathcal{I}$
$\boldsymbol{y}_\mathcal{I}$
$\{\mathcal{I}\}$
$\boldsymbol{x}_\mathcal{I}$
$\boldsymbol{x}_L \in \mathbb{R}^3$

Figure 1.1: Coordinate frames and state variables of the systems. (a) A single multi-rotor with a suspended load. (b) Multiple multi-rotors with a suspended load.

The respective flow charts of the proposed algorithms for the two targeted systems are shown in Fig. 1.2.

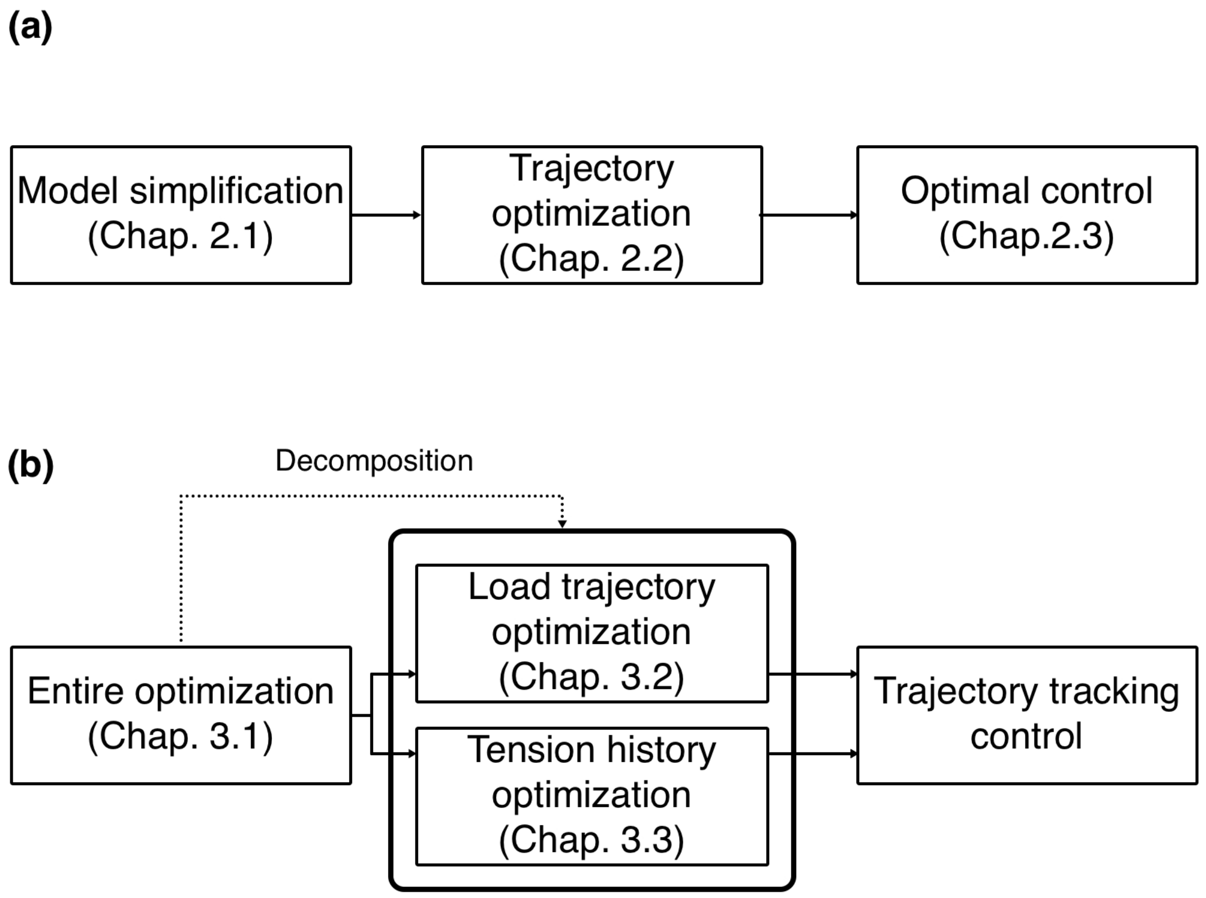

(a)
Model simplification (Chap. 2.1)
Trajectory optimization (Chap. 2.2)
Optimal control (Chap.2.3)
(b)
Decomposition
Entire optimization (Chap. 3.1)
Load trajectory optimization (Chap. 3.2)
Tension history optimization (Chap. 3.3)
Trajectory tracking control

Figure 1.2: Flow charts of the proposed algorithms for the slung load systems. (a) A single multi-rotor with a suspended load. (b) Multiple multi-rotors with a suspended load.

1.1 Literature Survey

There exists an extensive body of literature in operating the slung load systems. In this dissertation, the related works are introduced into two categories: single vehicle and multiple vehicles with a suspended load. Each category is once more divided into trajectory generation and control.

1.1.1 Single Vehicle with a Suspended Load

Trajectory Generation

There exist many works for trajectory generation of a bare multi-rotor [10, 11, 12, 13, 14]. Meanwhile, due to the complexity of the slung load system, relatively smaller number of algorithms have been proposed for transportation of a suspended load. In [15], mixed integer quadratic programming is used to generate narrow-gap-passing trajectories. Another work demonstrates load throwing and obstacle avoidance maneuvers using mathematical program with complementarity constraints along with quadratic programming [16]. While the referred works show good performance, it is not possible to generate collision-free trajectories in real-time. In [5], the authors show slalom maneuvers with real-time trajectory generation. Still, the work has a limitation in that only the suspended load is considered for collision avoidance.

Control

Various controllers are proposed to control the slung load system. They can be classified into three types depending on the objective, and the first type considers tension from the load as an impediment to control of the multi-rotor and focuses on reducing swing of the load. Input shaping and delayed feedback control are used to actively attenuate swing [17], [18]. An H_∞ controller with Lyapunov redesign technique is implemented in [19] to track the desired trajectory of the multi-rotor with swing damping. While the proposed controllers are proven to be stable since their focus is control of the multi-rotor, the load cannot be controlled but only the swing can be attenuated.

The second approach aims to control the passively suspended load through actuation of the multi-rotor. A geometric controller is proposed and performance of the controller is validated through various experiments [6], [8]. While desired trajectories must be a parametric curve of class at least C^5 for stability, the proposed controller in [20] shows little error in tracking. [21] controls the system using the nonlinear dynamic inversion to employ standard linear controllers. However, the referred controllers are designed with a cascade structure where the desired cable angle is computed from the desired load position, and the desired multi-rotor attitude is computed from the desired cable angle. Since the cascade structure cannot control both the vehicle and the load simultaneously, directly applying it for obstacle avoidance is not appropriate.

The last approach is optimal control which can control both the multi-rotor and load by optimization. Since the system has many state variables for the general nonlinear optimization solvers to optimize in real-time, dynamic programming approach has been mainly used. A linear quadratic regulator and a model predictive control (MPC) are compared in [22] with stabilization and tracking simulations. In [23], MPC is used for load tracking experiments and it shows good performance. Recently, obstacle avoidance is successfully conducted by following the generated collision-free trajectories [4, 5].

1.1.2 Multiple Vehicles with a Suspended Load

Trajectory Generation

Differential flatness is derived and utilized to generate trajectories for the system with a point mass or a rigid body in [24]. However, the generated trajectory is a simple sinusoidal one without consideration of obstacle avoidance. A dynamic model with flexible cables is derived in [25] with which cable slackness can be considered. In this dissertation, the slackness needs not to be considered since the cable tautness is maintained with constraints. To avoid inter-agent collision between the vehicles, [26] proposes a cone constraint which limits the position of the vehicle in the designed cone. A recent work [27] proposes a distributed optimization algorithm that solves multi-rotor trajectories in parallel and updates a load trajectory with them. The work shows near-real-time performance for obstacle avoidance.

Control

A linear quadratic regulator and leader-follower control scheme are used for collaborative transportation of two multi-rotors [28]. [29] uses robust model predictive control and conducts numerical simulations where four multi-rotors follow a reference trajectory. [30] proposes a nonlinear geometric controller with a stability proof which can be applied to any number of vehicles.

1.1.3 Feature of the Work

The most studies about trajectory generation for the single multi-rotor with a load cannot generate trajectories in real-time. While some works show real-time performance, they cannot guarantee collision avoidance of the entire components, i.e., the vehicle, cable, and load. The first two categories of the works related to control have difficulties in tracking the desired trajectory since they consider the load as a disturbance or use a cascaded structure.

The proposed work addresses both the trajectory generation and control problems. To avoid obstacles, all the components of the system are considered in a less-conservative way by encompassing them with three ellipsoids. By formulating a receding horizon constrained optimization, real-time trajectory generation is possible. To track the generated trajectory precisely, an optimal control algorithm is proposed which can control the vehicle, cable, and load at the same time.

In the second targeted system, since the dimension increases proportionally to the number of the vehicles which makes designing efficient algorithms more difficult, there are relatively fewer studies than the single vehicle slung load system. Since the position of the suspended load is passively decided by the positions of at least three vehicles, controller design is a relatively lighter topic. For safe trajectory generation, not only the obstacle avoidance of the load, but also the inter-agent collision avoidance between the vehicles and obstacle avoidance of each vehicle should be considered. While [27] addresses the constraints, only cylindrical obstacles are addressed and complex environments cannot be easily considered.

To generate safe trajectories efficiently, the proposed work decomposes a big problem into two sub-problems. For each sub-problem, the non-convex constraints are convexified to generate convex safe regions. Then, the convex sub-problems can be solved by any convex optimization solvers. For control of the system, a robust controller proposed in [31] is used to construct a decentralized control system where each vehicle follows the respective trajectory compensating the tension from the cable actively without communication.

1.2 Contributions

The contributions of this dissertation are summarized as follows.

Single multi-rotor with a suspended load

- **Practical dynamic model**: A simplified but practical dynamic model is proposed with the consideration of the control delay. The considered delay reduces the control error, and the reduced size of the dynamic equation helps to decrease the computational load required for trajectory generation and optimal control.

- **Real-time trajectory generation**: The proposed trajectory generation and optimal control algorithms can run on onboard computer in real-time. This makes it possible to re-generate new trajectories when new obstacles appear and to update the control inputs rapidly.

Multiple multi-rotors with a suspended load

- **Convex optimization formulation**: The non-convex problem can be solved with the proposed two convex optimization sub-problems where many solvers can be used.

- **Independence of environment complexity**: The proposed trajectory generation algorithm considers obstacles using a single map. The number of obstacles does not necessarily increase the number of collision avoidance constraints, which maintains the order of the computational load even in a complex environment.

1.3 Outline

The outline of the dissertation is as follows. Chapter 2 addresses trajectory generation and optimal control methods for the system composed of a single multi-rotor with a suspended load. Chapter 3 presents trajectory generation methods for the system of multiple multi-rotors with a load. For validation, Chapter 4 provides experimental results, and Chapter 5 ends the dissertation with concluding remarks.

2 Single Multi-rotor with a Suspended Load

This chapter addresses overall algorithms of trajectory generation and optimal control to operate a single multi-rotor with a suspended load, with the derivation of a practical dynamic model. Refer to Fig. 1.2(a) for the flow chart of the chapter.

2.1 Dynamics

Various dynamic models are proposed for the slung load system. [32] derives a dynamic model using the Udwadia-Kalaba equation with the consideration of rotor downwash. Assuming that one end of the cable is attached to the center of mass of the multi-rotor, [8] derives a dynamic model using the method of Lagrange while another model is derived using the Newtonian method [33]. The assumption is used also in this dissertation. The dynamic model proposed in [33] is expressed as

$$\begin{aligned}
&\frac{d}{dt}\boldsymbol{x}_L = \boldsymbol{v}_L,\\
&\frac{d}{dt}\boldsymbol{v}_L = \frac{1}{m_M + m_L}\big(fR\boldsymbol{e}_3 - m_M\ddot{\boldsymbol{q}}\big) - g\boldsymbol{e}_3,\\
&\frac{d}{dt}\boldsymbol{q} = \boldsymbol{\omega} \times \boldsymbol{q},\\
&\frac{d}{dt}\boldsymbol{\omega} = \frac{1}{m_M l^2}\boldsymbol{q} \times fR\boldsymbol{e}_3,\\
&\frac{d}{dt}R = R\hat{\boldsymbol{\Omega}}_{\mathcal{B}},\\
&\frac{d}{dt}\boldsymbol{\Omega}_{\mathcal{B}} = J_M^{-1}(\boldsymbol{M} - \boldsymbol{\Omega}_{\mathcal{B}} \times J_M\boldsymbol{\Omega}_{\mathcal{B}}),
\end{aligned} \tag{2.1}$$

with a control input $\boldsymbol{u} = \begin{bmatrix} f & \boldsymbol{M}^\top \end{bmatrix}^\top \in \mathbb{R}^4$ and a rotation matrix $R = R_z(\psi)R_y(\theta)R_x(\phi)$. Note that the dynamic model assumes the cable tautness, and the assumption is validated through the experiment results in Fig. 4.5 and Fig. 4.10. Nomenclature for the system is listed in Table 2.1.

Multi-rotor generates thrust and moments by making differences in the thrust of each motor. Although the dynamic model in (2.1) describes the coupling between the vehicle and the load well, there exists a limitation to be used with optimization-based controllers. While direct speed control of BLDC motors is possible to generate desired thrust [34, 35], precise attitude control is difficult since measurement of the moment of inertia requires additional processes. To overcome the limitation in the attitude control, other research uses angular velocity, $\boldsymbol{w}$, as a control input assuming that the angular velocity control is immediate [23]. Not only be the simplified dynamics more reasonable with a stable low-level controller, but it also reduces the size of the state variable. However, such dynamic models require very fast update of the input, and the assumption of delay-less angular velocity control is too harsh. Accordingly, such dynamic model is not appropriate for optimization-based control which requires longer computation time.

Two kinds of simplified attitude dynamics are proposed and compared in this disser-

Table 2.1: Nomenclature of Chapter 2

Notation	Meaning
$\mathcal{I}$	Inertial frame
$\mathcal{B}$	Body-fixed frame of the multi-rotor
$\boldsymbol{x}_{\mathcal{I}},\ \boldsymbol{y}_{\mathcal{I}},\ \boldsymbol{z}_{\mathcal{I}} \in \mathbb{R}^3$	x, y, z axes of $\mathcal{I}$
$\boldsymbol{x}_{\mathcal{B}},\ \boldsymbol{y}_{\mathcal{B}},\ \boldsymbol{z}_{\mathcal{B}} \in \mathbb{R}^3$	x, y, z axes of $\mathcal{B}$
$\boldsymbol{x}_M, \boldsymbol{x}_L \in \mathbb{R}^3$	Position of the multi-rotor and the load expressed in $\mathcal{I}$
$\boldsymbol{v}_M, \boldsymbol{v}_L \in \mathbb{R}^3$	Velocity of the multi-rotor and the load expressed in $\mathcal{I}$
$\boldsymbol{q} \in \mathbb{S}^2$	Position of the multi-rotor with respect to the load expressed in $\mathcal{I}$
$R \in SO(3)$	Rotation matrix of the multi-rotor from $\mathcal{B}$ to $\mathcal{I}$
$\boldsymbol{\Phi} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top \in \mathbb{R}^3$	Roll, pitch, and yaw angle of the multi-rotor
$\boldsymbol{\Omega}_{\mathcal{B}} = \begin{bmatrix}p & q & r\end{bmatrix}^\top \in \mathbb{R}^3$	Angular velocity of the multi-rotor expressed in $\mathcal{B}$
$\boldsymbol{\omega} \in \mathbb{R}^3$	Angular velocity of $\boldsymbol{q}$ expressed in $\mathcal{I}$
$f \in \mathbb{R}^1$	Thrust input of the multi-rotor
$\boldsymbol{M} \in \mathbb{R}^3$	Moment input of the multi-rotor
$g \in \mathbb{R}^1$	Gravitational constant
$l \in \mathbb{R}^1$	Length of the cable
$m_M,\ m_L \in \mathbb{R}^1$	Mass of the multi-rotor and the load
$J_M \in \mathbb{R}^{3\times 3}$	Inertia matrix of the multi-rotor
$\boldsymbol{e}_i \in \mathbb{R}^3$	Unit vector whose i^{th} element is 1

tation. The first model, *time-delay model*, is simplified by considering the time-delay of attitude control explicitly, and the control input of the system is a desired force vector. The other model, *first-order model*, is simplified by assuming that the attitude control dynamics is a first-order system where the control delay is implicitly considered and control input is a desired attitude vector. Two models are constructed using system identification from experiment data, and more practical and accurate one is adopted for the trajectory planner and the controller.

2.1.1 Simplified Dynamics: Time-delay Model

Firstly, a bare multi-rotor dynamics can be expressed as follows:

$$\begin{aligned}
&\frac{d}{dt}\boldsymbol{x}_M = \boldsymbol{v}_M, \\
&\frac{d}{dt}\boldsymbol{v}_M = \frac{1}{m_M} f R\boldsymbol{e}_3 - g\boldsymbol{e}_3, \\
&\frac{d}{dt}R = R\hat{\boldsymbol{\Omega}}_{\mathcal{B}}, \\
&\frac{d}{dt}\boldsymbol{\Omega}_{\mathcal{B}} = J_M^{-1}(\boldsymbol{M} - \boldsymbol{\Omega} \times J_M \boldsymbol{\Omega}_{\mathcal{B}}).
\end{aligned} \tag{2.2}$$

If a stable attitude controller is implemented, the attitude of the vehicle can approximated as a delayed signal of the desired attitude as

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(t) \\ \theta(t) \\ \psi(t) \end{bmatrix} \approx \begin{bmatrix} \phi_{des}(t - \gamma_\phi) \\ \theta_{des}(t - \gamma_\theta) \\ \psi_{des}(t - \gamma_\psi) \end{bmatrix}, \tag{2.3}$$

where $*_{des}$ are desired values of $*$, and γ_* are non-negative time-delay. In [36], with some reasonable assumptions, it is shown that the input/output relationship of the translational acceleration can also be expressed similar to (2.3) as

$$\tilde{\ddot{\boldsymbol{x}}}_M = \begin{bmatrix} \tilde{\ddot{x}}_M(t) \\ \tilde{\ddot{y}}_M(t) \\ \tilde{\ddot{z}}_M(t) \end{bmatrix} \approx \begin{bmatrix} \tilde{\ddot{x}}_{M.des}(t-\gamma_\phi) \\ \tilde{\ddot{y}}_{M.des}(t-\gamma_\theta) \\ \tilde{\ddot{z}}_{M.des}(t) \end{bmatrix}, \tag{2.4}$$

where a pseudo-acceleration vector is defined as $\tilde{\ddot{\boldsymbol{x}}}_M = R_z^{-1}(\psi)\big(\ddot{\boldsymbol{x}}_M + g\boldsymbol{e}_3\big) = R_z^{-1}(\psi)\frac{1}{m_M} f\boldsymbol{R}\boldsymbol{e}_3$. By multiplying both sides of (2.4) by $R_z(\psi)m_M$, the input/output relationship of the force generated by the multi-rotor can be considered as a point-mass force generator with time-delay in only horizontal directions. The equation is expressed as

$$\boldsymbol{F} = \begin{bmatrix} F_x(t) \\ F_y(t) \\ F_z(t) \end{bmatrix} \approx \begin{bmatrix} F_{x.des}(t-\gamma_h) \\ F_{y.des}(t-\gamma_h) \\ F_{z.des}(t) \end{bmatrix}, \tag{2.5}$$

with the assumption that γ_ϕ and γ_θ have the same value of horizontal time-delay, γ_h. Although it is difficult to directly consider the time-delay, γ_h, in the form of a state-space equation, it can easily be modeled in a frequency domain. For example, the force relationship in x axis can be expressed as follows:

$$G(s) = \frac{F_x(s)}{F_{x.des}(s)} = e^{-\gamma_h s} \approx \frac{-s + \frac{2}{\gamma_h}}{s + \frac{2}{\gamma_h}}, \tag{2.6}$$

which is approximated using Padé approximation. Time domain expression of the equation can be expressed as

$$\begin{aligned} \dot{x}_{F_x} &= A_{F_x} x_{F_x} + B_{F_x} F_{x.des} \\ F_x &= C_{F_x} x_{F_x} + D_{F_x} F_{x.des}, \end{aligned} \tag{2.7}$$

where an additional state, x_{F_x}, is required. Accordingly, to model input/output relationship of generated force vector, only two slack variables, x_{F_x} and x_{F_y}, are required.

Since the attitude dynamics in (2.1) is independent from the translation dynamics, the new simplified dynamics for rotation can be expressed by replacing the attitude dynamics with (2.7) in both the x and y axes as follows:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{x}_{F_{xy}} &= A_{F_{xy}}\boldsymbol{x}_{F_{xy}} + B_{F_{xy}}\boldsymbol{F}_{xy.des} \\ \boldsymbol{F} &= \begin{bmatrix} C_{F_{xy}}\boldsymbol{x}_{F_{xy}} + D_{F_{xy}}\boldsymbol{F}_{xy.des} \\ F_{z.des} \end{bmatrix}, \end{aligned} \tag{2.8}$$

with a new control input $\boldsymbol{u} = \begin{bmatrix} F_{x.des} & F_{y.des} & F_{z.des} \end{bmatrix}^\top \in \mathbb{R}^3$. Note that $\boldsymbol{x}_{F_{xy}}$, $A_{F_{xy}}$, $B_{F_{xy}}$, $C_{F_{xy}}$, $D_{F_{xy}}$, and $\boldsymbol{F}_{xy.des}$ are defined to express (2.7) in both the x and y axes in a compact form. The yaw angle is not considered in this equation since it can be controlled independently.

2.1.2 Simplified Dynamics: First-order Model

Motivated by [37], the following *first-order model* approximates the input/output relationship of attitude control as a first-order system where the time-delay and the response time are considered implicitly.

$$\begin{aligned} \frac{d}{dt}\tilde{\phi} &= A_h\tilde{\phi} + B_h\tilde{\phi}_{des}, \\ \frac{d}{dt}\tilde{\theta} &= A_h\tilde{\theta} + B_h\tilde{\theta}_{des}. \end{aligned} \tag{2.9}$$

Again, the yaw angle of the vehicle is assumed to be controlled separately by a low-level controller. Note that $\tilde{*}$ mean yaw-compensated values using the following equations:

$$\begin{aligned} \tilde{R}\boldsymbol{e}_3 &= R\boldsymbol{e}_3 \\ \tilde{R} &= R_y(\tilde{\theta})R_x(\tilde{\phi}). \end{aligned} \tag{2.10}$$

2.1.3 Identification and Comparison of the Simplified Models

For identification of the two candidate dynamic models, circular trajectory flight experiments are conducted with different periods. The trajectories enable the attitude controller to send time-varying commands. The position and attitude tracking results of the multi-rotor are shown in Fig. 2.1, Fig. 2.2, and Fig. 2.3.

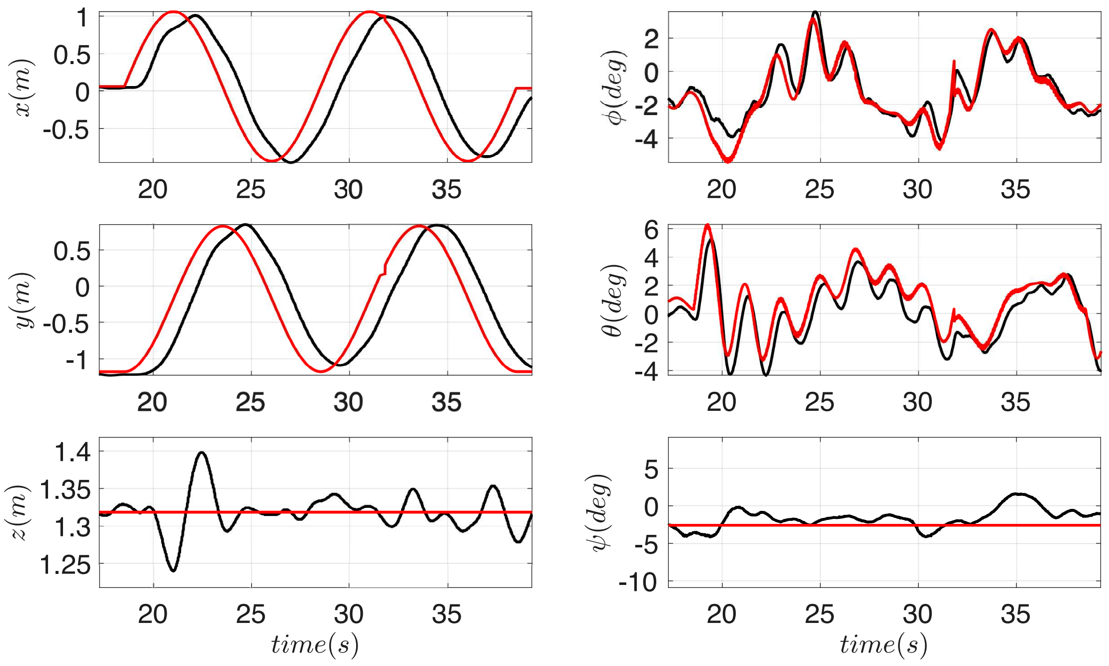


Figure 2.1: Circular trajectory (10 seconds period) tracking experiment result with (left) position and (right) attitude histories of the multi-rotor. The red line denotes desired values, and the black line denotes measurement data

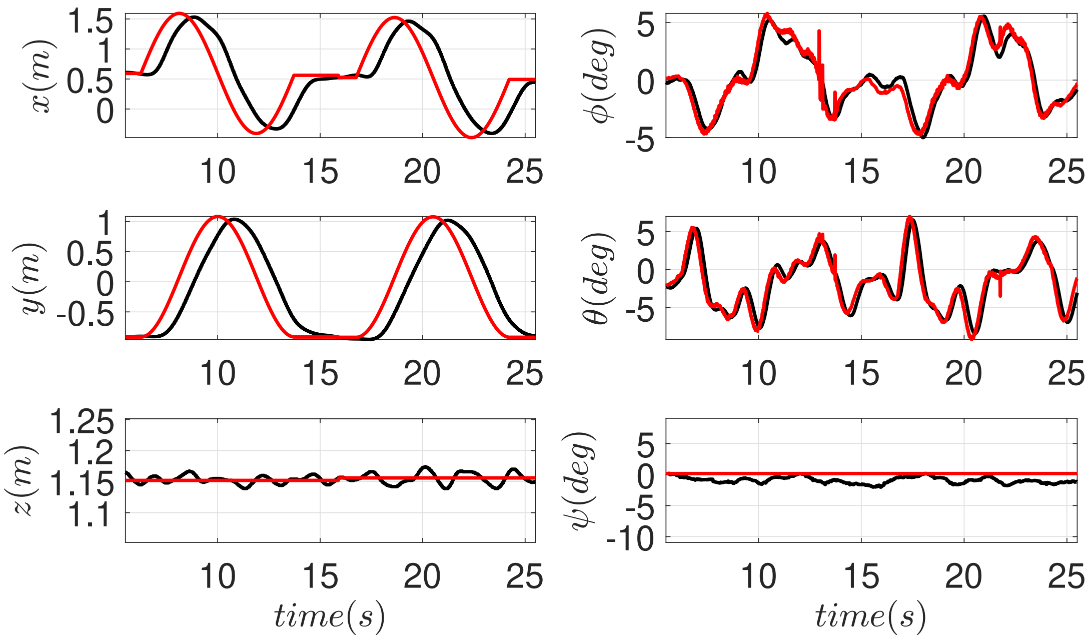


Figure 2.2: Circular trajectory (7.5 seconds period) tracking experiment result with (left) position and (right) attitude histories the a multi-rotor. The red line denotes desired values, and the black line denotes measurement data

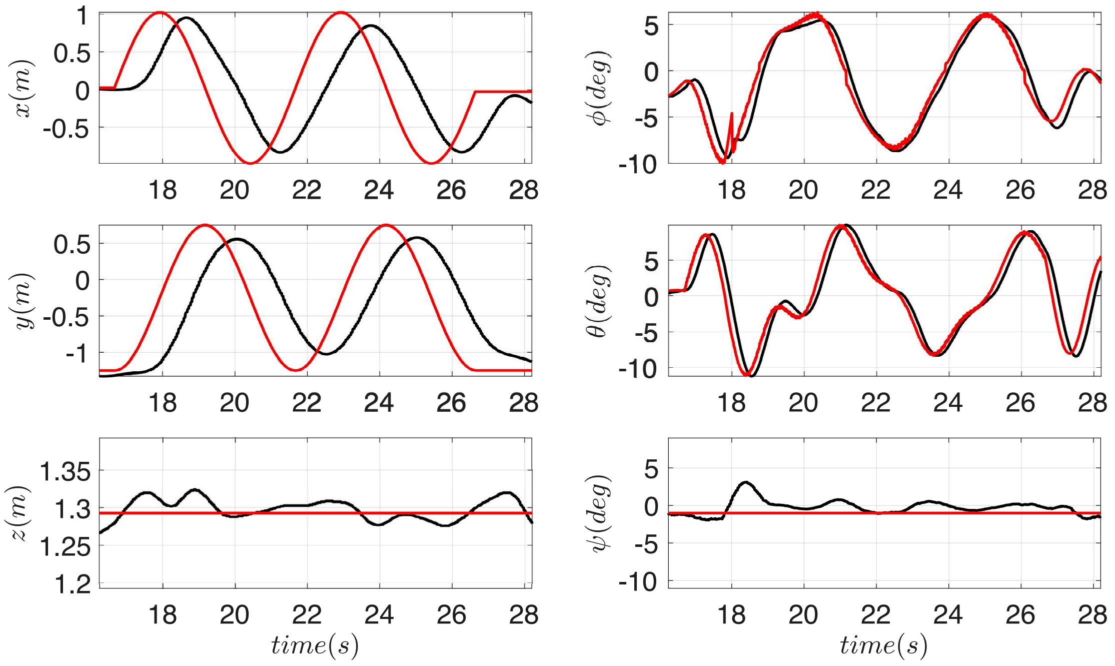


Figure 2.3: Circular trajectory (5 seconds period) tracking experiment result with (left) position and (right) attitude histories of the a multi-rotor. The red line denotes desired values, and the black line denotes measurement data

For system identification of the *time-delay model* in (2.8), the parameter γ_h is estimated by minimizing the following error metric:

$$\gamma_h = \underset{t_d}{\operatorname{argmin}} \sum \Big(|\phi(t) - \phi_{des}(t - t_d)| + |\theta(t) - \theta_{des}(t - t_d)| \Big). \tag{2.11}$$

Fig. 2.4 shows the estimation result from the brute-force search, and the estimated γ_h has the same value regardless of the trajectory periods.

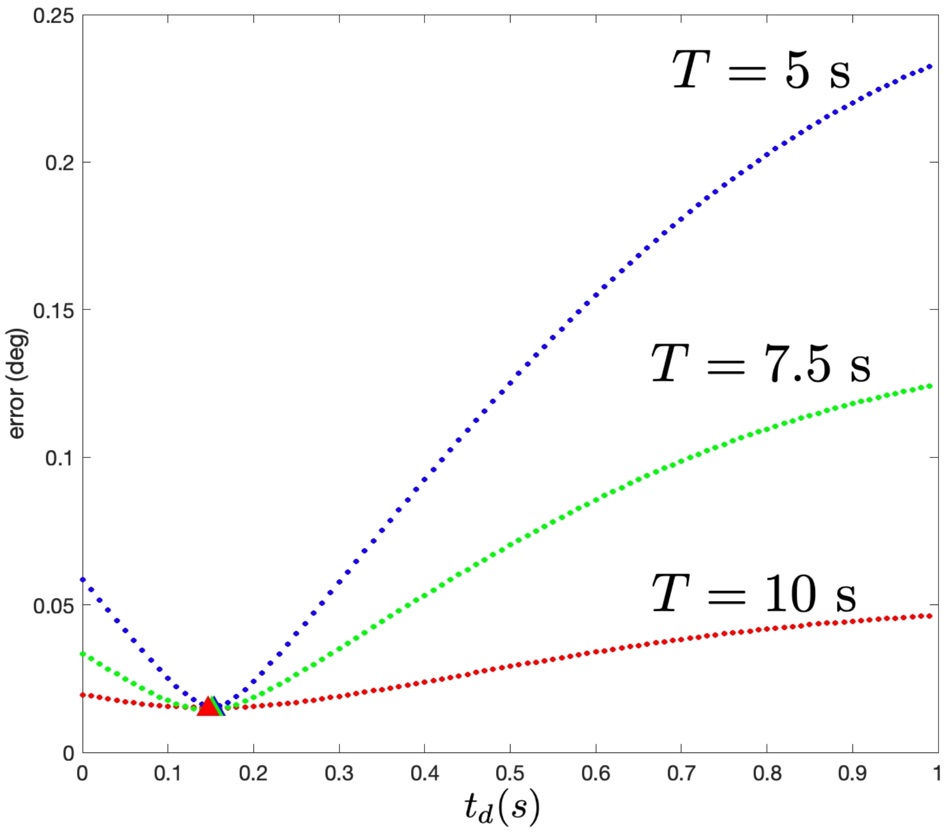


Figure 2.4: Horizontal time-delay estimation using (2.11) where the red, green, and blue lines are computed from the trajectories with 10, 7.5, and 5 seconds periods. The triangles denote the estimated time-delays and the minimum errors.

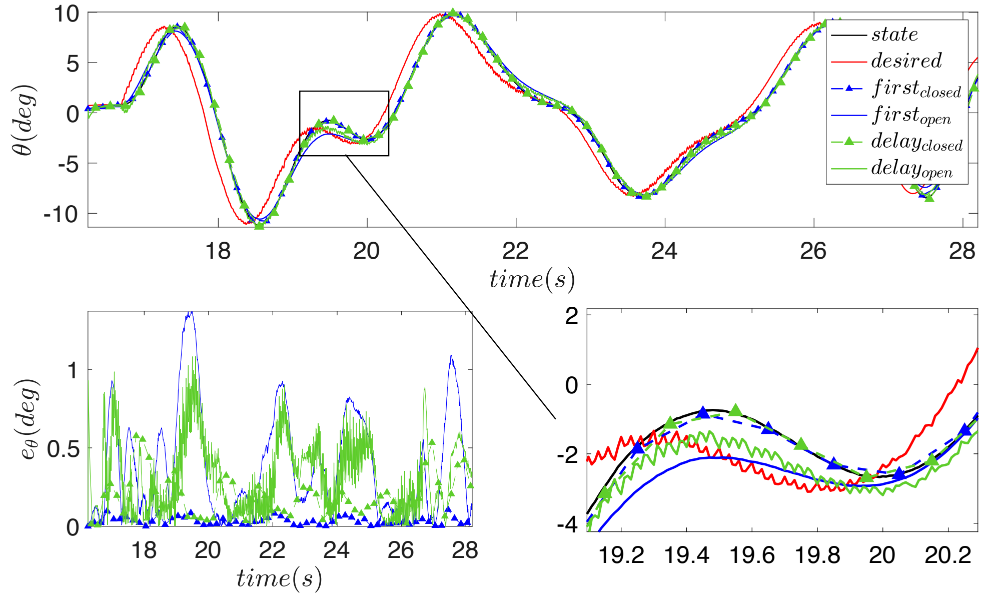


Figure 2.5: Simulation result from the system identification. The black and red lines denote the measured and desired values, and the blue and green lines denote the simulation results of the *time-delay model* and the *first-order model*, respectively. For the simulation results, solid lines mean open-loop results while dashed lines with triangles mean closed-loop results.

Table 2.2: RMSE of the system identification for the candidate models.

	open-loop	closed-loop
time-delay	0.3883°	0.2722°
first-order	0.5244°	**0.0391°**

For identification of the second candidate model, the *first-order model* in (2.9), the subspace method is applied [38]. Since the input/output response of roll and pitch angles is similar, the *first-order model* requires only two parameters, A_h and B_h. Identification is conducted with the concatenated roll and pitch data where the offset at the conjunction is removed.

Fig. 2.5 shows the simulation result from the identification. The RMSE of the candidate models is shown in Table 2.2. Since tendencies in roll and pitch angle are very similar regardless of the periods of the desired trajectories, only pitch angle result with five seconds period trajectory is shown here. Two kinds of the simulation are conducted for each candidate model which are open-loop and closed-loop ones. For the open-loop simulation, only initial condition is given, and integration is carried out with the desired values. On the other hand, the measurement data is given at each time step for the closed-loop simulation. Since state feedback of $\boldsymbol{x}_{F_{xy}}$ for the *time-delay model* cannot be done directly from measurement values while feedback of $\tilde{\phi}$ and $\tilde{\theta}$ is possible for the *first-order model* by IMU measurements, the following equation is used to compute $\boldsymbol{x}_{F_{xy}}$.

$$\boldsymbol{F} = \begin{bmatrix} C_{F_{xy}}\boldsymbol{x}_{F_{xy}} + D_{F_{xy}}\boldsymbol{F}_{xy.des} \\ F_{z.des} \end{bmatrix} = fR\boldsymbol{e}_3. \tag{2.12}$$

While open-loop simulation results show quite larger error, closed-loop simulation results show good performance. Especially, the *first-order model* shows more exact result in the closed-loop simulation since the *time-delay model* requires magnitude of thrust for feedback of $\boldsymbol{x}_{F_{xy}}$ state, which is not accurate. Compared to the original dynamics which has eighteen state variables and four control inputs, the *first-order model* has fourteen state variables and three control inputs. This simplification enables the trajectory generation and optimal control algorithms to find solutions faster. Therefore, the *first-order model* is used in this dissertation for the dynamics of the system. Note that the state vector and the control input vector *first-order model* are defined as:

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} \boldsymbol{x}_L^\top & \boldsymbol{q}^\top & \boldsymbol{v}_L^\top & \boldsymbol{\omega}^\top & \tilde{\phi} & \tilde{\theta} \end{bmatrix}^\top \in \mathbb{R}^{14\times 1}, \\ \boldsymbol{u} &= \begin{bmatrix} f & \tilde{\phi}_{des} & \tilde{\theta}_{des} \end{bmatrix}^\top \in \mathbb{R}^{3\times 1}. \end{aligned} \tag{2.13}$$

2.2 Trajectory Generation

For safe flight of the vehicle, an efficient planning algorithm which generates collision-free trajectories fast enough is required. While only the load is considered in the previous work [5], all the multi-rotor, cable, and load are considered for collision avoidance in this dissertation. Moreover, the current state can be fully considered, which is also not possible in [5]. An augmented Lagrangian method in [39] is employed to solve a nonlinear optimization problem with the proposed obstacle avoidance constraints.

2.2.1 Cost Functional

To find an optimal trajectory given the current state, $\boldsymbol{x}(1)$, and the goal state, $\boldsymbol{x}_f$, the following cost functional is designed to reach the goal state while regulating the state and minimizing the control input:

$$J_T\big(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)\big) = \frac{1}{2}\|\boldsymbol{x}(N_T+1) - \boldsymbol{x}_f\|_L^2 + \frac{1}{2}\sum_{k=1}^{N_T}\Big(\|\boldsymbol{x}(k) - \boldsymbol{x}_f\|_Q^2 + \|\boldsymbol{u}(k) - \boldsymbol{u}_{des}\|_R^2\Big)dt, \tag{2.14}$$

where $\| * \|_X = \sqrt{*^\top X *}$ and N_T is the number of control inputs to be optimized. The matrices $L, Q \in \mathbb{R}^{14\times 14}$, and $R \in \mathbb{R}^{3\times 3}$ are weight matrices for the optimization. The state and input trajectory are defined as follows:

$$\begin{aligned} \boldsymbol{x}(\cdot) &= \{\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots \boldsymbol{x}(N_T+1)\}, \\ \boldsymbol{u}(\cdot) &= \{\boldsymbol{u}(1), \boldsymbol{u}(2), \cdots \boldsymbol{u}(N_T)\}. \end{aligned} \tag{2.15}$$

The desired input vector, $\boldsymbol{u}_{des} = \begin{bmatrix}(m_M + m_L)g \ 0 \ 0\end{bmatrix}^\top$, is a constant vector required to maintain equilibrium when hovering with the load halted.

2.2.2 Collision Avoidance Constraints

Considering obstacles in trajectory generation as constraints is necessary for the safety. As an example of the constraint, the system can be considered as a sphere, and a simple constraint function keeping the sphere from colliding with the obstacles can be imposed, which is widely used for the bare multi-rotor systems. However, for the system in this work, such a constraint is very conservative since the enclosing sphere becomes bigger as the cable length grows. To design a less-conservative constraint function, this dissertation considers the system as three safety ellipsoids which enclose the multi-rotor, cable, and suspended load, respectively. The following equation is used to express ellipsoids in this work:

$$\mathcal{E}(\boldsymbol{p}, Q) = \{\boldsymbol{p} + Q^{1/2}\boldsymbol{v} \mid \boldsymbol{v}^\top \boldsymbol{v} = 1\}, \tag{2.16}$$

where $\boldsymbol{p}$ is the center of the ellipsoid, and Q is a shape matrix decided by length and direction of the principal axes.

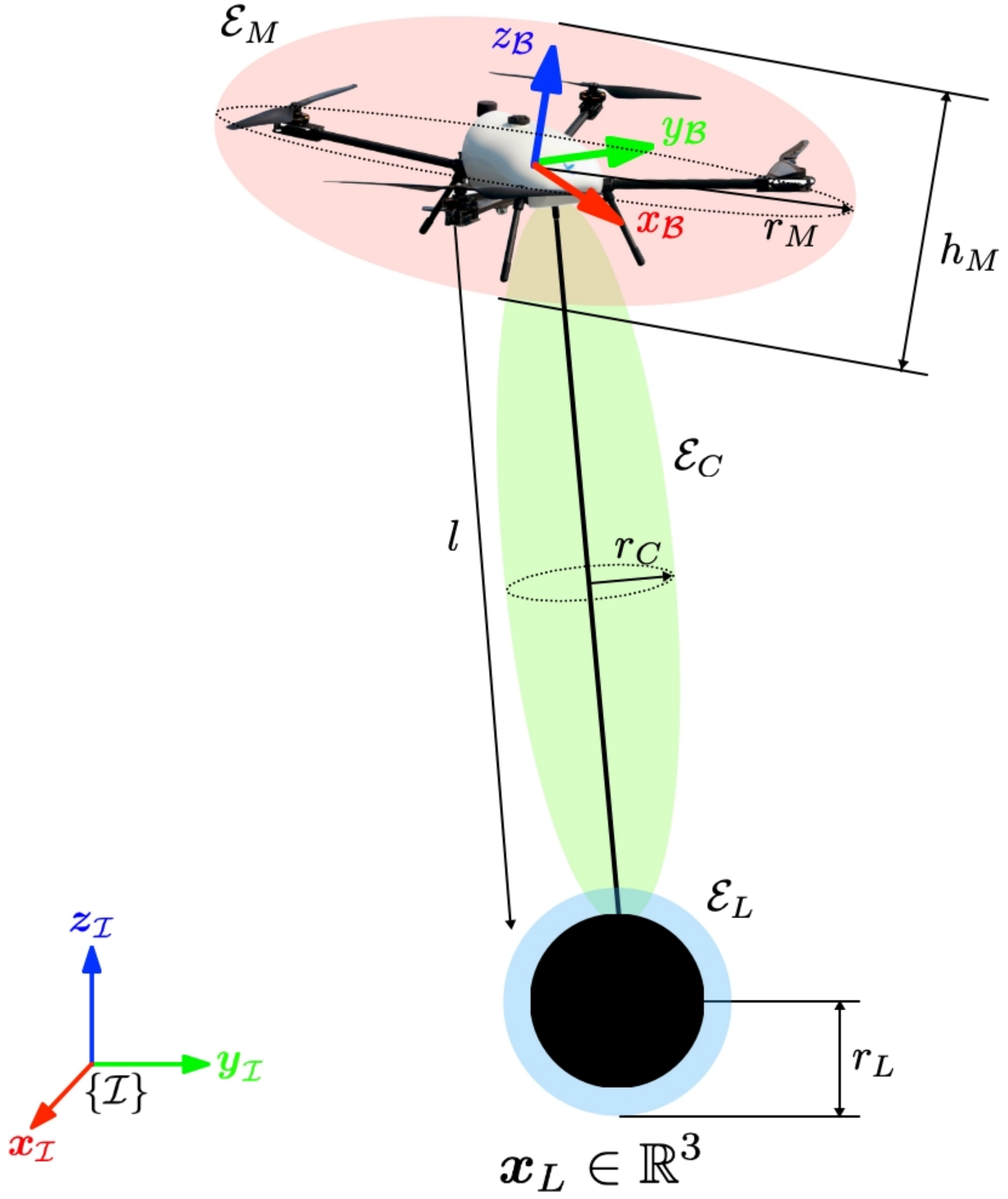


Figure 2.6: Coordinate frames and safety ellipsoids with the dimensions.

The safety ellipsoids are shown in Fig. 2.6 with parameters depending on the state variables and the physical size of the system. Each ellipsoid enclosing the multi-rotor, cable, and load is defined as $\mathcal{E}_M(\boldsymbol{p}_M, Q_M)$, $\mathcal{E}_C(\boldsymbol{p}_C, Q_C)$, and $\mathcal{E}_L(\boldsymbol{p}_L, Q_L)$. The parameters for them are defined as follows:

$$\begin{aligned}
&\boldsymbol{p}_M = \boldsymbol{x}_L - \boldsymbol{q}, \quad && Q_M = \tilde{R}^\top diag\big(r_M^2,\ r_M^2,\ (h_M/2)^2\big)\tilde{R},\\
&\boldsymbol{p}_C = \boldsymbol{x}_L - \frac{1}{2}\boldsymbol{q}, && Q_C = R_C^\top diag\big(r_C^2,\ r_C^2,\ (l/2)^2\big)R_C,\\
&\boldsymbol{p}_L = \boldsymbol{x}_L, && Q_L = diag\big(r_L^2,\ r_L^2,\ r_L^2\big),
\end{aligned} \tag{2.17}$$

where the load is assumed to be enclosed by a sphere with a radius of r_L. The matrix R_C is defined from the current cable direction as follows:

$$\begin{aligned}
\boldsymbol{r}_{C.3} &= -\frac{\boldsymbol{q}}{l},\\
\boldsymbol{r}_{C.1} &= \boldsymbol{r}_{C.3} \times \boldsymbol{e}_3,\\
\boldsymbol{r}_{C.2} &= \boldsymbol{r}_{C.3} \times \boldsymbol{r}_{C.1},\\
R_C &= \begin{bmatrix} \boldsymbol{r}_{C.1} & \boldsymbol{r}_{C.2} & \boldsymbol{r}_{C.3} \end{bmatrix}.
\end{aligned} \tag{2.18}$$

This kind of safety ellipsoid enclosure expands to represent obstacles also. This work assumes that the position and shape of the obstacles are known in advance because detection of them is beyond the scope of the dissertation. Therefore, the obstacles are expressed with $\mathcal{E}_{O.i}(\boldsymbol{p}_{O.i}, Q_{O.i})$ for $i = 1 \cdots N_O$ where N_O is the number of the obstacles.

Safety is assured by eliminating intersection between the ellipsoids enclosing the system and the ellipsoids enclosing the obstacles. Existence of intersection of two ellipsoids, $\mathcal{E}_1(\boldsymbol{p}_1, Q_1)$ and $\mathcal{E}_2(\boldsymbol{p}_2, Q_2)$, can be checked using Minkowski sum, and it does not exist if the following condition is satisfied:

$$\boldsymbol{0} \notin \mathcal{E}_1 \oplus (-\mathcal{E}_2). \tag{2.19}$$

Since Minkowski sum of two ellipsoids is usually not an ellipsoid, it requires approximation to find an analytic solution and to check the condition (2.19). The trace minimization approach in [40] is adopted to find the smallest ellipsoid encompassing the Minkowski sum. The approximated ellipsoid, $\hat{\mathcal{E}}_{1,2}(\hat{\boldsymbol{p}}_{1,2}, \hat{Q}_{1,2})$, can be analytically computed using the following equations:

$$\begin{aligned}
&\mathcal{E}_1 \oplus (-\mathcal{E}_2) \subset \hat{\mathcal{E}}_{1,2}(\hat{\boldsymbol{p}}_{1,2}, \hat{Q}_{1,2}) \\
&\hat{\boldsymbol{p}}_{1,2} = \boldsymbol{p}_1 - \boldsymbol{p}_2 \\
&\hat{Q}_{1,2} = Q_1\left(1 + \frac{\sqrt{tr(Q_2)}}{\sqrt{tr(Q_1)}}\right) + Q_2\left(1 + \frac{\sqrt{tr(Q_1)}}{\sqrt{tr(Q_2)}}\right).
\end{aligned} \tag{2.20}$$

Therefore, the safety condition (2.19) can be expressed as a constraint function with the approximated ellipsoid as follows:

$$\hat{\boldsymbol{p}}_{1,2}^{\top}\hat{Q}_{1,2}^{-1}\hat{\boldsymbol{p}}_{1,2} - 1 > 0, \tag{2.21}$$

and collision avoidance constraints used in trajectory generation are defined as follows:

$$\begin{aligned}
&\hat{\boldsymbol{p}}_{M,O.i}^{\top}\hat{Q}_{M,O.i}^{-1}\hat{\boldsymbol{p}}_{M,O.i} - 1 > 0 \\
&\hat{\boldsymbol{p}}_{C,O.i}^{\top}\hat{Q}_{C,O.i}^{-1}\hat{\boldsymbol{p}}_{C,O.i} - 1 > 0 \quad \text{for} \;\; i = 1 \cdots N_O. \\
&\hat{\boldsymbol{p}}_{L,O.i}^{\top}\hat{Q}_{L,O.i}^{-1}\hat{\boldsymbol{p}}_{L,O.i} - 1 > 0
\end{aligned} \tag{2.22}$$

Note that $\hat{\boldsymbol{p}}_{i,j}$ and $\hat{Q}_{i,j}$ are computed from $p_i,\ p_j,\ Q_i$, and Q_j using the same procedure in (2.20).

2.2.3 Augmented Lagrangian Method

For the optimization with the designed cost functional and the proposed constraints, the augmented Lagrangian method with the differential dynamic programming (DDP) approach is adopted. The following minimization problem is what needs to be solved for the trajectory generation:

$$\begin{aligned} \underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & J_T(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) \\ \text{subject to} \; & \boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \text{ for } k = 1 \cdots N_T, \\ & \boldsymbol{x}(1) = \boldsymbol{x}_0, \\ & c_{M.i}(\boldsymbol{x}(k)) > 0 \text{ for } k = 1 \cdots N_T,\; i = 1 \cdots N_O, \\ & c_{C.i}(\boldsymbol{x}(k)) > 0 \text{ for } k = 1 \cdots N_T,\; i = 1 \cdots N_O, \\ & c_{L.i}(\boldsymbol{x}(k)) > 0 \text{ for } k = 1 \cdots N_T,\; i = 1 \cdots N_O, \end{aligned} \tag{2.23}$$

where the first constraint is for the dynamic feasibility discretized from the proposed dynamic model, (2.1) and (2.9). The second one is for the initial condition and the last three terms are for collision avoidance in (2.22) for each time step and obstacle. The augmented Lagrangian method transforms the original constrained optimization into unconstrained optimization by adding both the penalty terms and the Lagrangian multiplier terms as follows:

$$\begin{aligned} \underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & \mathcal{L}_a = J_T(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) + \\ & \sum_{k=1}^{N_T} \left\{ \boldsymbol{c}^\top(\boldsymbol{x}(k)) I_{\mu_k} \boldsymbol{c}(\boldsymbol{x}(k)) + \boldsymbol{\lambda}_k^\top \boldsymbol{c}(\boldsymbol{x}(k)) \right\}, \end{aligned} \tag{2.24}$$

where $\boldsymbol{c}(\boldsymbol{x}(k)),\ \boldsymbol{\lambda}_k \in \mathbb{R}^{3N_O}$ are the vertical concatenation of the collision avoidance constraints and Lagrange multipliers at the k^{th} time step, respectively. The matrix $I_{\mu_k} \in \mathbb{R}^{3N_O \times 3N_O}$ is a diagonal matrix whose diagonal terms are defined as:

$$I_{\mu_k}(i,i) = \begin{cases} \mu_k^i & \text{if } c_i(\boldsymbol{x}(k)) < 0 \text{ or } \lambda_k^i > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.25}$$

where $c_i(\boldsymbol{x}(k))$ and λ_k^i are the i^{th} elements of $\boldsymbol{c}(\boldsymbol{x}(k))$ and $\boldsymbol{\lambda}_k$, respectively.

The unconstrained optimization problem in (2.24) with initial values of μ and λ is solved by iterative Linear Quadratic Regulation (iLQR) [41]. After convergence, the constraint functions are checked, and μ and λ are updated if any magnitude of constraint violation is larger than a threshold value. While μ is updated according to a predefined schedule, λ value is updated using the first-order necessary condition for optimality as follows:

$$\lambda_k^i \leftarrow \lambda_k^i - \mu_k^i c_i(\boldsymbol{x}(k)). \tag{2.26}$$

The detailed algorithm for the optimization is explained in Algorithm 1. The unconstrained problem, (2.24), is solved using the iLQR method (*line 3*). After convergence, the constraint violation is checked. If the i^{th} constraint for the k^{th} time step is not violated more than a threshold value κ_k^i (*line 7*), the Lagrangian multipliers are updated and the threshold value is reduced (*lines 8-9*). If the constraint violates more than the threshold value, the penalty term is increased (*line 11*). After the outer loop update, an unconstrained problem with the updated parameters are optimized again. These procedures of inner loop optimization and outer loop update are repeated until all the constraints are satisfied (*line 2*).

Algorithm 1 Augmented Lagrangian method [39]

```
Input: $I_\mu,\ \boldsymbol{\lambda},\ \boldsymbol{\kappa},\ \boldsymbol{x}(1),\ \boldsymbol{u}_n(\cdot)$
Output: $J^*,\ \boldsymbol{x}(\cdot)^*,\ \boldsymbol{u}(\cdot)^*$
 1: $\boldsymbol{u}_{temp}(\cdot) = \boldsymbol{u}_n(\cdot)$
 2: while $max(\boldsymbol{c}) > \epsilon_c$ do
 3:     Optimize the unconstrained problem (2.24) (inner loop)
        $\rightarrow \{\boldsymbol{x}_{temp}(\cdot), \boldsymbol{u}_{temp}(\cdot)\} = iLQR(\boldsymbol{x}(1), \boldsymbol{u}_n(\cdot), I_\mu, \lambda, \kappa)$ [41]
 4:     Update the penalty terms and Lagrangian multipliers (outer loop)
 5:     for $k = 1, \cdots, N_T + 1$ do
 6:         for $i = 1, \cdots, 3$ do Three obstacle avoidance constraints
 7:             if $c_i(\boldsymbol{x}(k)) < \kappa_k^i$ then
 8:                 $\lambda_k^i = \lambda_k^i - \mu_k^i c_i(\boldsymbol{x}(k))$
 9:                 Reduce $\kappa_k^i$
10:             else
11:                 Increase $\mu_k^i$
12:             end if
13:         end for
14:     end for
15: end while
```

2.3 Optimal Control

An optimization-based controller is adopted to track the generated trajectory. Compared to the conventional ones, this kind of controller can consider the state fully at once without the cascade structure. Moreover, the priority of the state and input variables in tracking can be modulated using gain matrices. The optimal control is computed using the following cost functional:

$$\begin{aligned} J_C\big(\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)\big) &= \frac{1}{2}\|\boldsymbol{x}(N_C+1)-\boldsymbol{x}^*(N_C+1)\|_L^2 \\ &\qquad + \frac{1}{2}\sum_{k=1}^{N_C}\Big\{\|\boldsymbol{x}(k)-\boldsymbol{x}^*(k)\|_Q^2+\|\boldsymbol{u}(k)-\boldsymbol{u}_{des}\|_R^2\Big\}dt \\ &= g(\boldsymbol{x}(N_C+1))+\frac{1}{2}\sum_{i=1}^{N_C}\big\{h(\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot))\big\} \end{aligned} \tag{2.27}$$

where N_C is a control horizon calculated by subtracting time-delay in trajectory generation from N_T. The desired trajectory, $\boldsymbol{x}^*(\cdot)$, is also front-truncated from the generated collision-free trajectory by the amount of the time-delay, which is implemented to compensate and minimize the effect of computation time in trajectory generation. To solve the optimal control problem, the sequential linear quadratic (SLQ) solver is implemented in a model predictive control (MPC) manner.

2.3.1 Sequential Linear Quadratic Solver

A stable and fast optimal control solver is required to compute an optimal control problem at each time step. The sequential linear quadratic (SLQ) solver is implemented whose speed and performance has been previously demonstrated in agile flight experiments [12].

The detailed algorithm of SLQ is shown in Algorithm 2. At first, the current state and nominal control input are given for initialization. After forward simulation with the nominal input (*line 2*), the dynamics is linearized along the computed nominal trajectory (*line 3*). Here, analytic derivatives are used instead of numerical ones to reduce computation time.

Symbolic expressions are computed in MATLAB and the result is embedded in the solver. Then, the cost functional is quadratized to compute gradients of the cost functional (*line 4*). If the form of the designed cost functional are similar to the quadratic function, the analytic derivatives can easily be computed. With the linearized dynamics and the quadratized cost functional, it is possible to find a feedforward input, $\boldsymbol{l}(k)$, and a feedback gain, $K(k)$, by solving backward Riccati-like equations (*line 5*). Since the dynamics is linearized, using the feedforward input and the feedback gain directly is not appropriate since there exist linearization error. Iteratively finding a lower cost by line search is a good method which can reduce the linearization error by finding the proper magnitude of the update (*lines 6-13*). Also, it can save time and reduce the number of iterations of SLQ. An exponentially decreasing values of the step size, α, is used with a scale parameter k_{line}. After completing the line search, if the new cost does not decrease more than the predefined value, $stop_{SLQ}$, the optimization is judged to be converged (*lines 14-19*).

Algorithm 2 SLQ

```
Input: $\boldsymbol{x}(1)$, $\boldsymbol{u}_n(\cdot)$
Output: $J^*$, $\boldsymbol{x}(\cdot)^*$, $\boldsymbol{u}(\cdot)^*$, $K(\cdot)^*$, $\boldsymbol{l}(\cdot)^*$, $\alpha^*$
 1: while ($iter_{SLQ} <= max_{SLQ}$)  {SLQ loop} do
 2:     Simulate the system forward with the dynamics
        $\rightarrow \boldsymbol{x}_n(1), \boldsymbol{x}_n(2), \cdots, \boldsymbol{x}_n(N_C), \boldsymbol{x}_n(N_C+1)$
 3:     Linearize the dynamics along $\boldsymbol{x}_n$
        $\rightarrow \delta\boldsymbol{x}(k+1) = \delta\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \approx A(k)\delta\boldsymbol{x}(k) + B(k)\delta\boldsymbol{u}(k)$
 4:     Quadratize the cost functional $J\big(\boldsymbol{x}(k)+\delta\boldsymbol{x}(k), \boldsymbol{u}(k)+\delta\boldsymbol{u}(k)\big)$
        $\rightarrow J \approx s_0(N_C+1) + \delta\boldsymbol{x}^\top(N_C+1)\boldsymbol{s}_1(N_C+1) + \frac{1}{2}\delta\boldsymbol{x}^\top(N_C+1)S_2(N_C+1)\delta\boldsymbol{x}(N_C+1)$
          $+ \sum_{k=1}^{N_C}\Bigg(q_0(k) + \delta\boldsymbol{x}^\top(k)\boldsymbol{q}_1(k) + \frac{1}{2}\delta\boldsymbol{x}^\top(k)Q_2(k)\delta\boldsymbol{x}(k)$
          $+\delta\boldsymbol{u}^\top(k)\boldsymbol{r}_1(k) + \frac{1}{2}\delta\boldsymbol{u}^\top(k)R_2(k)\delta\boldsymbol{u}(k)\Bigg)dt$
          where $\boldsymbol{s}_1(N_C+1) = \frac{\partial g}{\partial \boldsymbol{x}}\Big|_{N_C+1}^\top,\ S_2(N_C+1) = \frac{\partial g^2}{\partial \boldsymbol{x}^2}\Big|_{N_C+1},$
                $\boldsymbol{q}_1(k) = \frac{\partial h}{\partial \boldsymbol{x}}\Big|_k^\top,\ Q_2(k) = \frac{\partial h^2}{\partial \boldsymbol{x}^2}\Big|_k,\ \ \boldsymbol{r}_1(k) = \frac{\partial h}{\partial \boldsymbol{u}}\Big|_k^\top,\ R_2(k) = \frac{\partial h^2}{\partial \boldsymbol{u}^2}\Big|_k$
 5:     Minimize the cost functional by solving backward
        $\rightarrow \boldsymbol{s}_1(k-1) = \boldsymbol{q}_1(k-1) + A^\top(k-1)\boldsymbol{s}_1(k) - G^\top H^{-1}\boldsymbol{g}$
            $S_2(k-1) = Q_2(k-1) + A^\top(t-dt)S_2(k)A(k-1)$
                        $-G^\top H^{-1}G$
            $K(k-1) = -H^{-1}G$
            $\boldsymbol{l}(k-1) = -H^{-1}\boldsymbol{g}$
          where $G = B^\top(k-1)S_2(k)A(k-1)$
                $H = R_2(k-1) + B^\top(k-1)S_2(k)B(k-1)$
                $\boldsymbol{g} = \boldsymbol{r}_1(k-1) + B^\top(k-1)\boldsymbol{s}_1(k)$
 6:     while ($iter_{line} \le max_{line}$)  {line search loop} do
 7:         $\alpha = 10^{-k_{line}(iter_{line}-1)/(max_{line}-1)}$
 8:         Simulate the system forward
            $\rightarrow \boldsymbol{u}_{temp}(k) = \boldsymbol{u}_n(k) + \alpha\boldsymbol{l}(k) + K(k)\big(\boldsymbol{x}_{temp}(k) - \boldsymbol{x}_n(k)\big)$
               $\boldsymbol{x}_{temp}(k+1) = \boldsymbol{f}\big(\boldsymbol{x}_{temp}(k), \boldsymbol{u}_{temp}(k)\big)$
 9:         Compute the temporary cost
            $\rightarrow J_{temp} = J\big(\boldsymbol{x}_{temp}(\cdot), \boldsymbol{u}_{temp}(\cdot)\big)$
10:         if ($J_{temp} \le J_{old}$) then
11:             break with $\{J_{temp}, \boldsymbol{x}_{temp}(k), \boldsymbol{u}_{temp}(k), K(k), \alpha\boldsymbol{l}(k)\}$
12:         end if
13:     end while
14:     if ($J_{new} \ge J_{old} - stop_{SLQ}$) then
15:         return $\{J_{new},\ \boldsymbol{x}_{new}(\cdot),\ \boldsymbol{u}_{new}(\cdot),\ K(k),\ \boldsymbol{l}(\cdot),\ \alpha\}$
16:     else
17:         $J_{old} = J_{new}, \boldsymbol{u}_n(k) = \boldsymbol{u}_{new}(k)$
18:         $iter_{SLQ} = iter_{SLQ} + 1$
19:     end if
20: end while
```

2.3.2 Model Predictive Control

In this section, customized MPC settings are addressed which increase stability and agility in experiments. In reality, since there exist various disturbances and errors unlike simulations, the system cannot be controlled perfectly. For continuous feedback control, MPC is implemented. However, MPC algorithms tend to reduce magnitudes of control input as vehicle approaches final desired state, and such small control input might be unable to control the system stably due to disturbances or actuator delay. The proposed MPC algorithm is described in detail in Algorithm 3.

By changing the length of the time horizon adaptively, it is possible to solve the above-mentioned issue of decreasing input magnitude when approaching the final desired state. By changing the time horizon proportionally to the distance from the load to the final position (*line 1*), the control input can maintain appropriate magnitudes. Conversely, the length of time horizon decreases as the distance to final position becomes closer. If the time horizon is too long, the computation time is lengthened, and if it is too short, the solution becomes less stable. These problems are eliminated by empirically setting the upper and lower limits to length of time horizon (*lines 2-6*).

With the computed time horizon and the final desired state, a stable control input is computed using a nominal controller. For a nominal controller, the geometric controller [42] is used whose stability is verified. Using the control input from the nominal controller (*lines 7-10*) as a starting point, optimal control input and trajectory can be computed by the SLQ solver (*line 11*). For the first optimization, the geometric controller computes the nominal control input to follow a simple polynomial trajectory avoiding obstacles (*line 8*). After one convergence, the previous SLQ result is used for warm start (*line 9*). Since the previous result can avoid obstacles, giving it to the nominal controller enables generating collision-free nominal trajectory with a near-optimal control. This method not only reduces the computation time but also increases stability since the previous SLQ is near the optimal result.

Algorithm 3 MPC

```
Input: $t_{now}$, $\boldsymbol{x}(t_{now})$
Output: $J^*$, $\boldsymbol{x}(\cdot)^*$, $\boldsymbol{u}(\cdot)^*$
 1: Compute time horizon length adaptively
    $\rightarrow d_f = \sqrt{\left(\boldsymbol{x_L} - \begin{bmatrix} x_{L_f} & y_{L_f} & z_{L_f} \end{bmatrix}^T\right)^T \left(\boldsymbol{x_L} - \begin{bmatrix} x_{L_f} & y_{L_f} & z_{L_f} \end{bmatrix}^T\right)}$
       $H = round(d_f \times k_{adaptive})$
 2: if $H > H_{max}$ then
 3:     $H = H_{max}$
 4: else if $H < H_{min}$ then
 5:     $H = H_{min}$
 6: end if
 7: Find a nominal stable control input
 8: if (*At the beginning*) then
    $\rightarrow \boldsymbol{u}_n(\cdot) = nominalController(\boldsymbol{x}(1),\ \boldsymbol{x}_f)$
 9: else if (*Afterward*) then
    $\rightarrow \boldsymbol{u}_n(\cdot) = nominalController(\boldsymbol{x}(1),\ \boldsymbol{x}(\cdot)^*_{previous})$
10: end if
11: Solve SLQ
    $\rightarrow \{J^*,\ \boldsymbol{x}(\cdot)^*,\ \boldsymbol{u}(\cdot)^*,\ K(\cdot)^*,\ \boldsymbol{l}(\cdot)^*,\ \alpha\} = SLQ\big(\boldsymbol{x}(1), \boldsymbol{u}_n(\cdot)\big)$
12: Repeat until the system meets the final desired state
```

3

Multiple Multi-rotors with a Suspended Load

This chapter addresses the system composed of multiple multi-rotors with a suspended load. Sine this system has more number of states and control inputs increasing proportionally to the number of the vehicle, applying the proposed algorithms in Chapter 2 results in a high computational load. Not using the large dynamic model as it is, this dissertation utilizes the differential flatness to reduce the number of the optimization variables. For the further efficiency, an entire optimization problem is parameterized and decomposed into two sub-problems which are for the load trajectory and tension histories, respectively. Refer to Fig. 1.2(b) for the flow chart of the chapter.

3.1 Problem Setting

3.1.1 Dynamics

Various dynamic models are used to express the internal effects of the tension of the cables [24, 25, 30]. The model in [24] is derived using the Newtonian mechanics with the explicit cable tension terms. Since tension is an internal force that cannot be known prior to exerting control inputs, Lagrangian mechanics can be used to express the motion without the explicit cable tension terms as derived in [25, 30]. While [30] assumes cable tautness, [25] derives a dynamic model using flexible cables without the tautness assumption.

In this paper, since the cable tensions are the optimization variables and tautness can be guaranteed in the trajectory generation step through constraints, the model in [24] is used:

$$\begin{aligned} &\text{(vehicle)} \quad && \frac{d}{dt}\boldsymbol{x}_i = \dot{\boldsymbol{x}}_i, \\ & && \frac{d}{dt}\dot{\boldsymbol{v}}_i = \frac{1}{m_i}\big(f_i R_i \boldsymbol{e}_3 - m_i g \boldsymbol{e}_3 - T_i \boldsymbol{q}_i\big), \\ & && \frac{d}{dt}R_i = R_i \hat{\boldsymbol{\Omega}}_{\mathcal{B}_i}, \\ & && \frac{d}{dt}\boldsymbol{\Omega}_{\mathcal{B}_i} = J_i^{-1}\big(\boldsymbol{M}_i - \boldsymbol{\Omega}_{\mathcal{B}_i} \times J_i \boldsymbol{\Omega}_{\mathcal{B}_i}\big), \\ &\text{(load)} && \frac{d}{dt}\boldsymbol{x}_L = \dot{\boldsymbol{x}}_L, \\ & && \frac{d}{dt}\dot{\boldsymbol{v}}_L = \frac{1}{m_L}\left(\sum_{i=1}^{n_M} T_i \boldsymbol{q}_i - m_L g \boldsymbol{e}_3\right). \end{aligned} \tag{3.1}$$

The definition of the variables is listed in Table 3.1, and a three-vehicle system is illustrated in Fig. 1.1(b), i.e., $n_M = 3$. Note that, compared to [24], the only difference is that the sign of $\boldsymbol{q}_i$ is flipped in this work for intuitive interpretation during the convexification process. The state and the input vector are defined as follows:

$$\begin{aligned} \boldsymbol{x} &= \left[\boldsymbol{x}_L^\top \ \boldsymbol{v}_L^\top \ \boldsymbol{x}_M^\top \ \boldsymbol{v}_M^\top \ \boldsymbol{\Phi}_M^\top \ \boldsymbol{\Omega}_M^\top\right]^\top \in \mathbb{R}^{6+12n_M}, \\ \boldsymbol{u} &= \left[\boldsymbol{f}_M^\top \ \boldsymbol{M}_M^\top\right]^\top \in \mathbb{R}^{4n_M}, \end{aligned} \tag{3.2}$$

where $*_M = [*_1^\top \ *_2^\top \ \cdots \ *_{n_M}^\top]^\top$ is a concatenated vector of elements for all the multi-rotors. This dynamics cannot be directly used since the tension vectors, $\boldsymbol{T}_M$, require additional sensors to be measured. The motion of the load and the multi-rotor is innately coupled and the dynamic model is highly non-linear with a large number of states and inputs, which results in difficulties to secure short computation time for optimization.

Table 3.1: Nomenclature of Chapter 3

Notation	Meaning
$n_x,\ x_u,\ n_M,\ n_B,\ n_C,\ n_\zeta,\ n_\nu \in \mathbb{R}^1$	The number of states, inputs, multi-rotors, trajectory segments, control points, flat outputs, and vertices of SFCs
$\mathcal{I}$	Inertial frame
$\mathcal{B}_i$	Body-fixed frame of the i^{th} multi-rotor
$\boldsymbol{x}_\mathcal{I},\ \boldsymbol{y}_\mathcal{I},\ \boldsymbol{z}_\mathcal{I} \in \mathbb{R}^3$	x, y, z axes of $\mathcal{I}$
$\boldsymbol{x}_{\mathcal{B}_i},\ \boldsymbol{y}_{\mathcal{B}_i},\ \boldsymbol{z}_{\mathcal{B}_i} \in \mathbb{R}^3$	x, y, z axes of $\mathcal{B}_i$
$\boldsymbol{x}_L, \boldsymbol{x}_i \in \mathbb{R}^3$	Position of the load and the i^{th} multi-rotor expressed in $\mathcal{I}$
$\boldsymbol{x}_M \in \mathbb{R}^{3n_M}$	Concatenated positions of the multi-rotors
$\boldsymbol{v}_L, \boldsymbol{v}_i \in \mathbb{R}^3$	Velocity of the load and the i^{th} multi-rotor expressed in $\mathcal{I}$
$\boldsymbol{v}_M \in \mathbb{R}^{3n_M}$	Concatenated velocities of the multi-rotors
$T_i \in \mathbb{R}^1$	Tension magnitude of the i^{th} cable
$\boldsymbol{q}_i \in \mathbb{S}^2$	Direction of the i^{th} multi-rotor with respect to the load expressed in $\mathcal{I}$
$\boldsymbol{T}_i \in \mathbb{R}^3$	Tension vector of the i^{th} multi-rotor with respect to the load expressed in $\mathcal{I}$
$\boldsymbol{T}_M \in \mathbb{R}^{3n_M}$	Concatenated tension vectors of the multi-rotors
$R_i \in SO(3)$	Rotation matrix of the i^{th} multi-rotor from $\mathcal{I}$ to $\mathcal{B}_i$
$\boldsymbol{\Phi}_i = \begin{bmatrix}\phi_i\ \theta_i\ \psi_i\end{bmatrix}^\top \in \mathbb{R}^3$	Roll, pitch, and yaw angle of the i^{th} multi-rotor
$\boldsymbol{\Phi}_M \in \mathbb{R}^{3n_M}$	Concatenated attitudes of the multi-rotors
$\boldsymbol{\Omega}_{\mathcal{B}_i} = \begin{bmatrix}p_i\ q_i\ r_i\end{bmatrix}^\top \in \mathbb{R}^3$	Angular velocity of the i^{th} multi-rotor expressed in $\mathcal{B}_i$
$\boldsymbol{\Omega}_{\mathcal{I}_i} \in \mathbb{R}^3$	Angular velocity of the i^{th} multi-rotor expressed in $\mathcal{I}$
$\boldsymbol{\Omega}_M \in \mathbb{R}^{3n_M}$	Concatenated angular velocities of the multi-rotors

$f_i \in \mathbb{R}^1$	Thrust input of the i^{th} multi-rotor
$\boldsymbol{f}_M \in \mathbb{R}^{n_M}$	Concatenated thrusts of the multi-rotors
$\boldsymbol{M}_i \in \mathbb{R}^3$	Moment input of the i^{th} multi-rotor expressed in $\mathcal{B}_i$
$\boldsymbol{M}_M \in \mathbb{R}^{3n_M}$	Concatenated moment inputs of the multi-rotors
$\boldsymbol{x} \in \mathbb{R}^{n_x}$	States of the system
$\boldsymbol{u} \in \mathbb{R}^{n_u}$	Inputs of the system
$\boldsymbol{\zeta} \in \mathbb{R}^{n_\zeta}$	Flat outputs of the system
$\eta_L,\ \eta_T \in \mathbb{R}^1$	Maximum differentiation order of the flat outputs corresponding to the load and the tension for the differential flatness
$d_L,\ d_T \in \mathbb{R}^1$	Degree of the Bernstein polynomial corresponding to the load and the tension
$\boldsymbol{Z} \in \mathbb{R}^{3(\eta_L+1)+3(n_M-1)(\eta_T+1)}$	Flat outputs and their derivatives to be optimized
$g \in \mathbb{R}^1$	Gravitational constant
$l \in \mathbb{R}^1$	Length of the cables
$m_i,\, m_L \in \mathbb{R}^1$	Mass of the i^{th} multi-rotor and the load
$J_i \in \mathbb{R}^{3\times 3}$	Inertia matrix of the i^{th} multi-rotor
$\boldsymbol{e}_i \in \mathbb{R}^3$	Unit vector whose i^{th} element is 1

3.1.2 Differential Flatness

The system with multi-rotors and a suspended load is proved to be differentially flat [24]. While both the configuration with a point-mass load or a rigid-body load are proved to have the differential flatness, this dissertation focuses on only the point-mass type.

A system is said to be differentially flat if the state vector, $\boldsymbol{x} \in \mathbb{R}^{n_x}$, and the input vector, $\boldsymbol{u} \in \mathbb{R}^{n_u}$, can be expressed by flat outputs, $\boldsymbol{\zeta} \in \mathbb{R}^{n_\zeta}$, which are chosen as follows:

$$\boldsymbol{\zeta} = \left[\boldsymbol{x}_L^\top \ \boldsymbol{T}_2^\top \ \boldsymbol{T}_3^\top \ \cdots \ \boldsymbol{T}_{n_M}^\top \ \boldsymbol{\psi}_M^\top\right]^\top \in \mathbb{R}^{n_\zeta}, \tag{3.3}$$

where $\boldsymbol{\psi}_M = [\psi_1 \ \cdots \ \psi_{n_M}]^\top \in \mathbb{R}^{n_M}$ is a concatenated vector of yaw angles [24]. Note that the first tension vector, $\boldsymbol{T}_1$, is not included in the flat outputs. The detailed derivation of the differential flatness is shown in Appendix A.

Since the yaw angle of a multi-rotor can be easily controlled independently by an efficient attitude controller, it is assumed that the yaw angles of the multi-rotors are maintained at zero, which results in a reduction in the number of the flat outputs to be optimized, from $3 + 3(n_M - 1) + (n_M - 1)$ to $3 + 3(n_M - 1)$.

3.1.3 Optimization Problem

To generate trajectories quickly, trajectory optimization is performed using not the original state $\boldsymbol{x}$ and the input $\boldsymbol{u}$, but the flat output $\boldsymbol{\zeta}$. Then, the dynamics of $\boldsymbol{\zeta}$ can be considered with an affine function. For consideration of the constraints, Bernstein polynomials are used which have a convex hull property. Additionally, the polynomials can parameterize the continuous trajectory using a discrete number of control points. By performing convexification for the non-convex collision constraints, this dissertation proposes a convex optimization algorithm that can be solved efficiently. Refer to Appendix B for the detailed definition and the properties of Bernstein polynomials.

As stated in Appendix A, the flat output $\boldsymbol{x}_L$ and $\boldsymbol{T}_i$ need to be differentiated at least four and six times to express the states and the inputs of the original system, i.e. $\eta_L = 6$ and $\eta_T = 4$. It results in $3 \times (\eta_L + 1) + 3(n_M - 1)(\eta_T + 1)$ optimization variables which is expressed as

$$\boldsymbol{Z} = \left[\boldsymbol{Z}_L^\top \ \boldsymbol{Z}_T^\top\right]^\top, \tag{3.4}$$

where $\boldsymbol{Z}_L = \left[\boldsymbol{x}_L^\top \ (\boldsymbol{x}_L^{(1)})^\top \ \cdots \ (\boldsymbol{x}_L^{(6)})^\top\right]^\top$ and $\boldsymbol{Z}_T = \left[\boldsymbol{Z}_2^\top \ \boldsymbol{Z}_3^\top \ \cdots \ \boldsymbol{Z}_{n_M}^\top\right]^\top$. $\boldsymbol{Z}_i$ is defined as $\boldsymbol{Z}_i = \left[\boldsymbol{T}_i^\top \ (\boldsymbol{T}_i^{(1)})^\top \ \cdots \ (\boldsymbol{T}_i^{(4)})^\top\right]^\top$ and $*^{(\eta)}$ is the η^{th} derivative of $*$.

Cost Functional

The aim of the trajectory optimization is minimizing the following cost functional:

$$J(\boldsymbol{Z}_L(t), \boldsymbol{Z}_T(t)) = J_L(\boldsymbol{Z}_L(t)) + J_T(\boldsymbol{Z}_T(t)). \tag{3.5}$$

$J_L(\boldsymbol{Z}_L(t)) = \frac{1}{2}\int_{t_0}^{t_f} \left\|\boldsymbol{Z}_L^{(6)}(t)\right\|^2 dt$ and $J_T(\boldsymbol{Z}_T(t)) = \frac{1}{2}\sum_{i=2}^{n_M}\left\{\int_{t_0}^{t_f}\left\|\boldsymbol{T}_i^{(4)}(t)\right\|^2 dt\right\}$ indirectly minimize the energy consumption by reducing the magnitude of the control input.

Trajectory Parameterization

For optimization, the trajectories are parameterized using piece-wise Bernstein polynomials. To decide the degrees of the polynomials, the following Euler-Lagrange equation is used:

$$\frac{\partial \mathcal{L}}{\partial f_{\mathcal{L}}} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{f}_{\mathcal{L}}}\right) + \cdots + (-1)^\eta \frac{d^\eta}{dt^\eta}\left(\frac{\partial \mathcal{L}}{\partial f_{\mathcal{L}}^{(\eta)}}\right) = 0, \tag{3.6}$$

under fixed boundary conditions for $f_{\mathcal{L}}$ and its derivatives of orders up to $\eta - 1$. $\mathcal{L}\left(t, f_{\mathcal{L}}, \dot{f}_{\mathcal{L}}, \cdots, f_{\mathcal{L}}^{(\eta)}\right)$ is a Lagrangian of the cost functional. As an example, $f_{\mathcal{L}} = x_L(t)$ and $\mathcal{L} = \left(x_L^{(6)}(t)\right)^2$ for the x-component load trajectory based on (3.5). Then, (3.6) can be simplified with $\eta = \eta_L = 6$ as follows:

$$(-1)^6 \cdot \frac{d^6}{dt^6}\left(x_L^{(6)}(t)\right) = x_L^{(12)}(t) = 0, \tag{3.7}$$

which can be satisfied with the polynomials of degree $d_L = 11$. The same procedures are used for the tension histories, and (3.6) is satisfied with $\eta = \eta_T = 4$ and $d_T = 7$.

Then, the trajectories of the load and the tensions can be expressed as follows where only the x-components are shown as examples:

$$x_L(t) = \begin{cases} \sum_{j=1}^{d_L+1} c^1_{Lx.j} B_{d_L \cdot j}(\tau_1) & \text{for } t \in [t_0, t_1] \\ \vdots & \vdots \\ \sum_{j=1}^{d_L+1} c^{n_B}_{Lx.j} B_{d_L \cdot j}(\tau_{n_B}) & \text{for } t \in [t_{n_B-1}, t_{n_B}], \end{cases} \tag{3.8}$$

$$T_{ix}(t) = \begin{cases} \sum_{j=1}^{d_T+1} c^1_{ix.j} B_{d_T \cdot j}(\tau_1) & \text{for } t \in [t_0, t_1] \\ \vdots & \vdots \\ \sum_{j=1}^{d_T+1} c^{n_B}_{ix.j} B_{d_T \cdot j}(\tau_{n_B}) & \text{for } t \in [t_{n_B-1}, t_{n_B}], \end{cases} \tag{3.9}$$

which are n_B-segment polynomials with one second duration for each segment, i.e., $t_0 = 0$ and $t_{k+1} - t_k = 1$. The control points are the optimization variables, and they are defined as follows:

$$\begin{aligned} \boldsymbol{c}_L &= \left[\boldsymbol{c}_{Lx}^\top \ \boldsymbol{c}_{Ly}^\top \ \boldsymbol{c}_{Lz}^\top\right]^\top \in \mathbb{R}^{3n_B(d_L+1)}, \\ \boldsymbol{c}_{L*} &= \left[(\boldsymbol{c}^1_{L*})^\top \ (\boldsymbol{c}^2_{L*})^\top \ \cdots \ (\boldsymbol{c}^{n_B}_{L*})^\top\right]^\top \in \mathbb{R}^{n_B(d_L+1)}, \\ \boldsymbol{c}^k_{L*} &= \left[c^k_{L*.1} \ c^k_{L*.2} \ \cdots \ c^k_{L*.d_L+1}\right]^\top \in \mathbb{R}^{d_L+1}. \end{aligned} \tag{3.10}$$

$$\begin{aligned} \boldsymbol{c}_T &= \left[\boldsymbol{c}_2^\top \ \boldsymbol{c}_3^\top \ \cdots \ \boldsymbol{c}_{n_M}^\top\right]^\top \in \mathbb{R}^{3n_B(d_T+1)(n_M-1)}, \\ \boldsymbol{c}_i &= \left[\boldsymbol{c}_{ix}^\top \ \boldsymbol{c}_{iy}^\top \ \boldsymbol{c}_{iz}^\top\right]^\top \in \mathbb{R}^{3n_B(d_T+1)}, \\ \boldsymbol{c}_{i*} &= \left[(\boldsymbol{c}^1_{i*})^\top \ (\boldsymbol{c}^2_{i*})^\top \ \cdots \ (\boldsymbol{c}^{n_B}_{i*})^\top\right]^\top \in \mathbb{R}^{n_B(d_T+1)}, \\ \boldsymbol{c}^k_{i*} &= \left[c^k_{i*.1} \ c^k_{i*.2} \ \cdots \ c^k_{i*.d_T+1}\right]^\top \in \mathbb{R}^{d_T+1}. \end{aligned} \tag{3.11}$$

Then, with the defined trajectories and control points, the cost functional (3.5) can be parameterized as follows:

$$
\begin{aligned}
J(\boldsymbol{Z}_L(t), \boldsymbol{Z}_T(t)) = J(\boldsymbol{c}_L, \boldsymbol{c}_T) = J_L(\boldsymbol{c}_L) + J_T(\boldsymbol{c}_T), \\
J_L(\boldsymbol{c}_L) = \boldsymbol{c}_L^\top diag(H_L, H_L, H_L)\boldsymbol{c}_L, \\
J_T(\boldsymbol{c}_T) = \sum_{i=2}^{n_M} \left\{ \boldsymbol{c}_T^\top diag(H_T, H_T, H_T)\boldsymbol{c}_T \right\}.
\end{aligned}
\tag{3.12}
$$

$H_L \in \mathbb{S}_+^{n_B(d_L+1)}$ and $H_T \in \mathbb{S}_+^{n_B(d_T+1)}$ can easily be computed using the definition of the cost functional.

3.1.4 Problem Formulation

With the constraints for boundary conditions and collision avoidance, the following trajectory generation problem can be formulated:

Problem 1. *Entire optimization*

$$
\begin{aligned}
\textit{minimize} \quad & J(\boldsymbol{c}_L, \boldsymbol{c}_T) = J_L(\boldsymbol{c}_L) + J_T(\boldsymbol{c}_T) \\
\textit{subject to} \quad & A_{L.boundary}\boldsymbol{c}_L = \boldsymbol{b}_{L.boundary}, \\
& A_{T.boundary}\boldsymbol{c}_T = \boldsymbol{b}_{T.boundary}, \\
& A_{L.continuity}\boldsymbol{c}_L = \boldsymbol{b}_{L.continuity}, \\
& A_{T.continuity}\boldsymbol{c}_T = \boldsymbol{b}_{T.continuity}, \\
& g_L(\boldsymbol{c}_L) \geq \delta_L, \\
& \boldsymbol{g}_T(\boldsymbol{c}_L, \boldsymbol{c}_T) \succeq \delta_T, \\
& \boldsymbol{g}_I(\boldsymbol{c}_L, \boldsymbol{c}_T) \succeq \delta_I,
\end{aligned}
$$

where the first and second constraints are the boundary conditions which encode the following example equation of the x-component of the load for all axes of the load and tension:

$$\begin{aligned}
&\sum_{j=1}^{d_L+1} c^{1}_{Lx.j} B^{(\eta)}_{d_L \cdot j}(0) = x^{(\eta)}_{L.initial} \quad \text{for } \eta = 0 \cdots \eta_L - 1, \\
&\sum_{j=1}^{d_L+1} c^{n_B}_{Lx.j} B^{(\eta)}_{d_L \cdot j}(1) = x^{(\eta)}_{L.final} \quad \text{for } \eta = 0 \cdots \eta_L - 1.
\end{aligned} \tag{3.13}$$

For continuity of the trajectories including the derivatives, continuity constraints are also added which are the third and fourth ones. One example for x-component of the load trajectory is as follows:

$$\begin{gathered}
\sum_{j=1}^{d_L+1} c^{k}_{Lx.j} B^{(\eta)}_{d_L \cdot j}(1) = \sum_{j=1}^{d_L+1} c^{k+1}_{Lx.j} B^{(\eta)}_{d_L \cdot j}(0) \\
\text{for } k = 1 \cdots n_B - 1 \quad \text{and} \quad \eta = 0 \cdots 5.
\end{gathered} \tag{3.14}$$

The fifth constraint is for obstacle avoidance of the suspended load which encodes $\|\boldsymbol{x}_L - \boldsymbol{x}_O\| \geq \delta_L$. The last two constraints are for obstacle avoidance of the multi-rotors and inter-agent collision avoidance between them which encodes $\|\boldsymbol{x}_i - \boldsymbol{x}_O\| \geq \delta_T$ and $\|\boldsymbol{x}_{i_1} - \boldsymbol{x}_{i_2}\| \geq \delta_I$, respectively. Note that, as shown (A.1), the position of the first multi-rotor requires the load acceleration, which is the reason why $\boldsymbol{g}_T$ and $\boldsymbol{g}_I$ depend on $\boldsymbol{c}_L$.

3.1.5 Decomposed Optimization

As shown in Problem 1, the cost functional is decoupled as two terms J_L and J_T. The boundary and continuity constraints are also separated, and the only coupled terms are $\boldsymbol{g}_T$ and $\boldsymbol{g}_I$. Then, this problem can be efficiently solved with two sub-problems after decomposition: load trajectory generation and tension history generation. The load trajectory for $\boldsymbol{c}_L$ is solved first and the tension histories for $\boldsymbol{c}_T$ are solved fixing $\boldsymbol{c}_L$. Since clearance constraints for the multi-rotors are considered in load trajectory generation, the loss of optimality from the decomposition is not critical.

3.2 Load Trajectory Generation

This section addresses a trajectory generation problem for the suspended load which is the first sub-problem of Problem 1. While all the cost functional, boundary constraint, and continuity constraint are convex functions of $\boldsymbol{c}_L$, the constraint for load obstacle avoidance, $g_L(\boldsymbol{c}_L)$, is not. For formulation of a convex optimization problem which can be easily solved by many solvers, the non-convex constraint is convexified by generating convex regions called safe flight corridors (SFCs) [43]. Since generation of SFCs requires initial guess of the trajectory, a sampling-based trajectory initialization method is proposed in the following subsection, and a convexification method and the resultant optimization problem are addressed.

3.2.1 Trajectory Initialization

Since a good initial guess of the trajectory is required to generate SFCs, an efficient sampling-based initialization is proposed which is shown in Algorithm 4.

Algorithm 4 Sampling-based trajectory initialization (Load)

```
Input: $\boldsymbol{x}_{init}$, $\boldsymbol{y}_{init}$, $d_B$, $n_B$, $n_{sample}$, $H$, $A_{KKT}$, $b_{KKT}(\boldsymbol{x}_W)$, $O$, $dt$
Output: $\boldsymbol{c}^*_{x.sample}$, $\boldsymbol{c}^*_{y.sample}$
 1: Set the sampling area
 2: for $k = 1 \cdots n_B - 1$ do
 3:     $\epsilon^k = c_{distance} * min\_distance\left(O, x^k_{init}, y^k_{init}\right)$
 4: end for
 5: Perform sampling
 6: $\bar{J}_{temp} = \infty$
 7: for $s = 1 \cdots n_{sample}$ do
 8:     for $k = 2 \cdots n_B - 1$ do
 9:         $x^k_{sample} = x^k_{init} + S^k$ where $S^k \sim U(-\epsilon^k, \epsilon^k)$
10:         $y^k_{sample} = y^k_{init} + S^k$ where $S^k \sim U(-\epsilon^k, \epsilon^k)$
11:     end for
12:     $\boldsymbol{c}_{x.sample} = A^{-1}_{KKT} b_{KKT}(\boldsymbol{x}_{sample})$
13:     $\boldsymbol{c}_{y.sample} = A^{-1}_{KKT} b_{KKT}(\boldsymbol{y}_{sample})$
14:     $\boldsymbol{x}_{traj} = compute\_trajectory(\boldsymbol{c}_{x.sample}, d_B, n_B, dt)$
15:     $\boldsymbol{y}_{traj} = compute\_trajectory(\boldsymbol{c}_{y.sample}, d_B, n_B, dt)$
16:     if not $check\_collision(\boldsymbol{x}_{traj}, \boldsymbol{y}_{traj}, O)$ then
17:         $\bar{J}_{x.temp} = \boldsymbol{c}^\top_{x.sample} H \boldsymbol{c}_{x.sample}$
18:         $\bar{J}_{y.temp} = \boldsymbol{c}^\top_{y.sample} H \boldsymbol{c}_{y.sample}$
19:         if $\bar{J}_{x.temp} + \bar{J}_{y.temp} < \bar{J}_{temp}$ then
20:             $\boldsymbol{c}^*_{x.sample} = \boldsymbol{c}_{x.sample}$
21:             $\boldsymbol{c}^*_{y.sample} = \boldsymbol{c}_{y.sample}$
22:             $\bar{J}_{temp} = \bar{J}_{x.temp} + \bar{J}_{y.temp}$
23:         end if
24:     end if
25: end for
```

The algorithm receives (1) centers of samples $\boldsymbol{x}_{init}$ and $\boldsymbol{y}_{init}$, (2) polynomial parameters d_B and n_B, (3) sampling number n_{sample}, (4) cost matrix H, (5) KKT system (A_{KKT}, $b_{KKT}(\boldsymbol{x}_W)$), (6) occupancy matrix O, and (7) time step dt. The centers of samples are set using A^* graph search method since it provides the shortest collision-free paths quickly. As shown in Fig. 3.1(b), A^* search results in paths with an arbitrary number of nodes which is drawn with asterisk markers. To fit the number of the nodes with the number of the joints of the polynomial segments, the A^* search result is uniformly interpolated to match with the number of polynomial segments which is depicted with triangle markers. The results $\boldsymbol{x}_{init}$, $\boldsymbol{y}_{init} \in \mathbb{R}^{n_B-1}$ are sent as one of the inputs. The degree of the polynomial and the cost matrix are set as $d_B = d_L$ and $H = H_L$.

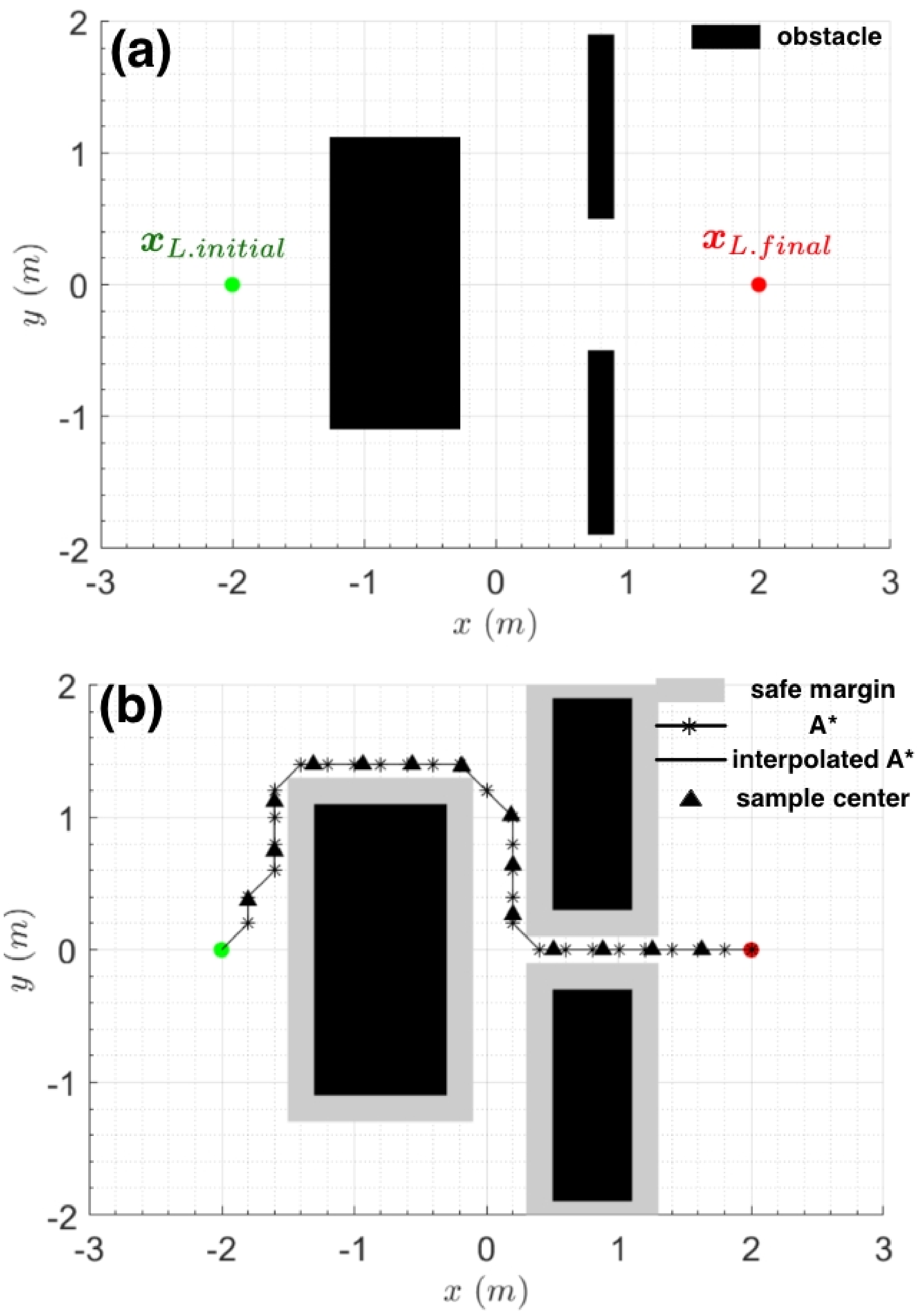


Figure 3.1: Procedures for trajectory generation of the suspended load. (a) The initial and desired positions of the load with the obstacle information. (b) A^* search result and the interpolated path with the safe margin shaded in gray.

Since assigning hard waypoints to the segment joints incurs high-cost samples, the trajectory samples are generated with soft waypoints. As an example, the cost functional for the x-component is defined as

$$\begin{aligned} &J_{Lx.w}(\boldsymbol{c}_{Lx}, \boldsymbol{x}_W) \\ &\quad = w \sum_{k=1}^{n_B-1} \left\{ \sum_{j=1}^{d_L+1} \left(c^k_{Lx.j} B_{d_L.j}(1) \right) - x^k_W \right\}^2 \\ &\quad = w \sum_{k=1}^{n_B-1} \left\{ c^k_{Lx.d_L+1} - x^k_W \right\}^2 \\ &\quad = w \boldsymbol{c}_{Lx}^\top M^\top M \boldsymbol{c}_{Lx} - 2w \boldsymbol{x}_W^\top M \boldsymbol{c}_{Lx} + w \boldsymbol{x}_W^\top \boldsymbol{x}_W, \end{aligned} \tag{3.15}$$

where $\boldsymbol{x}_W = [x^1_W \cdots x^{n_B-1}_W]^\top \in \mathbb{R}^{n_B-1}$ is the waypoints for the x-axis, and w is the weight parameter. $M \in \mathbb{R}^{(n_B-1)\times n_B(d_B+1)}$ is a mapping matrix from $\boldsymbol{c}_{Lx}$ to $[c^1_{Lx.d_L+1} \ \cdots \ c^{n_B}_{Lx.d_L+1}]^\top$. Since the last term in (3.15) is not related to the optimization variables, the resultant cost functional which is the sum of an input minimization term and the soft waypoint term can be expressed as a quadratic function of $\boldsymbol{c}_{Lx}$ as follows:

$$\boldsymbol{c}_{Lx}^\top (H_L + wM^\top M) \boldsymbol{c}_{Lx} - 2w \boldsymbol{x}_W^\top M \boldsymbol{c}_{Lx}. \tag{3.16}$$

When the KKT condition is applied to the cost functional with the affine boundary constraints, $A_{Lx.boundary}\boldsymbol{c}_{Lx} = \boldsymbol{b}_{Lx.boundary}$, and the continuity constraints, $A_{Lx.continuity}\boldsymbol{c}_{Lx} = b_{Lx.continuity}$, the optimization problem can be solved analytically without iteration using the KKT system as follows:

$$\begin{gathered} \begin{bmatrix} \boldsymbol{c}^*_{Lx} \\ \boldsymbol{\lambda}^*_{Lx} \end{bmatrix} = A^{-1}_{KKT} \boldsymbol{b}_{KKT}, \\ A_{KKT} = \begin{bmatrix} H_L + wM^\top M & A^\top_{Lx} \\ A_{Lx} & 0 \end{bmatrix}, \ \boldsymbol{b}_{KKT} = \begin{bmatrix} wM^\top \boldsymbol{x}_W \\ \boldsymbol{b}_{Lx} \end{bmatrix}, \end{gathered} \tag{3.17}$$

where $\boldsymbol{c}^*_{Lx}$ is the optimal solution, and $A_{Lx} = [A^\top_{Lx.boundary} \; A^\top_{Lx.continuity}]^\top$ and $\boldsymbol{b}_{Lx} = [\boldsymbol{b}^\top_{Lx.boundary} \; \boldsymbol{b}^\top_{Lx.continuity}]^\top$. The KKT systems for the x and y axes are sent to Algorithm 4 as a part of the inputs.

Given the input data, Algorithm 4 sets the sampling area using the A^* result $\boldsymbol{x}_{init}$, $\boldsymbol{y}_{init} \in \mathbb{R}^{n_B+1}$ (*lines 1-4*). The *min_distance* function computes the minimum distance between $[x^k_{init} \; y^k_{init}]^\top$ and the obstacles with a positive scaling factor $c_{distance}$ at the k^{th} segment. After that, the sampling is performed based on the sampling range, ϵ_k. With the uniformly distributed samples (*lines 9-10*), the control points minimizing (3.16) with the KKT system (*lines 12-13*) can be computed. The trajectory corresponding to the control points is easily computed using (3.8) with the fixed time step, dt (*lines 14-15*). Each trajectory is checked for collision, and if it is safe and has a lower overall cost compared with the saved minimum cost, J_{temp}, the new control points and the cost are stored (*lines 16-24*). The result of the algorithm is shown in Fig. 3.2.

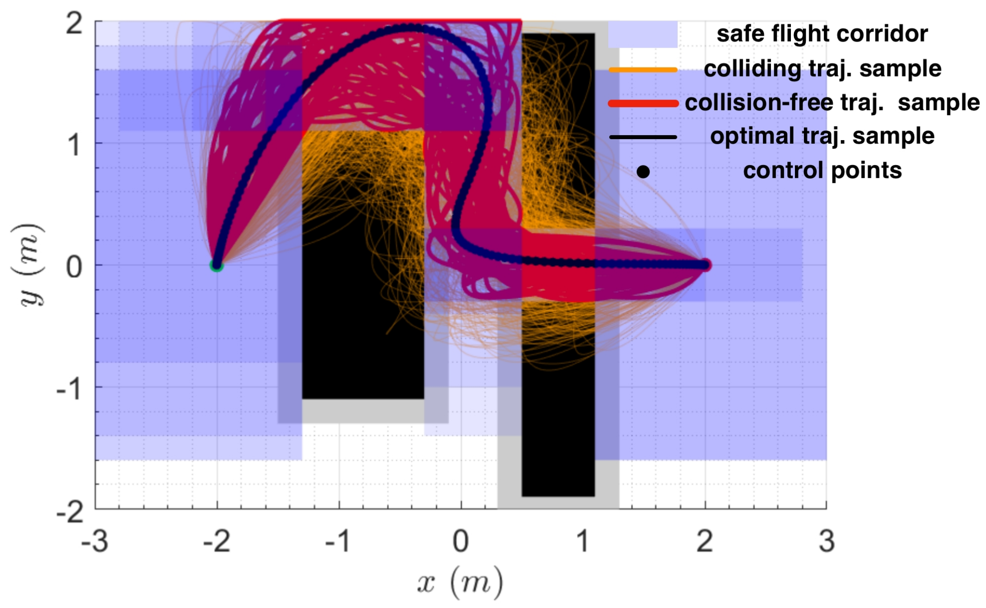


Figure 3.2: Sampling-based initialization. The trajectories where collision exist are depicted in translucent orange, and the collision-free trajectory samples are in solid red. The sample with the lowest cost is denoted in a black line with the circle control points. SFCs are generated based on the lowest-cost-sample and depicted in the blue area.

3.2.2 Convexification

Since the soft waypoint constraints in the sampling process deteriorate the quality of the trajectory by deviating it unnecessarily into the waypoints, it is necessary to optimize the result without the soft waypoints. The convexification method proposed in [43] is applied where safe flight corridors (SFCs) are generated for each segment of the position trajectory. The method convexifies the non-convex collision avoidance constraints, and a convex optimization problem is formulated to generate collision-free trajectories stably and fast. Fig. 3.2 shows the SFCs generated based on the initialized trajectory.

In addition to collision avoidance of the load, clearance for the multi-rotors needs to be considered since the acceleration of the load is decided by the tension vectors and the multi-rotors are located along the directions of the cables. $\hat{\boldsymbol{c}}^k_{Lx.acc}$ and $\hat{\boldsymbol{c}}^k_{Ly.acc}$ are defined as representations of the control points for the acceleration trajectory of each axis, and $\hat{\boldsymbol{c}}_{Lx.clearance}$ and $\hat{\boldsymbol{c}}_{Lx.clearnace}$ as the control points for the polynomials $x_L(t) + \frac{\delta_{max}}{a_{max}}\ddot{x}_L(t)$ and $x_L(t) + \frac{\delta_{max}}{a_{max}}\ddot{x}_L(t)$ where δ_{max} is the maximum horizontal displacement of the multi-rotor from the load computed in the following paragraph. As derived in Appendix B.4, the acceleration control points can be computed using the following equation:

$$\hat{\boldsymbol{c}}^k_{L*.acc} = S^{-1}_{B_{d_L-2}} S_{B^{(2)}_{d_L}} \boldsymbol{c}^k_{L*}, \tag{3.18}$$

and the control points for the clearance can be computed as

$$\hat{\boldsymbol{c}}^k_{L*.clearance} = \boldsymbol{c}^k_{L*} + \frac{\delta_{max}}{a_{max}} S^{-1}_{B_{d_L}} \begin{bmatrix} \boldsymbol{0}_2 \\ S_{B_{d_L-2}} \hat{\boldsymbol{c}}^k_{L*.acc} \end{bmatrix}. \tag{3.19}$$

Note that the equation assumes a linear relationship between displacement and acceleration. This approximation is acceptable since the collision avoidance constraints are handled tightly in Section 3.3.

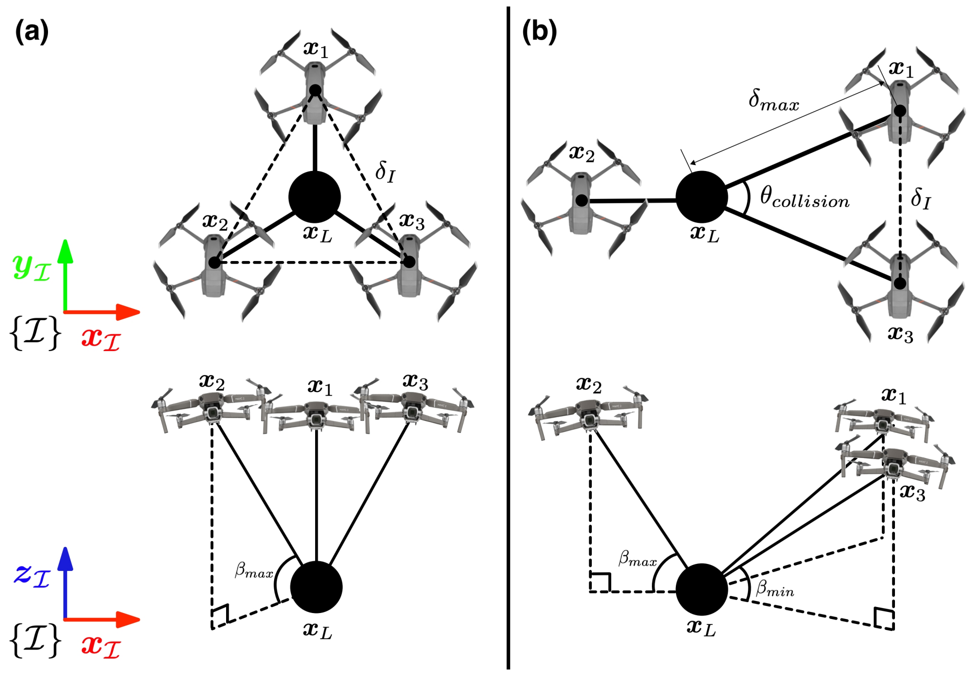


Figure 3.3: Multi-rotor configurations for computation of the clearance constraints.

The clearance can be set depending on the number of vehicles. A system with three multi-rotors is handled in this work as shown in Fig. 3.3. The maximum acceleration and tension boundaries are computed as the following procedures. First, the minimum vertical tension required to maintain hovering is computed which is $T_{min} = m_L g/(3 sin\beta_{max})$ with $\beta_{max} = acos(\delta_I/\sqrt{3}l)$. δ_I is the minimum distance between multi-rotors to avoid collision. This configuration is shown in Fig. 3.3(a). Second, a constraint on the minimum beta angle, β_{min}, is imposed since low beta angles result in very high tension to cancel out the gravitational force on the suspended load. To compute the maximum acceleration, two multi-rotors are located along the same direction with the minimum distance and the minimum beta angle, δ_I and β_{min}, while the other positioned in the opposite direction with the maximum beta angle as shown in Fig. 3.3(b). The magnitude of the tension is set to maintain constant altitude with the same contribution from each multi-rotor to the mass of the load. Then, the maximum tension in this configuration with the minimum beta angle is $T_{max} = m_L g/(3 sin\beta_{min})$. The maximum acceleration can also be computed as

$a_{max} = \left(2T_{max}cos\beta_{min}cos(\frac{\theta_{collision}}{2}) - T_{min}cos\beta_{max}\right)/m_L$ where $\theta_{collision}$ is the angle between two cables when the multi-rotors are placed with the distance δ_I. Finally, the maximum displacement of the multi-rotors from the load in this configuration is $\delta_{max} = lcos\beta_{min}$.

3.2.3 Optimization sub-problem

With the consideration of the SFCs and the clearance constraints, the resultant subproblem is formulated as follows:

Problem 2. *Load optimization*

$$\begin{aligned}
\text{find} \quad & \boldsymbol{c}_L^* = \underset{\boldsymbol{c}_L}{\text{argmin}}\ J_L(\boldsymbol{c}_L) \\
\text{subject to} \quad & A_{L.boundary}\boldsymbol{c}_L = \boldsymbol{b}_{L.boundary}, \\
& A_{L.continuity}\boldsymbol{c}_L = \boldsymbol{b}_{L.continuity}, \\
& \underline{\boldsymbol{x}}^k_{SFC} \preceq \boldsymbol{c}^k_{Lx} \preceq \overline{\boldsymbol{x}}^k_{SFC} \qquad \text{for} \quad k = 1, \cdots, n_B, \\
& \underline{\boldsymbol{y}}^k_{SFC} \preceq \boldsymbol{c}^k_{Ly} \preceq \overline{\boldsymbol{y}}^k_{SFC} \qquad \text{for} \quad k = 1, \cdots, n_B, \\
& -a_{max} \preceq \hat{\boldsymbol{c}}^k_{Lx.acc} \preceq a_{max} \qquad \text{for} \quad k = 1, \cdots, n_B, \\
& -a_{max} \preceq \hat{\boldsymbol{c}}^k_{Ly.acc} \preceq a_{max} \qquad \text{for} \quad k = 1, \cdots, n_B, \\
& \underline{\boldsymbol{x}}^k_{SFC} \preceq \hat{\boldsymbol{c}}^k_{Lx.clearance} \preceq \overline{\boldsymbol{x}}^k_{SFC} \\
& \qquad\qquad\qquad\qquad\qquad \text{for} \quad k = 1, \cdots, n_B, \\
& \underline{\boldsymbol{y}}^k_{SFC} \preceq \hat{\boldsymbol{c}}^k_{Ly.clearance} \preceq \overline{\boldsymbol{y}}^k_{SFC} \\
& \qquad\qquad\qquad\qquad\qquad \text{for} \quad k = 1, \cdots, n_B,
\end{aligned}$$

where $\preceq$ is the generalized inequality for element-wise comparison.

The first two constraints are the boundary and continuity constraints. In the third and fourth constraints, $\underline{\boldsymbol{x}}^k_{SFC}$ and $\overline{\boldsymbol{x}}^k_{SFC}$ are the x-axis boundaries of the SFC for the k^{th} segment, and the y-axis boundaries are indicated by $\underline{\boldsymbol{y}}^k_{SFC}$ and $\overline{\boldsymbol{y}}^k_{SFC}$. The fifth and sixth constraints are about the maximum acceleration of the load. The last two constraints denote the clearance requirements for the multi-rotors with the threshold value a_{max}.

Fig. 3.4 shows the resultant trajectory of Problem 2, $\boldsymbol{x}^*_L(t)$ and $\boldsymbol{y}^*_L(t)$ which can be computed using $\boldsymbol{c}_L^*$ after optimization. The arrow denotes the normalized acceleration, $\frac{\delta_{max}}{a_{max}}\ddot{\boldsymbol{x}}$, and the clearance for the multi-rotors is secured in the direction of the acceleration. The z-axis trajectory, $\boldsymbol{z}_L^*(t)$ can be easily computed using the KKT system in (3.17) with the boundary and continuity constraints in addition to setting $w = 0$. For flight area constraints like positiveness or maximum of the altitude, the generated SFCs can be expanded or shrank along the z direction.

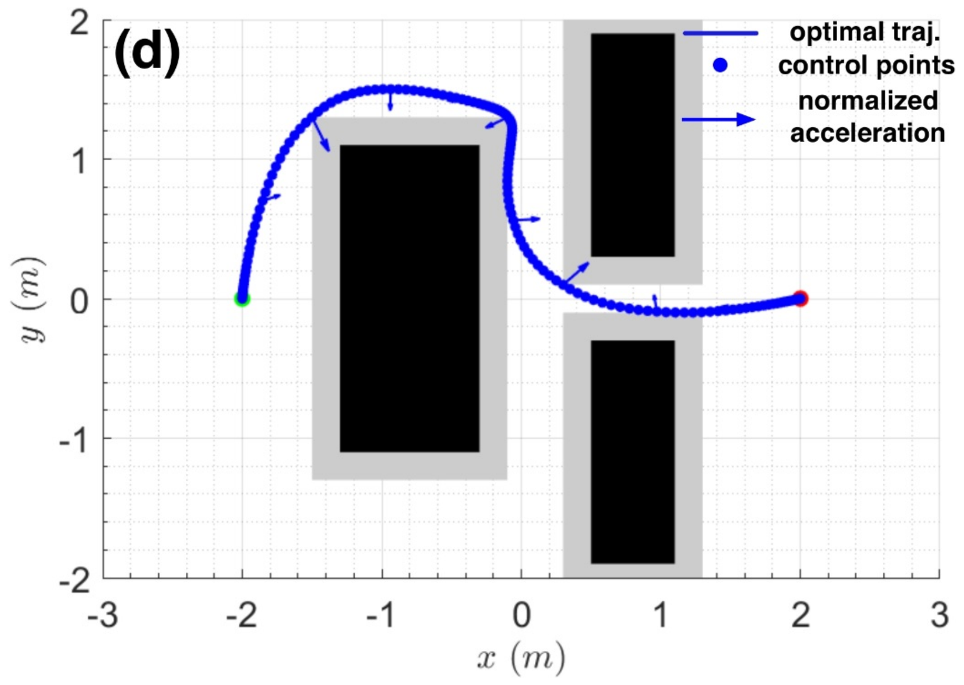


Figure 3.4: Optimized trajectory with the consideration of the clearance for the multi-rotors and maximum acceleration constraints.

3.3 Tension History Generation

Now that optimization variables for the load are optimized in the previous subsection, the remaining variables are for the cable tension histories. Not only the minimization of the cost but also the obstacle avoidance of the multi-rotors and inter-agent collision between the vehicles should be considered in this step.

Since the optimization variables do not include terms about the first multi-rotor, the tension history for the first one is expressed with the Bernstein polynomials of the order $d_L - 2$ based on (A.1), and the k^{th} segment trajectory can be expressed as follows:

$$T_{1*}(t) = \sum_{j=1}^{d_L-1} \hat{c}^k_{1*.j} B_{d_L-2.j}(\tau_k) \quad \text{for} \quad t \in [t_{k-1}, t_k] \tag{3.20}$$

where $*$ denotes the axes. The new control points $\hat{\boldsymbol{c}}^k_{1x}$, $\hat{\boldsymbol{c}}^k_{1y}$, and $\hat{\boldsymbol{c}}^k_{1z}$ can be computed as follows based on (A.1) and Appendix B.3:

$$\begin{aligned}
\hat{\boldsymbol{c}}^k_{1x} &= m_L \hat{\boldsymbol{c}}^k_{Lx.acc} - S^{-1}_{B_{d_L-2}} \begin{bmatrix} \mathbf{0}_2 \\ S_{B_{d_T}} \sum_{i=2}^{n_M} \boldsymbol{c}^k_{ix} \end{bmatrix}, \\
\hat{\boldsymbol{c}}^k_{1y} &= m_L \hat{\boldsymbol{c}}^k_{Ly.acc} - S^{-1}_{B_{d_L-2}} \begin{bmatrix} \mathbf{0}_2 \\ S_{B_{d_T}} \sum_{i=2}^{n_M} \boldsymbol{c}^k_{iy} \end{bmatrix}, \\
\hat{\boldsymbol{c}}^k_{1z} &= m_L \hat{\boldsymbol{c}}^k_{Lz.acc} - S^{-1}_{B_{d_L-2}} \left(\begin{bmatrix} \mathbf{0}_2 \\ S_{B_{d_T}} \sum_{i=2}^{n_M} \boldsymbol{c}^k_{iz} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{d_L-2} \\ m_L g \end{bmatrix} \right).
\end{aligned} \tag{3.21}$$

3.3.1 History Initialization

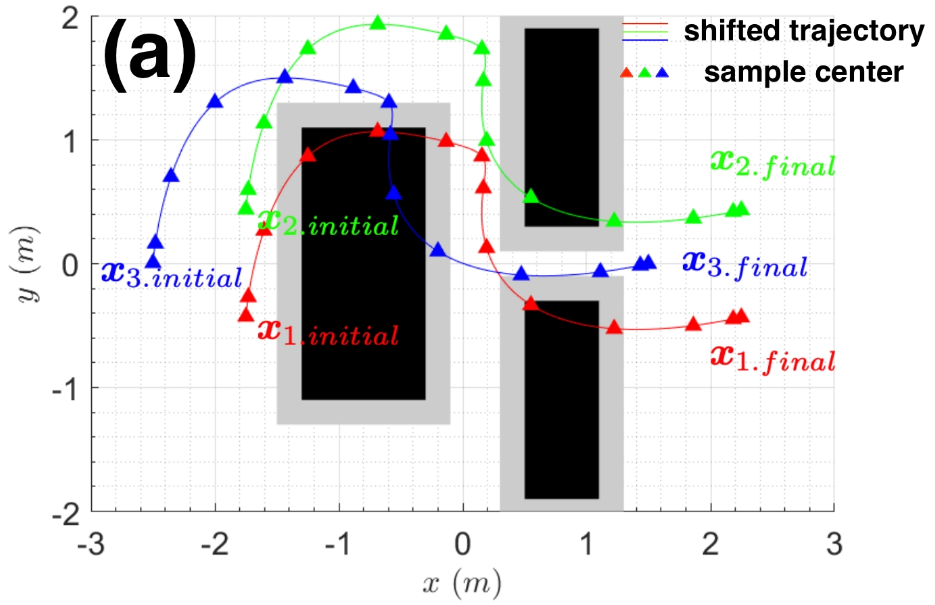
As in Section 3.2.1, initialization is also required here for convexification of non-convex collision avoidance constraints. Algorithm 5 shows the initialization process for the tension histories where some steps are added compared with Algorithm 4 to check inter-agent collision of the samples.

Algorithm 5 Sampling-based history initialization (Tension)

```
Input: $X_{init}$, $Y_{init}$, $d_B$, $n_B$, $n_{sample}$, $H$, $A_{KKT}$, $b_{KKT}(\boldsymbol{x}_W)$, $O$, $dt$
Output: $\hat{\boldsymbol{c}}_{1.sample}$, $\boldsymbol{c}^*_{T.sample}$
 1: Set the sampling area
 2: for $i = 2 \cdots n_M$ do
 3:     for $k = 1 \cdots n_B - 1$ do
 4:         $\epsilon_i^k = c_{distance} * min\_distance\left(O, x^k_{i.init}, y^k_{i.init}\right)$
 5:     end for
 6: end for
 7: Perform sampling
 8: $J_{temp} = \infty$
 9: for $s = 1 \cdots n_{sample}$ do
10:     for $i = 2 \cdots n_M$ do
11:         for $k = 2 \cdots n_B - 1$ do
12:             $x^k_{i.sample} = x^k_{i.init} + S^k_i$ where $S^k_i \sim U(-\epsilon^k_i, \epsilon^k_i)$
13:             $y^k_{i.sample} = y^k_{i.init} + S^k_i$ where $S^k_i \sim U(-\epsilon^k_i, \epsilon^k_i)$
14:             $z^k_{i.sample} = z^k_{nominal} + S_z$ where $S_z \sim U(-\epsilon_z, \epsilon_z)$
15:         end for
16:         $\boldsymbol{T}_{i.sample} = xyz\_to\_T(\boldsymbol{x}_{i.sample}, \boldsymbol{y}_{i.sample}, \boldsymbol{z}_{i.sample})$
17:         $\boldsymbol{c}_{i*.sample} = A^{-1}_{KKT} b_{KKT}(\boldsymbol{T}_{i*.sample})$
18:         $\boldsymbol{T}_{i.traj} = compute\_history(\boldsymbol{c}_{i.sample}, d_B, n_B, dt)$
19:     end for
20:     $\hat{\boldsymbol{c}}_{1.sample} = first\_tension(\hat{\boldsymbol{c}}_{L.acc}, \boldsymbol{c}_{2.sample},$
21:                 $\boldsymbol{c}_{3.sample}, \cdots, \boldsymbol{c}_{n_M.sample})$
22:     $\boldsymbol{T}_{1.traj} = compute\_history(\hat{\boldsymbol{c}}_{1.sample}, d_B, n_B, dt)$
23:     if not $check\_collision(\boldsymbol{T}_{1.sample}, \cdots, \boldsymbol{T}_{n_M.sample}, O)$ then
24:         for $i = 2 \cdots n_M$ do
25:             $J_{i*.temp} = \boldsymbol{c}^\top_{i*.sample} H \boldsymbol{c}_{i*.sample}$
26:             $J_{i.temp} = J_{ix.temp} + J_{iy.temp} + J_{iz.temp}$
27:         end for
28:         if $\sum_{i=2}^{n_M} J_{i.temp} < J_{temp}$ then
29:             $\hat{\boldsymbol{c}}^*_{1.sample} = \hat{\boldsymbol{c}}_{1.sample}$
30:             for $i = 2 \cdots n_M$ do
31:                 $\boldsymbol{c}^*_{i.sample} = \boldsymbol{c}_{i.sample}$
32:             end for
33:             $J_{temp} = \sum_{i=2}^{n_M} J_{i.temp}$
34:             $\boldsymbol{c}^*_{T.sample} = \left[\boldsymbol{c}^\top_{2.sample}\ \boldsymbol{c}^\top_{3.sample}\ \cdots\ \boldsymbol{c}^\top_{n_M.sample}\right]^\top$
35:         end if
36:     end if
37: end for
```

While performing A^* search for the multi-rotors is possible to get the centers of samples, the optimal result for the load computed in Problem 2 is shifted since it has been optimized with the clearance constraints. Additionally, A^* often results in partially overlapped paths despite different start and goal points. The shifted trajectories starting from initial positions of each multi-rotor are shown in Fig. 3.5(a), and $x-$ and $y-$axis trajectories are denoted as $X_{init} = \{\boldsymbol{x}_{i.init} \in \mathbb{R}^{n_B - 1} \mid i = 2 \cdots n_M\}$ and $Y_{init} = \{\boldsymbol{y}_{i.init} \in \mathbb{R}^{n_B - 1} \mid i = 2 \cdots n_M\}$.

Since the position sample does not satisfy the distance constraint between the load and the multi-rotor, only the direction information is extracted. The *xyz_to_T* function converts the position sample to a tension sample, the direction of which is parallel to the relative position of the vehicle with respect to the load (*line 16*). The magnitude is set to cancel out $m_L g / n_M$ in the z-axis. Using (3.21), the control points $\boldsymbol{c}_1$ and the history for the first multi-rotor can be computed (*lines 20-22*). After that, the collision check with the obstacles and the inter-agent collision check is performed (*lines 23*). The result of the algorithm is shown in Fig. 3.5(b).



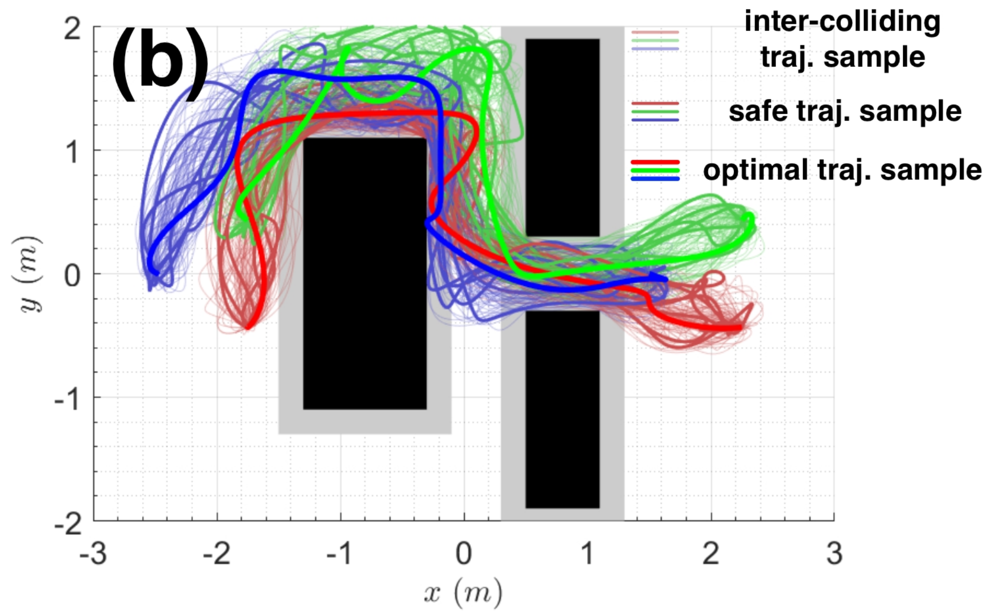


Figure 3.5: Procedures for trajectory initialization of the multi-rotors. The trajectories for the multi-rotors are represented by the colors in a RGB order. (a) The shifted trajectories from the optimized load trajectory are used for the sampling centers. (b) Sampling-based initialization where the trajectory samples with inter-agent collision are depicted in translucent color, and the collision-free trajectory samples are in solid colors. The sample with the lowest cost is denoted in the thickest lines.

3.3.2 Convexification

Similar to the load trajectory optimization, the sampled tension histories are also not efficient due to the soft waypoint constraints. While the SFC construction is an efficient convexification method for a point mass, it cannot be applied to this sub-problem since the multi-rotor position is decided by adding the normalized unit tension vector to the load position as $\boldsymbol{x}_i = \boldsymbol{x}_L + l\boldsymbol{T}_i/\|\boldsymbol{T}_i\|$, which is not a convex function of the flat outputs.

SFS generation

The collision avoidance constraints between the vehicles and the obstacles are convexified using safe flight sector (SFS) construction. For SFS generation, the relative obstacle position with respect to the load position is accumulated, and the collision region for the k^{th} segment is defined for each time segment as the following set:

$$X_O^k = \{\boldsymbol{x}_O(t) - \boldsymbol{x}_L(t) \mid t \in [t_{k-1}, t_k], \|\boldsymbol{x}_O(t) - \boldsymbol{x}_L(t)\| \leq l\}. \tag{3.22}$$

Some examples of X_O^k for $k = 3, 6, 9, 12$ are shown in Fig. 3.6(b) as blue markers. Then, collision avoidance for the i^{th} multi-rotor during the k^{th} time segment is satisfied if the relative position trajectory segment is located in the collision-free region, i.e. $\big(\boldsymbol{x}_i(t) - x_L(t)\big) \subset X_{Free}^k = S_l - X_O^k$ for $t \in [t_{k-1}, t_k]$ where S_l is a sphere centered at the origin with the radius l.

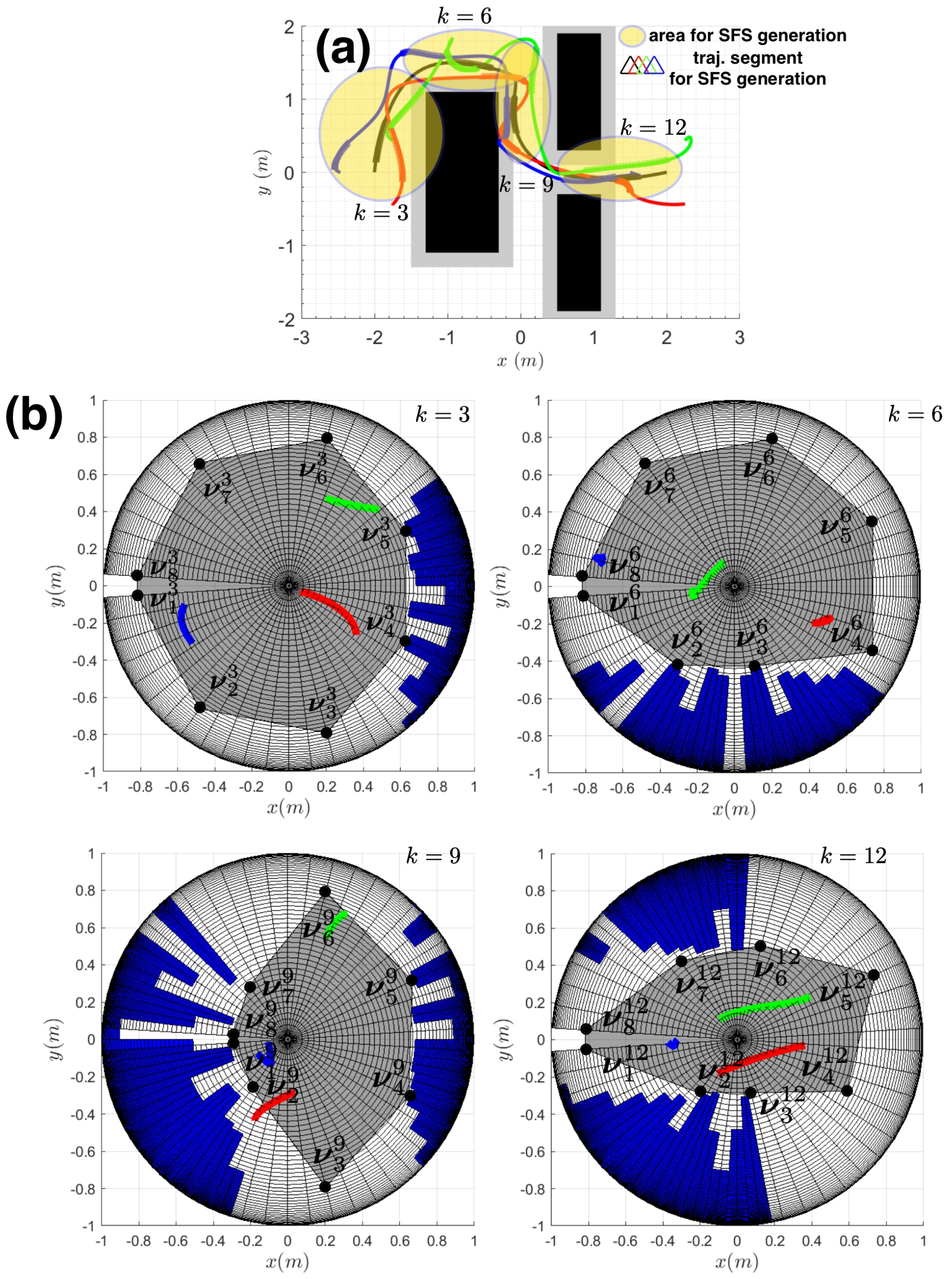


Figure 3.6: SFS generation results. (a) The lowest-cost-trajectories from the initialization are denoted in RGB lines, and the yellow area denotes the area of the trajectory segments for SFS generation. (b) Some examples of the generated SFSs are depicted as the gray area where the blue area denotes the obstacles. RGB lines are the parts of trajectory segments for each time segment.

Since the collision-free sets X^k_{Free} are not convex, they are convexified by constructing SFSs. The construction procedure is shown in Fig. 3.7.

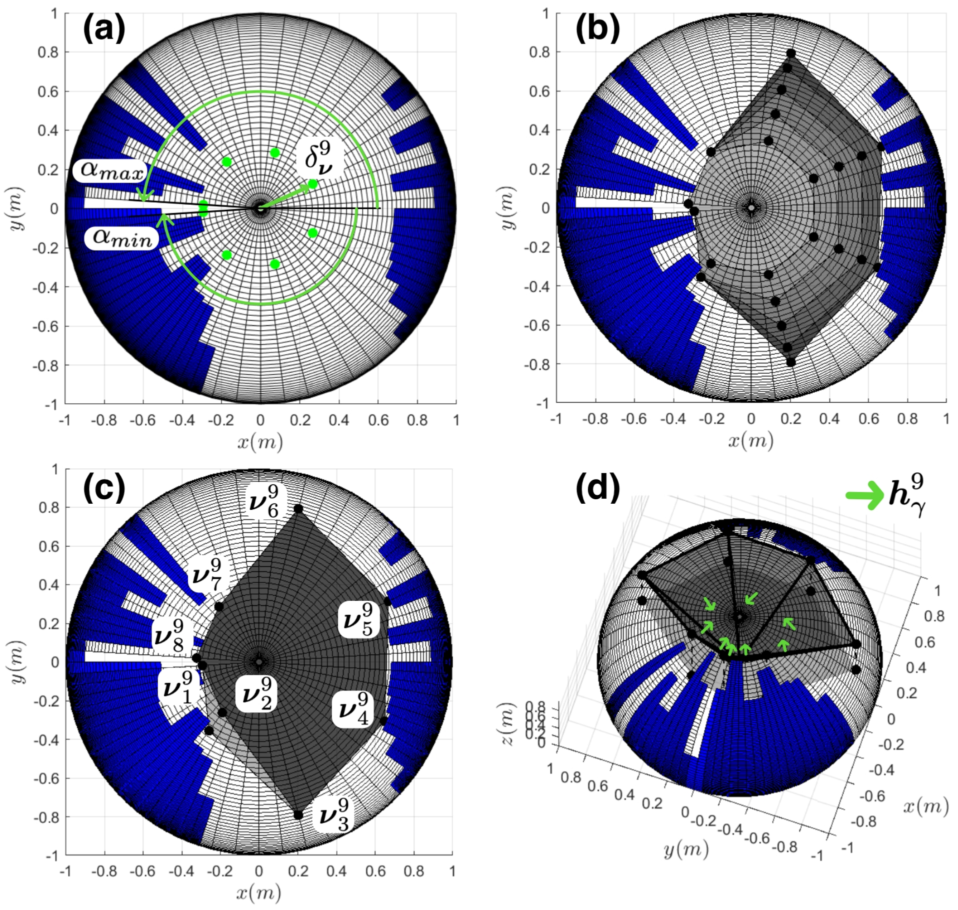


Figure 3.7: SFS construction procedure example for the 9th segment. (a) Initialization with the maximum radius without collision. (b) The vertex expansion process where the darker areas are the latter result. (c) The vertex shrinkage process to make the polyhedron convex where the darker area is the latter result. (d) The resultant collision-free convex polyhedron where the green arrows are the normal vectors of the triangular surfaces.

The number of the vertices $n_{\boldsymbol{\nu}} = 8$ is decided as a user-defined parameter. The horizontal positions of the vertices are initialized as far away as possible from the center of the sphere as follows:

$$\begin{aligned} &\boldsymbol{\nu}_\gamma^k(1) = \delta_{\boldsymbol{\nu}}^k cos\alpha_\gamma, \\ &\boldsymbol{\nu}_\gamma^k(2) = \delta_{\boldsymbol{\nu}}^k sin\alpha_\gamma, \qquad\qquad \text{for} \begin{cases} k = 1, \cdots, n_B, \\ \gamma = 1, \cdots, n_{\boldsymbol{\nu}}, \end{cases} \\ &\alpha_\gamma = \alpha_{min} + \frac{\alpha_{max} - \alpha_{min}}{n_{\boldsymbol{\nu}} - 1}(\gamma - 1) \end{aligned} \tag{3.23}$$

which is shown in Fig. 3.7(a).

The vertices are expanded to the outward direction of the circle one by one to maximize the area of SFSs. After each expansion, collision is checked along the newly generated polygon made of the vertices. If no collision is detected, the expanded vertex is saved, and the expansion proceeds until the polygon cannot expand more without collision. The expansion procedure is shown in Fig. 3.7(b) where the darker polygons are the broader ones after each expansion. The vertices are projected onto the sphere to generate a polyhedron by setting the z-axis value as $\boldsymbol{\nu}_\gamma^k(3) = lsin\beta_\gamma^k$ with $\beta_\gamma^k = acos\left(\sqrt{\boldsymbol{\nu}_\gamma^k(1)^2 + \boldsymbol{\nu}_\gamma^k(2)^2}/l\right)$. The vertices of the polyhedron are composed of the projected ones and the center of the sphere.

Since the polyhedron is not guaranteed to be a convex hull, a post processing is performed to turn it into a convex set. In contrast to the vertex-expansion, the vertices are shrunk at each step if the adjacent edges hinder convexity of the polyhedron. The shrinking process is explained in detail in Algorithm 6 where the input and output data is defined as $V = \{\boldsymbol{\nu}_\gamma^k \in \mathbb{R}^3 \mid k = 1, \cdots, n_B,\ \gamma = 1 \cdots n_{\boldsymbol{\nu}}\}$, $A_\alpha = \{\alpha_\gamma \mid \gamma = 1, \cdots, n_{\boldsymbol{\nu}}\}$, and $B_\beta = \{\beta_\gamma^k \mid k = 1, \cdots, n_B,\ \gamma = 1 \cdots n_{\boldsymbol{\nu}}\}$. Two new slack vertices $\boldsymbol{\nu}_0^k$ and $\boldsymbol{\nu}_{n_\nu+1}^k$ are added for computational convenience (*lines 2-3*). For each vertex, two edges $\boldsymbol{g}_{prev}$ and $\boldsymbol{g}_{next}$ are computed (*lines 7-9*), and the cross product of them, $\boldsymbol{g}_{normal}$, is used to check the convexity (*lines 10-12*). When it is not convex, the vertex with a lower β is moved toward the center of the sphere (*lines 14-26*). This process is repeated until the polyhedron becomes a convex set. Fig. 3.7(c) shows the result of the shrinking process where the darker area is the later result. The resultant convex hull is shown in Fig. 3.7(d), and it is guaranteed that the multi-rotors avoid collision when they are located above the triangular surfaces of the polyhedron. The convex set below the sphere and above the triangular surfaces is called as a SFS in this dissertation.

Algorithm 6 Vertex-processing for convex set generation

```
Input: $V$, $A_\alpha$, $B_\beta$, $n_B$, $n_{\boldsymbol{\nu}}$
Output: $V$, $B_\beta$
 1: for $k = 1 \cdots n_B$ do
 2:     $\boldsymbol{\nu}_0^k = \boldsymbol{\nu}_{n_\nu}^k$
 3:     $\boldsymbol{\nu}_{n_\nu+1}^k = \boldsymbol{\nu}_1^k$
 4:     $is\_convex = false$
 5:     for $\gamma = 1 \cdots n_{\boldsymbol{\nu}}$ do
 6:         while not $is\_convex$ do
 7:             Compute the edge vectors
 8:             $\boldsymbol{g}_{prev} = \boldsymbol{\nu}_\gamma^k - \boldsymbol{\nu}_{\gamma-1}^k$
 9:             $\boldsymbol{g}_{next} = \boldsymbol{\nu}_{\gamma+1}^k - \boldsymbol{\nu}_\gamma^k$
10:             Check the convexity
11:             $\boldsymbol{g}_{normal} = \boldsymbol{g}_{prev} \times \boldsymbol{g}_{next}$
12:             if $\boldsymbol{g}_{normal}(3) < 0$ then
13:                 $is\_convex = false$
14:                 Shrink one vertex with a lower $\beta$ angle
15:                 if $\beta_{\gamma-1}^k < \beta_{\gamma+1}^k$ then
16:                     $\beta_{\nu-1}^k = \beta_{\nu-1}^k + \Delta\beta$
17:                     $\boldsymbol{\nu}_{\gamma-1}^k(1) = lcos\beta_{\gamma-1}^k cos\alpha_\gamma$
18:                     $\boldsymbol{\nu}_{\gamma-1}^k(2) = lcos\beta_{\gamma-1}^k sin\alpha_\gamma$
19:                     $\boldsymbol{\nu}_{\gamma-1}^k(3) = lsin\beta_{\gamma-1}^k$
20:                 else
21:                     $\beta_{\gamma+1}^k = \beta_{\gamma+1}^k + \Delta\beta$
22:                     $\beta_{\gamma+1}^k = \beta_{\gamma-1}^k + \Delta\beta$
23:                     $\boldsymbol{\nu}_{\gamma+1}^k(1) = lcos\beta_{\gamma+1}^k cos\alpha_\gamma$
24:                     $\boldsymbol{\nu}_{\gamma+1}^k(2) = lcos\beta_{\gamma+1}^k sin\alpha_\gamma$
25:                     $\boldsymbol{\nu}_{\gamma+1}^k(3) = lsin\beta_{\gamma+1}^k$
26:                 end if
27:             else if $\gamma == n_{\boldsymbol{\nu}}$ then
28:                 $is\_convex = true$
29:             end if
30:         end while
31:     end for
32: end for
```

Placing the multi-rotors in the SFS can be achieved by manipulating the tension vectors. The normal vectors of the triangular surfaces pointing toward the center of the sphere are defined as

$$\boldsymbol{h}_{\gamma}^{k} = \frac{\boldsymbol{\nu}_{\gamma}^{k} \times \boldsymbol{\nu}_{\gamma+1}^{k}}{\|\boldsymbol{\nu}_{\gamma}^{k} \times \boldsymbol{\nu}_{\gamma+1}^{k}\|} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B, \\ \gamma = 1, \cdots, n_{\boldsymbol{\nu}}, \end{cases} \tag{3.24}$$

which can be easily computed using the vertices. Then, the collision avoidance constraints for the k^{th} segment of the tension vector multi-rotor can be expressed as follows:

$$\begin{bmatrix} c_{ix.j}^{k} & c_{iy.j}^{k} & c_{iz.j}^{k} \end{bmatrix}^{\top} \boldsymbol{h}_{\gamma}^{k} \geq 0 \quad \text{for} \quad \begin{cases} i = 2, \cdots, n_M, \\ \gamma = 1, \cdots, n_{\boldsymbol{\nu}}, \\ j = 1, \cdots, d_T + 1, \end{cases} \tag{3.25}$$

which means that all the tension vectors should be located in the generated SFSs.

Since the control points for the first multi-rotor can be expressed as affine functions of the control points of other vehicles as shown in (3.21), the following inequalities complete the collision avoidance constraint for the multi-rotors:

$$\begin{bmatrix} \hat{c}_{1x.j}^{k} & \hat{c}_{1y.j}^{k} & \hat{c}_{1z.j}^{k} \end{bmatrix}^{\top} \boldsymbol{h}_{\gamma}^{k} \geq 0 \quad \text{for} \quad \begin{cases} \gamma = 1, \cdots, n_{\boldsymbol{\nu}}, \\ j = 1, \cdots, d_L - 1. \end{cases} \tag{3.26}$$

RSFS generation

The last constraint to be considered for the safety is inter-agent collision avoidance between the multi-rotors. Since the multi-rotor positions cannot be expressed with the optimization variables as previously described, RSFSs are constructed using the relative tension history defined as follows:

$$\boldsymbol{T}^k_{i_1,i_2} = \boldsymbol{T}^k_{i_2} - \boldsymbol{T}^k_{i_1} \quad \text{for} \quad \begin{cases} i_1 = 1, \cdots, n_M - 1, \\ i_2 = i_1 + 1, \cdots, n_M, \end{cases} \tag{3.27}$$

which is a relative tension history of the i_2^{th} multi-rotor with respect to the i_1^{th} multi-rotor during the k^{th} time segment. As an example, the x-component trajectory of the k^{th} segment can be expressed using the tension control points as follows:

$$T^k_{i_1x,i_2x}(t) = \begin{cases} \sum_{j=1}^{d_T+1}(c^k_{i_1x,i_2x.j})B_{d_T.j}(\tau_k) & \text{if } i_1 \neq 1, \\ \sum_{j=1}^{d_L-1}(\hat{c}^k_{1x,i_2x.j})B_{d_L-2.j}(\tau_k) & \text{otherwise}, \end{cases} \tag{3.28}$$

where $c^k_{i_1x,i_2x.j} = c^k_{i_2x.j} - c^k_{i_1x.j}$. Note that $\hat{\boldsymbol{c}}^k_{1*,i_2*} = [\hat{c}^k_{1*,i_2*.1} \ \cdots \ \hat{c}^k_{1*,i_2*.d_L-1}]^\top \in \mathbb{R}^{d_L-1}$ can be computed similarly to (3.21). The x-axis trajectory, as an example, is shown in the following equation:

$$\hat{\boldsymbol{c}}^k_{1x,i_2x} = S^{-1}_{B_{d_L-2}} \begin{bmatrix} \boldsymbol{0}_{2\times 1} \\ S_{B_{d_T}} \boldsymbol{c}^k_{i_2x} \end{bmatrix} - \hat{\boldsymbol{c}}^k_{1x}. \tag{3.29}$$

To compute the RSFSs, the safe and collision regions are computed based on the system parameters firstly, which are shown in Fig. 3.8(a) with the parameters $T_{min} = 4.531$N, $T_{max} = 10.572$N, and $\theta_{collision} = 25.840°$. In the figure, the green arrow denotes the tension vector for the first multi-rotor, $\boldsymbol{T}_1$. The angle between the first and second tension vectors must be at least $\theta_{collision}$ to avoid inter-agent collision, which results in the blue safe region and the red collision region as shown. The RSFS computation procedures are shown in Fig. 3.8(b). The blue feasible set of $\boldsymbol{T}_{i_1,i_2}$ can be computed by changing the β of the first

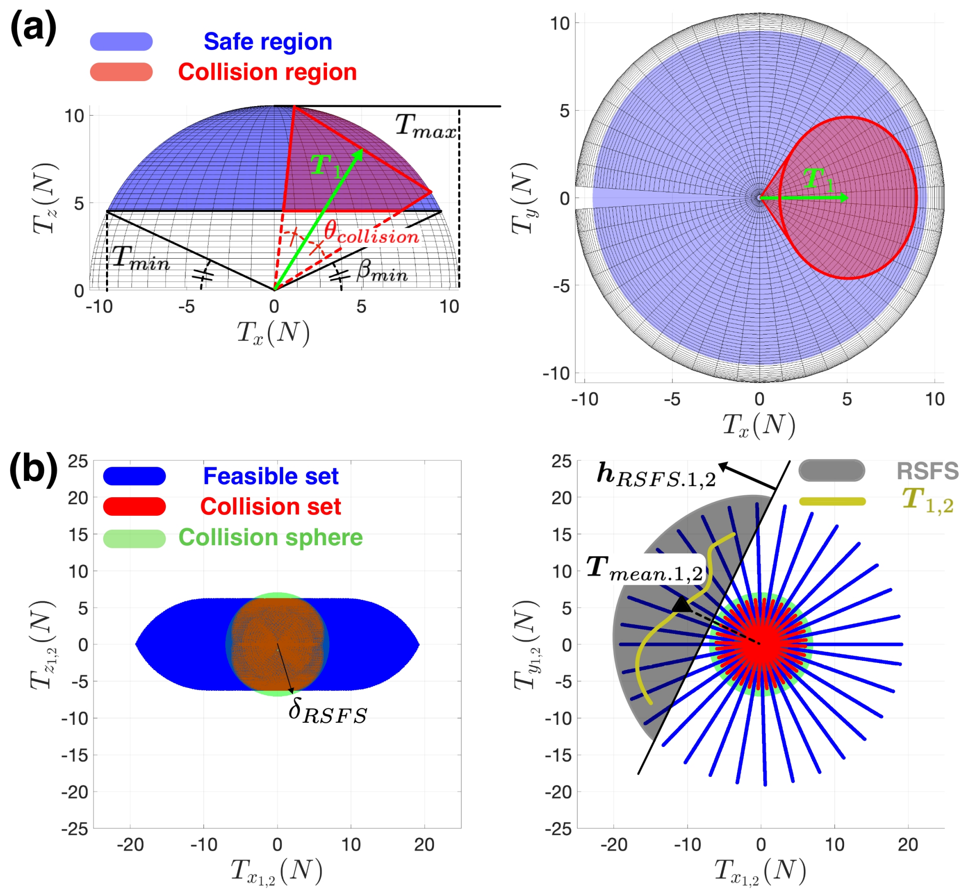


Figure 3.8: Computation procedures of the RSFS. (a) The safe and collision regions of $\boldsymbol{T}_2$ which are depicted in blue and red, respectively. (b) The feasible and collision sets of $\boldsymbol{T}_{1,2}$. The RSFS is computed based on the initialized relative tension history, $\boldsymbol{T}_{1,2}$, in gold.

tension from β_{min} to $\pi - \beta_{min}$ and accumulating the feasible values of the second tension. The red collision set, a subset of the feasible set is computed by accumulating the collision region. The inter-agent collision avoidance can be guaranteed if $\boldsymbol{T}_{i_1,i_2}(t)$ is located in the blue feasible set. Note that, for computation of y-component, the x- and z-component set is rotated with respect to the z-axis. The rotation is discretely performed to show the sets more clear. Since the collision set is not convex, a collision sphere of radius δ_{RSFS} is used to encompass the set, which is depicted in green.

With the computed collision sphere, the RSFS for the k^{th} segment can be convexified and computed as follows:

$$\begin{aligned}\boldsymbol{h}^k_{RSFS.1,2} &= \boldsymbol{T}^k_{1,2.mean} / \left\|\boldsymbol{T}^k_{mean.1,2}\right\|, \\ \boldsymbol{T}^k_{mean.1,2} &= \frac{1}{(t_k - t_{k-1})} \int_{t_{k-1}}^{t_k} \boldsymbol{T}_{1,2}(t) dt,\end{aligned} \tag{3.30}$$

where the project-and-linearize method in [14] is used for the convexification. Then, the inter-agent collision constraints for the k^{th} segment are imposed using the control points as

$$\begin{aligned}&(\boldsymbol{c}^k_{i_1,i_2.j})^\top \boldsymbol{h}^k_{RSFS.i_1,i_2} \geq \delta_{RSFS} \\ &\qquad\qquad \text{for} \begin{cases} i_1 = 2, \cdots, n_M - 1, \\ i_2 = i_1 + 1, \cdots, n_M, \\ j = 1, \cdots, d_T + 1, \end{cases} \\ &(\hat{\boldsymbol{c}}^k_{1,i_2.j})^\top \boldsymbol{h}^k_{RSFS.1,i_2} \geq \delta_{RSFS} \\ &\qquad\qquad \text{for} \begin{cases} i_2 = 2, \cdots, n_M, \\ j = 1, \cdots, d_L - 1, \end{cases}\end{aligned} \tag{3.31}$$

where $\boldsymbol{c}^k_{i_1,i_2.j} = \left[c^k_{i_1x,i_2x.j}\ c^k_{i_1y,i_2y.j}\ c^k_{i_1z,i_2z.j}\right]^\top$ and $\hat{\boldsymbol{c}}^k_{1,i_2.j} = \left[\hat{c}^k_{1x,i_2x.j}\ \hat{c}^k_{1y,i_2y.j}\ \hat{c}^k_{1z,i_2z.j}\right]^\top$.

3.3.3 Optimization sub-problem

With the computed constraints for collision avoidance, the second sub-problem of Problem 1 is expressed as Problem 3.

Problem 3. *Tension optimization (discrete)*

$$
\begin{aligned}
&\textit{find} && \boldsymbol{c}_T^* = \underset{\boldsymbol{c}_T}{\text{argmin}}\ J_T(\boldsymbol{c}_T)\\
&\textit{subject to} && A_{T.boundary}\boldsymbol{c}_T = \boldsymbol{b}_{T.boundary},\\
& && A_{T.continuity}\boldsymbol{c}_T = \boldsymbol{b}_{T.continuity},\\
& && \begin{bmatrix} c^k_{ix.j} & c^k_{iy.j} & c^k_{iz.j} \end{bmatrix}^\top \boldsymbol{h}^k_\gamma \geq 0 \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ i = 2, \cdots, n_M,\\ \gamma = 1, \cdots, n_{\boldsymbol{\nu}},\\ j = 1, \cdots, d_T + 1, \end{cases}\\
& && \begin{bmatrix} \hat{c}^k_{1x.j} & \hat{c}^k_{1y.j} & \hat{c}^k_{1z.j} \end{bmatrix}^\top \boldsymbol{h}^k_\gamma \geq 0 \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ \gamma = 1, \cdots, n_{\boldsymbol{\nu}},\\ j = 1, \cdots, d_L - 1, \end{cases}\\
& && (\boldsymbol{c}^k_{i_1,i_2.j})^\top \boldsymbol{h}^k_{RSFS.i_1,i_2} \geq \delta_{RSFS} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ i_1 = 2, \cdots, n_M - 1,\\ i_2 = i_1 + 1, \cdots, n_M,\\ j = 1, \cdots, d_T + 1, \end{cases}\\
& && (\hat{\boldsymbol{c}}^k_{1,i_2.j})^\top \boldsymbol{h}^k_{RSFS.1,i_2} \geq \delta_{RSFS} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ i_2 = 2, \cdots, n_M,\\ j = 1, \cdots, d_L - 1, \end{cases}\\
& && \left\| \begin{bmatrix} c^k_{ix.j} & c^k_{iy.j} & c^k_{iz.j} \end{bmatrix}^\top \right\| \leq T_{max} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ i = 2, \cdots, n_M,\\ j = 1, \cdots, d_T + 1, \end{cases}\\
& && \left\| \begin{bmatrix} \hat{c}^k_{1x.j} & \hat{c}^k_{1y.j} & \hat{c}^k_{1z.j} \end{bmatrix}^\top \right\| \leq T_{max} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ j = 1, \cdots, d_T + 1, \end{cases}\\
& && c^k_{iz.j} \succeq T_{min} \quad \text{for} \quad \begin{cases} k = 1, \cdots, n_B,\\ i = 2, \cdots, n_M, \end{cases}\\
& && \hat{c}^k_{1z.j} \succeq T_{min} \quad \text{for} \quad k = 1, \cdots, n_B.
\end{aligned}
$$

4 Experimental Validation

This section presents the results of the proposed trajectory planning and control algorithms through indoor obstacle avoidance and transportation flight experiments. In Section 4.1 and Section 4.2.1, the experiment results for the single multi-rotor system and the multiple multi-rotor system are described, respectively. Each section consists of the experiment setting and results with discussion.

4.1 Single Multi-rotor with a Suspended Load

4.1.1 Experiment Setting

The software setup for the experiments is as follows. All the programs run under Robot Operating System (ROS) Kinetic version in Linux 16.04. The multi-rotor is equipped with a Pixhawk4 mini module for attitude estimation and low-level controller. The translational states and the yaw angle are estimated from the measurement of VICON motion capture system running at 100Hz. For an onboard computer which performs real-time optimization, Intel NUC7i7BNH with i7-7567U processor is used.

For validation of real-time performance, two settings are used, the first of which is continuous replanning. Since it is not possible to track the generated collision-free trajectory without error, the trajectory generator continuously replans every 0.1 seconds while MPC computes optimal control input every 0.01 seconds. The next setting is change of the desired position during flight. The trajectory generator generates new trajectories as soon as it receives a new goal point during flight.

Parameters for the trajectory generation are same for both the simulation and the experiment which are listed in Table 4.1. Note that the weight matrices are chosen by firstly adjusting its order reflecting their physical meanings and fine-tuning empirically to increase stability and performance of flight.

Table 4.1: Parameters for trajectory generation.

Parameter	Value	Parameter	Value
m_M	2.473 kg	m_L	0.152 kg
A_h	-7.904	B_h	7.979
N_T	80	l	1.2 m
r_M	0.35 m	h_M	0.3 m
r_C	0.1 m	r_L	0.05 m
$L,\ Q$	$diag\{300, 300, 90, 30, 30, 30, 60, 60, 60, 10, 10, 10, 1000, 1000\}$		
R	$diag\{10, 200, 200\}$		

4.1.2 Experiment Results and Discussion

Scenario 1: Horizontal Obstacles with a Vertical Gap

In the first scenario, two obstacles are horizontally placed with a vertical gap. Since the gap is shorter than the distance between the top of the multi-rotor and the bottom of the load, the vehicle cannot pass between the obstacles unless the cable is tilted.

The simulation result for the first scenario is depicted in Fig. 4.1, and it can be seen that the generated trajectory avoids collision and reaches the desired position which is marked as a pink sphere. Note that the red ellipsoids are obstacles, and the ellipsoids enclosing the vehicle is depicted only in the right figure to show the simulation result better. The multi-rotors in the figure are depicted with the same time interval. The translucent blue bars are the overall results for the cable while the opaque ones are the results with the same time interval. The suspended load is depicted with a black sphere.

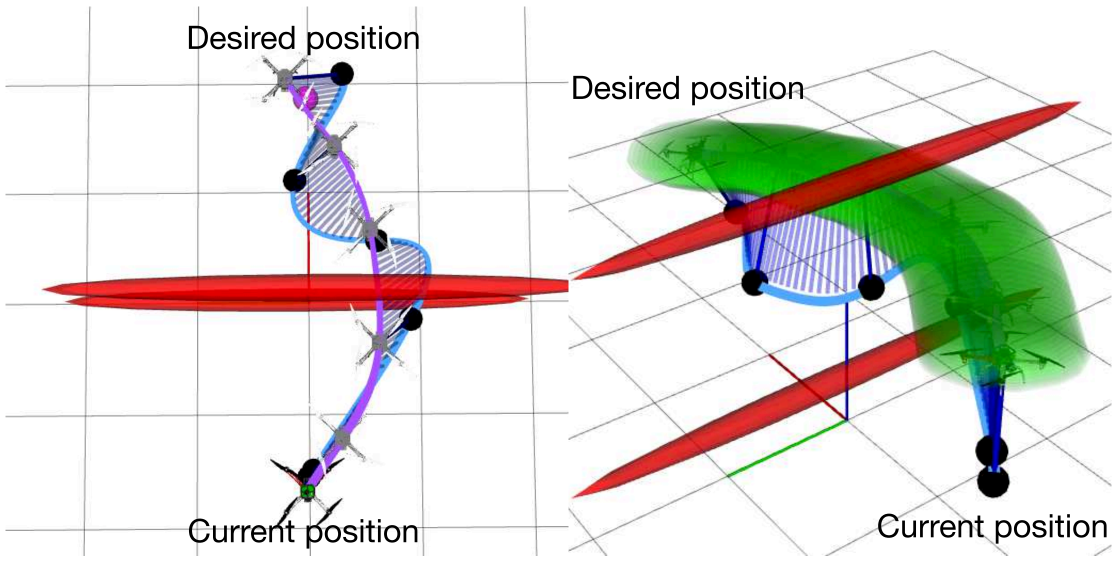


Figure 4.1: Simulation result for the first scenario. The generated trajectory passes the narrow gap by tilting the cable. The purple and the blue lines denote the path of the vehicle and the load, respectively.

An experiment is conducted to demonstrate the scenario. As shown in Fig. 4.2, the multi-rotor does not fly straight and moves quickly in the lateral direction to shake the suspended load and pass between the obstacles. The vehicle traversed six times without collision in the experiment. Fig. 4.3 shows the 2-D trajectory plot of the vehicle and the load. Fig. 4.4 shows the snapshot of the experiment, and it can be seen that the proposed algorithm enables the vehicle to successfully pass the narrow gap. The by showing the acceleration history of the load. The cable tautness assumption is validated by showing that the z-component of the acceleration is always greater than the gravitational acceleration as shown in Fig. 4.5.

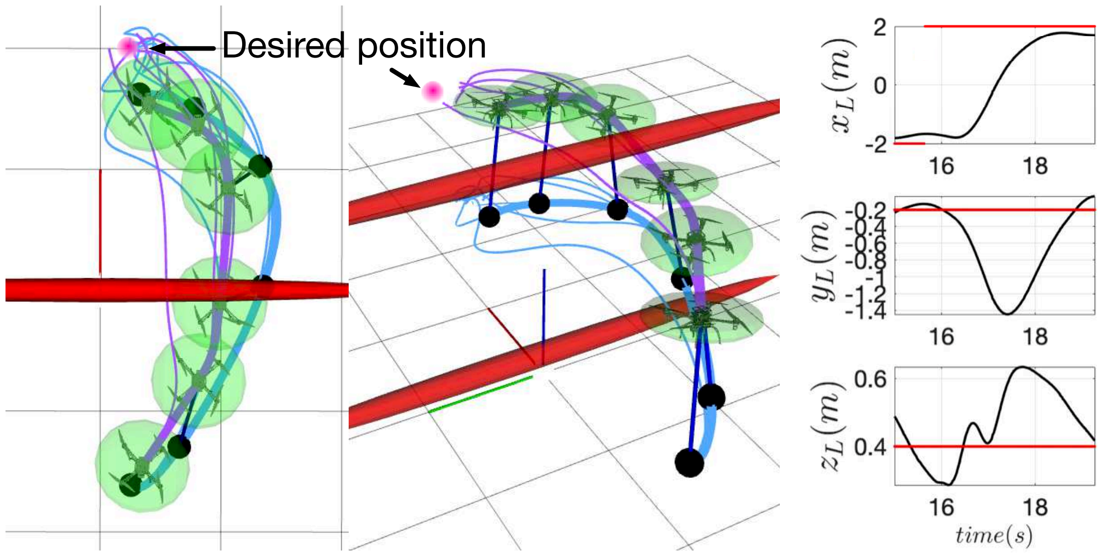


Figure 4.2: Experiment result of the first scenario. Thick lines denote actual positions of the multi-rotor and the load while thin lines denote generated trajectories at each time step.

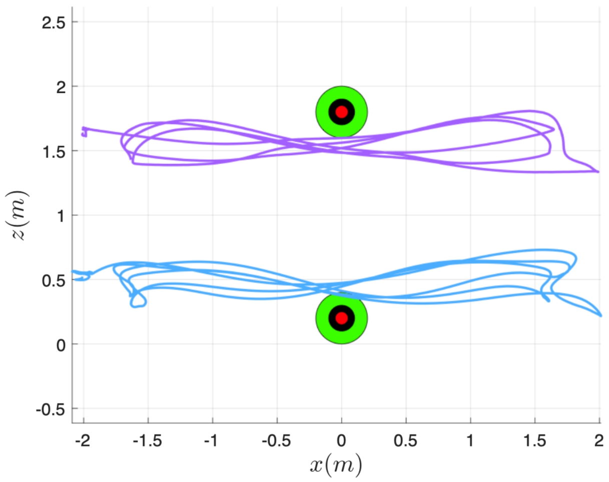


Figure 4.3: Side-view 2-D plot of the first scenario. Red circles denote the obstacles, and black and green circles are collision areas of the load and the multi-rotor.

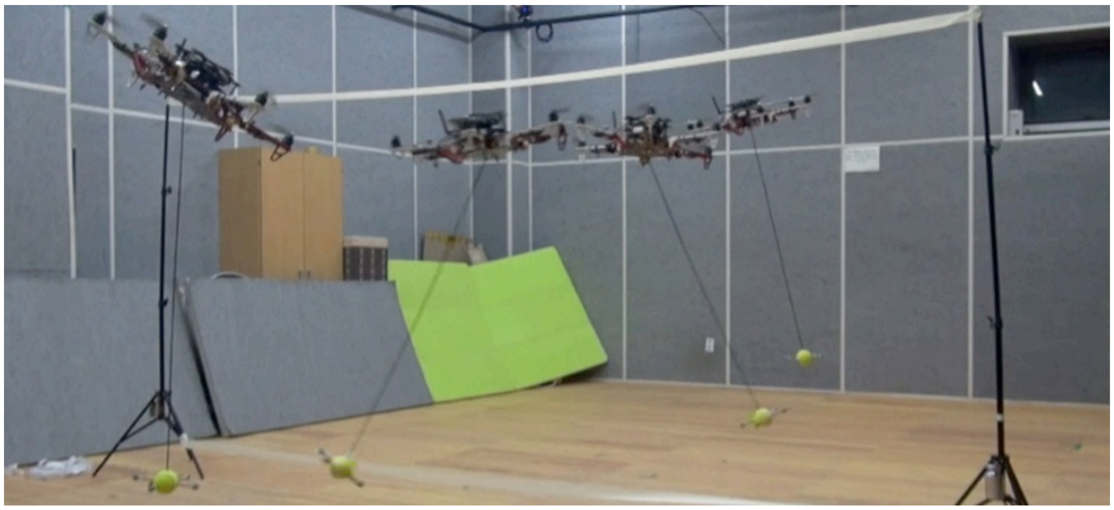


Figure 4.4: Experiment snapshot for the first scenario.

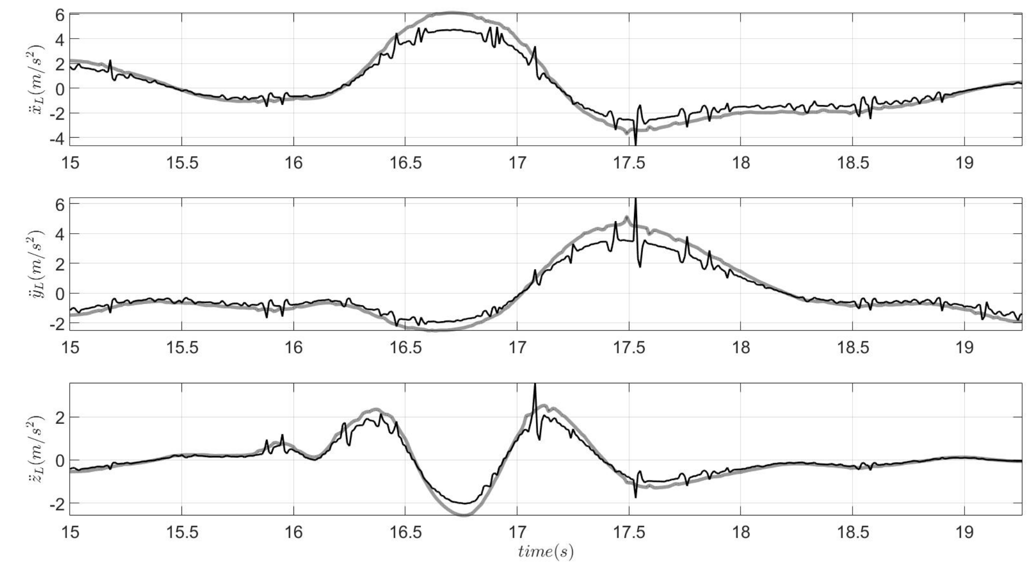


Figure 4.5: Acceleration history of the load for the first scenario. Translucent lines are the generated desired acceleration, and the solid lines are the experiment result.

Scenario 2: Vertical Obstacles Placed in Zigzags

In the second scenario, three obstacles are placed with short distances. Since the size of the multi-rotor hinders the vehicle and the load from flying straight, the trajectories should be generated avoiding the obstacles aggressively. Fig. 4.6 shows the simulation result for the second scenario. Again, the obstacles are drawn with red ellipsoids. The translucent blue bars are the overall results for the cable while the opaque ones are the results with the same time interval. The suspended load is depicted with a black sphere.

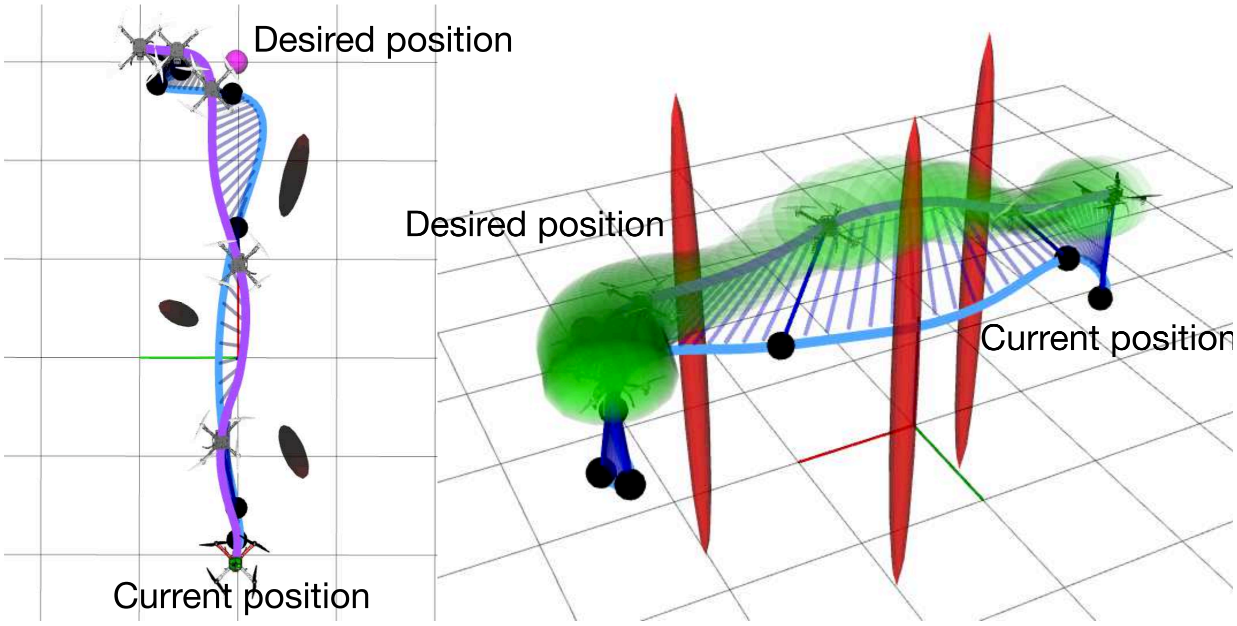


Figure 4.6: Simulation result for the second scenario. The generated trajectory shows a slalom-like maneuver. The purple and the blue line denote the path of the vehicle and the load, respectively.

In the experiment, as shown in Fig. 4.7, the multi-rotor successfully flies avoiding obstacles without losing agility. Four traversals are successfully completed in the experiment. Fig. 4.8 shows the 2-D trajectory plot of the vehicle and the load. Fig. 4.9 shows the snapshot of the experiment, and the proposed algorithms successfully operates the slung load system safely. The cable tautness assumption is validated by showing that the z-component of the acceleration is always greater than the gravitational acceleration as shown in Fig. 4.10.

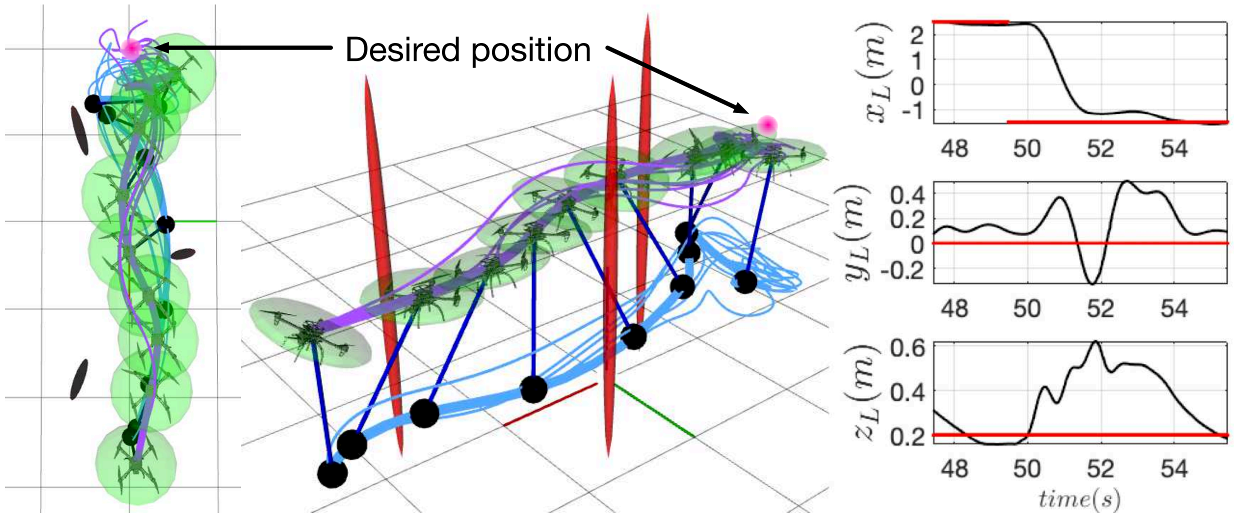


Figure 4.7: Experiment result of the second scenario. Thick lines denote actual positions of the multi-rotor and the load, while thin lines denote generated trajectories at each time step.

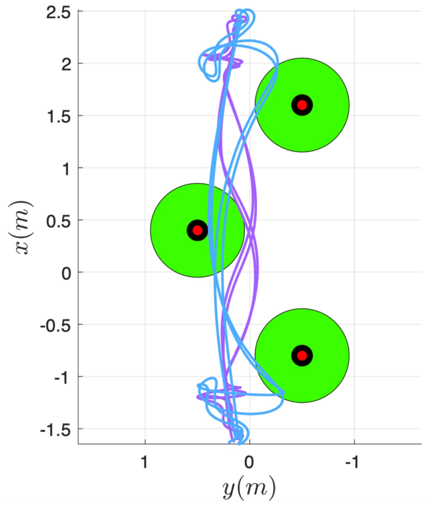


Figure 4.8: Side-view 2-D plot of the second scenario. Red circles denote the obstacles, and black and green circles are collision areas of the load and the multi-rotor.

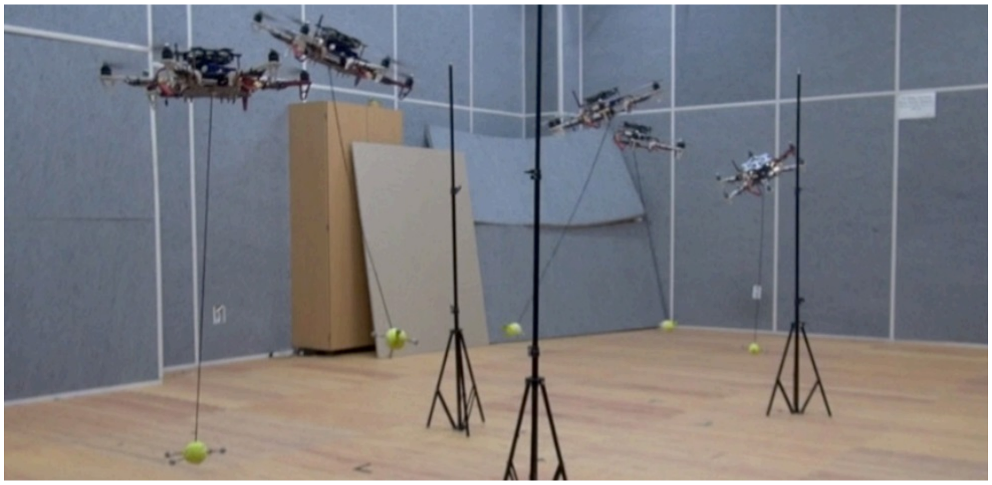


Figure 4.9: Experiment snapshot for the second scenario.

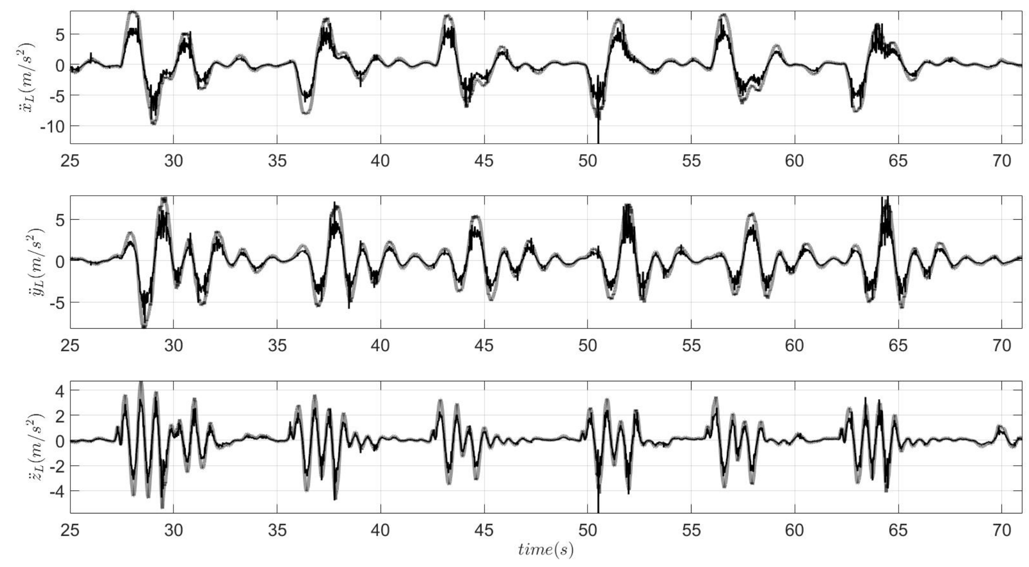


Figure 4.10: Acceleration history of the load for the second scenario. Translucent lines are the generated desired acceleration, and the solid lines are the experiment result.

Computation Time

In Table 4.2, the measured computation time for the planner and controller is listed. Note that even the maximum computation time meets the operating frequencies of the planner and the controller, which are 10 Hz and 100 Hz, respectively.

Table 4.2: Computation time data.

Algorithm	Average	Maximum	Minimum
Planner	60.9 ms	97.6 ms	10.4 ms
MPC	2.7 ms	10.5 ms	1.1 ms

4.2 Multiple Multi-rotors with a Suspended Load

4.2.1 Generated Trajectories

Scenario 1: Cluttered Environment

The environment in the first scenario has several obstacles with close distances. The first obstacle the vehicles meet hinders flying straightly. And the remaining two obstacles have short displacement, which makes the multi-rotors shrink toward the load.

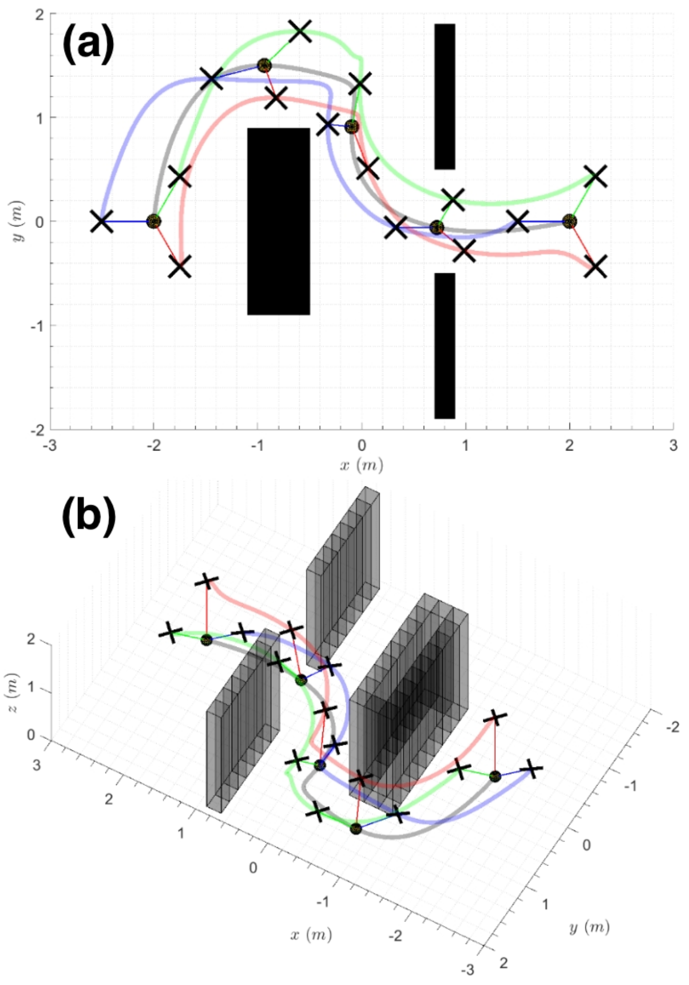


Figure 4.11: The generated trajectories for the first scenario. (a) Top view and (b) Perspective view.

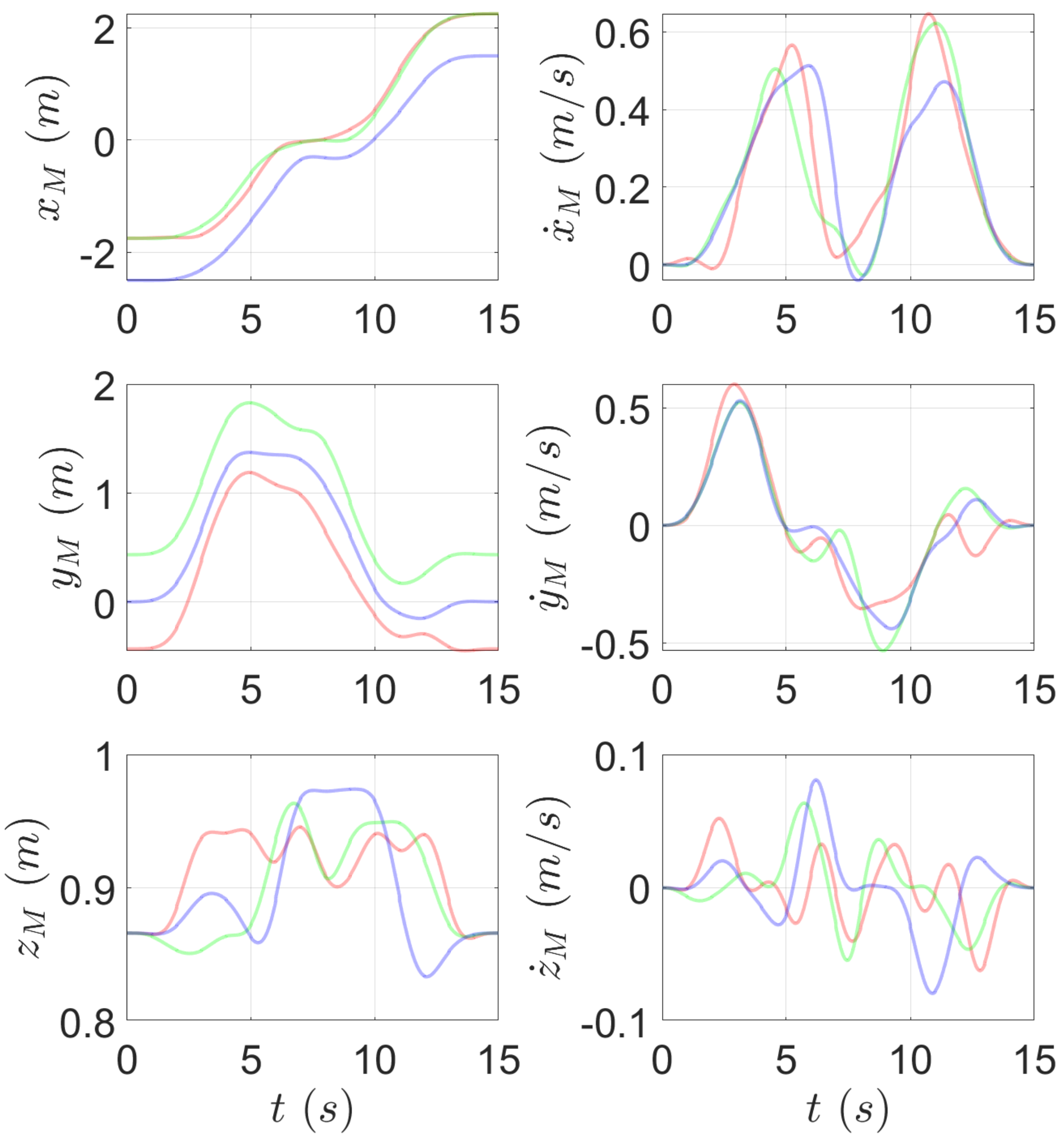


Figure 4.12: The generated trajectories of the multi-rotors for the first scenario.

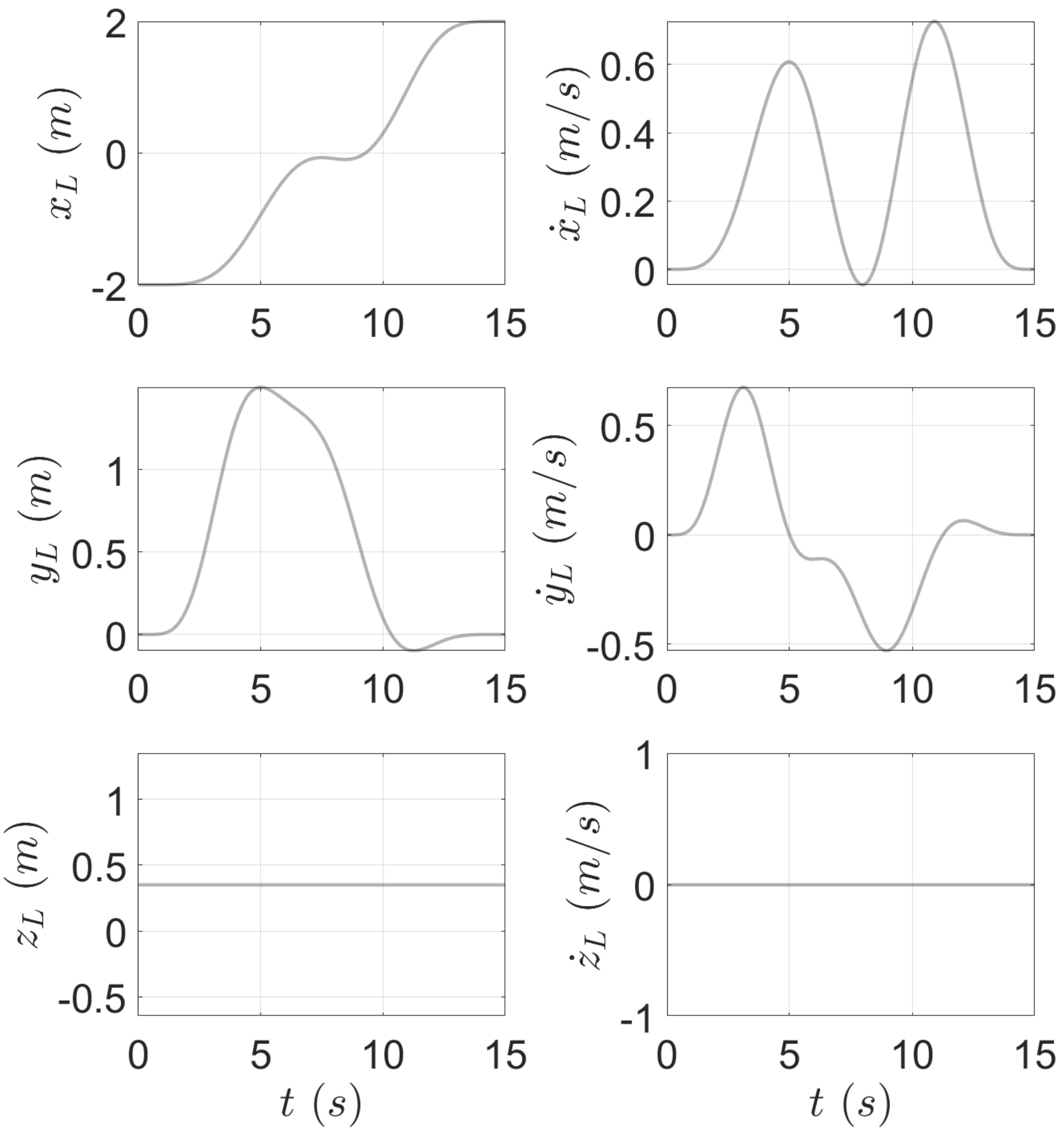


Figure 4.13: The generated trajectories of the load for the first scenario.

Scenario 2: Narrow Passage

The environment in the second scenario has two obstacles with the very short distance. To pass through the narrow gap, the multi-rotors fly in a almost straight line.

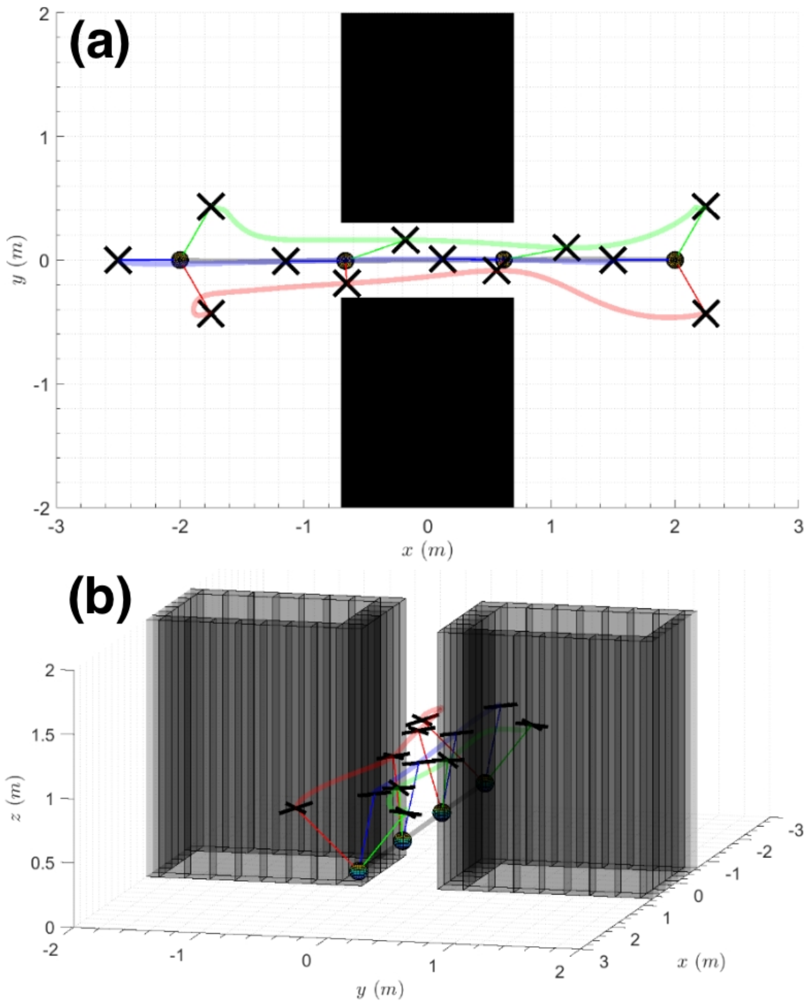


Figure 4.14: The generated trajectories for the second scenario. (a) Top view and (b) Perspective view.

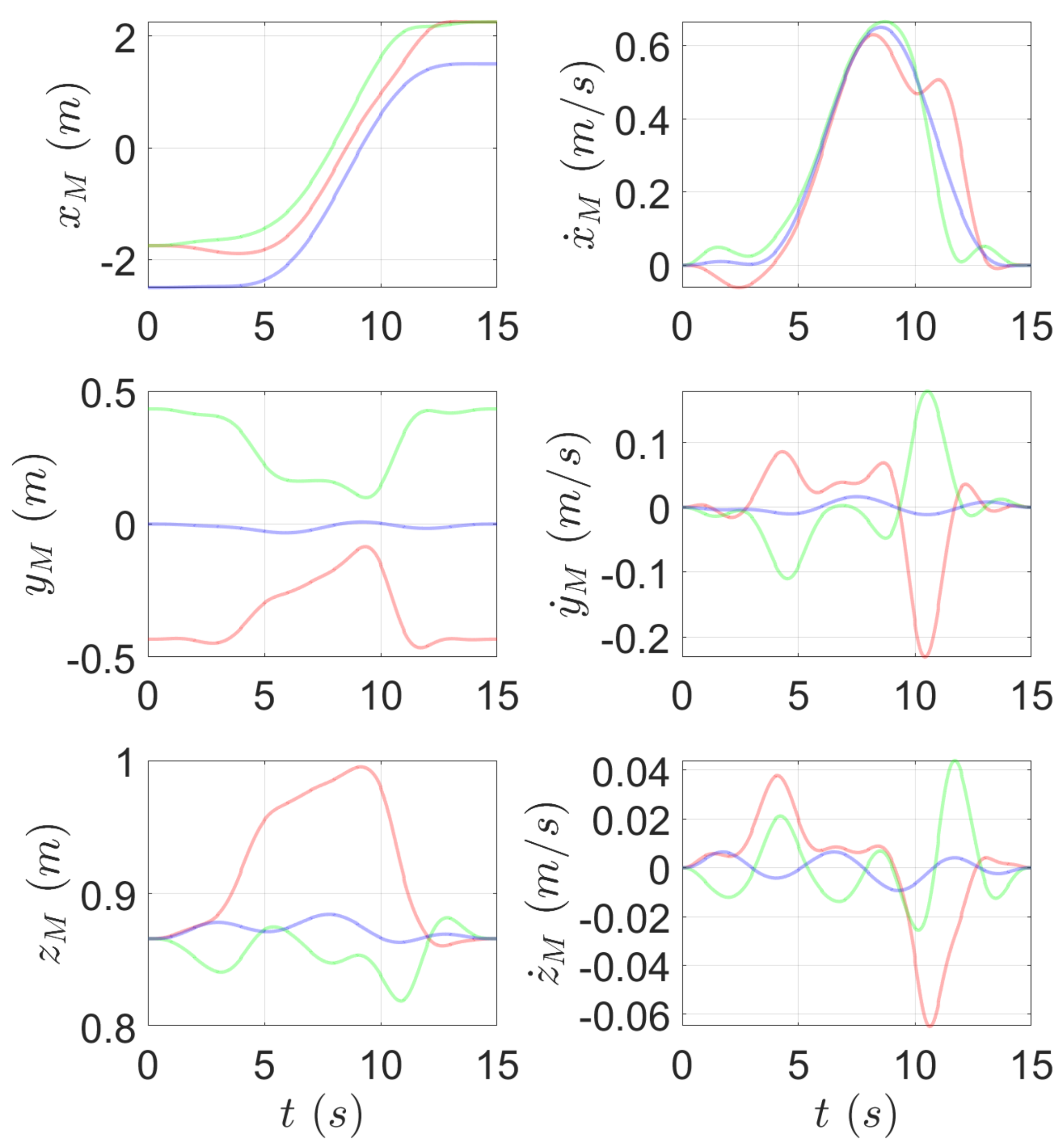


Figure 4.15: The generated trajectories of the multi-rotors for the second scenario.

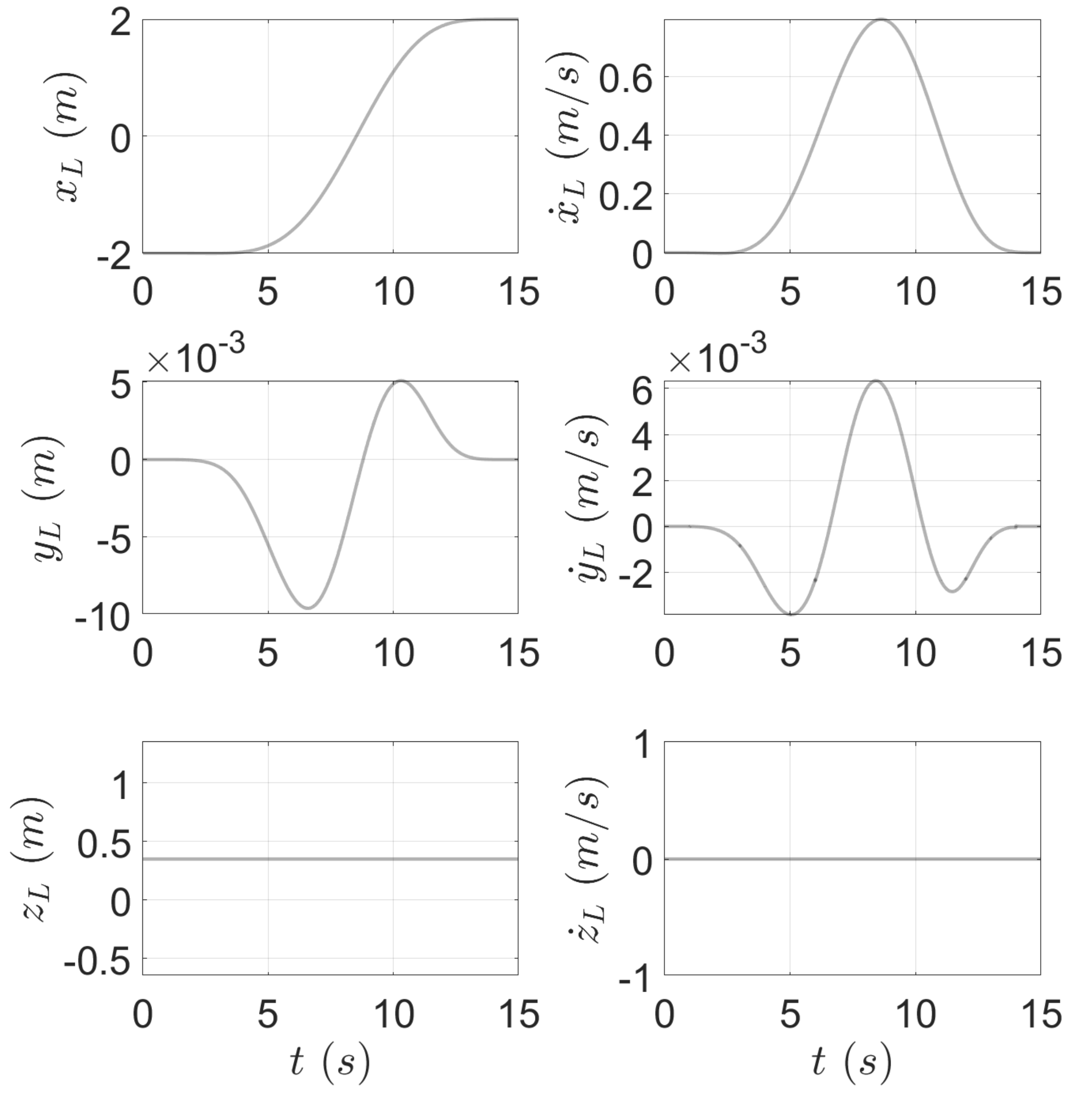


Figure 4.16: The generated trajectories of the load for the second scenario.

Computation Time

The trajectory generation problems Problem 2 and Problem 3 are sequentially optimized in advance on a desktop with *Intel* i7-4790 CPU and 16.0GB RAM. The optimization is performed using a convex optimization solver [44] in *MATLAB*. The parameters which mainly affect the computation time and the timing results are listed in Table 4.3 and Table 4.4. One hundred trajectory optimizations are performed for evaluation of the proposed algorithm, and the corresponding mean, minimum, maximum, and standard deviation of computation time are shown. In each optimization, the timing results are all different since trajectory initialization with sampling is newly performed. Although the overall computation currently takes around 7 seconds, it is expected to be at least ten times faster if embedded in C code and run with multiple cores.

In Table 4.4, three timing results for the first scenario and one result for the second scenario are shown. In the first scenario, the trajectory generation is performed by varying the obstacle number to check the effect. It can easily be checked the obstacle number scarcely effects the performance. Compared to the first scenario, the performance in load trajectory generation is very fast since it generates a simple straight trajectory in the second scenario.

Table 4.3: Parameters for the trajectory generation.

Parameter	n_M	n_B	n_{sample}	$n_{\boldsymbol{v}}$
Value	3	15	20000	8

Table 4.4: Timing results for the trajectory generation.

		Mean	Min	Max	Std Dev
Scenario 1 (one obstacle)	Load sampling (Algorithm 4)	0.20 s	0.10 s	0.23 s	0.03 s
	Load optimization (Problem 2)	0.52 s	0.29 s	0.51 s	0.13 s
	Tension sampling (Algorithm 5)	1.99 s	1.85 s	2.13 s	0.08 s
	SFS generation (Algorithm 6)	0.05 s	0.01 s	0.08 s	0.02 s
	Tension optimization (Problem 3)	4.01 s	2.11 s	5.23 s	1.33 s
Scenario 1 (two obstacles)	Load sampling (Algorithm 4)	0.21 s	0.13 s	0.26 s	0.02 s
	Load optimization (Problem 2)	0.56 s	0.30 s	0.63 s	0.14 s
	Tension sampling (Algorithm 5)	2.03 s	1.89 s	2.27 s	0.11 s
	SFS generation (Algorithm 6)	0.05 s	0.02 s	0.06 s	0.01 s
	Tension optimization (Problem 3)	4.05 s	2.19 s	6.2 s	1.20 s
Scenario 1 (three obstacles)	Load sampling (Algorithm 4)	0.22 s	0.14 s	0.24 s	0.03 s
	Load optimization (Problem 2)	0.54 s	0.32 s	0.69 s	0.11 s
	Tension sampling (Algorithm 5)	2.03 s	1.88 s	2.17 s	0.09 s
	SFS generation (Algorithm 6)	0.05 s	0.02 s	0.07 s	0.01 s
	Tension optimization (Problem 3)	4.06 s	2.29 s	6.34 s	1.23 s
Scenario 2	Load sampling (Algorithm 4)	0.21 s	0.13 s	0.27 s	0.04 s
	Load optimization (Problem 2)	0.10 s	0.05 s	0.14 s	0.03 s
	Tension sampling (Algorithm 5)	2.03 s	1.88 s	2.17 s	0.09 s
	SFS generation (Algorithm 6)	0.05 s	0.02 s	0.07 s	0.01 s
	Tension optimization (Problem 3)	3.26 s	2.01 s	5.11 s	0.56 s

4.2.2 Experiment Setting

To validate the proposed algorithm, trajectory tracking experiment is performed for the first scenario in Section 4.2.1. Three custom-made multi-rotors are used which are shown in Fig. 4.18. Each vehicle is built on a Armattan Rooster frame and uses four Armattan Oomph Titan motors. For attitude and thrust control, a Pixhawk 4 FCU (Flight Control Unit) is used which provides the current attitude information of the vehicle and sends PWM (Pulse Width Modulation) signals to the motors. Additionally, SparkFun WRL-00705 is used to receive the control commands and transmit the IMU (Inertial Measurement Unit) information to a laptop used as a GCS (Ground Control Station). The vehicle weighs 0.762 kg including a 4S 2200 mAh Lipo battery.

The laptop runs three position controllers to compute the desired rotational velocity and the thrust given the collision-free trajectory. Crazyradio PA is used with the laptop to communicate with the vehicles. The navigational information of the vehicles is estimated using an external motion capture system, OptiTrack, and a Kalman filter. To compensate the external force from the load, a robust controller proposed in [31] is used which estimates and compensates the disturbance using a disturbance observer. Fig. 4.17 shows the overall control structure for the experiment.

The weight of the suspended load is 0.976 kg, which is too heavy for a single multi-rotor to carry alone.

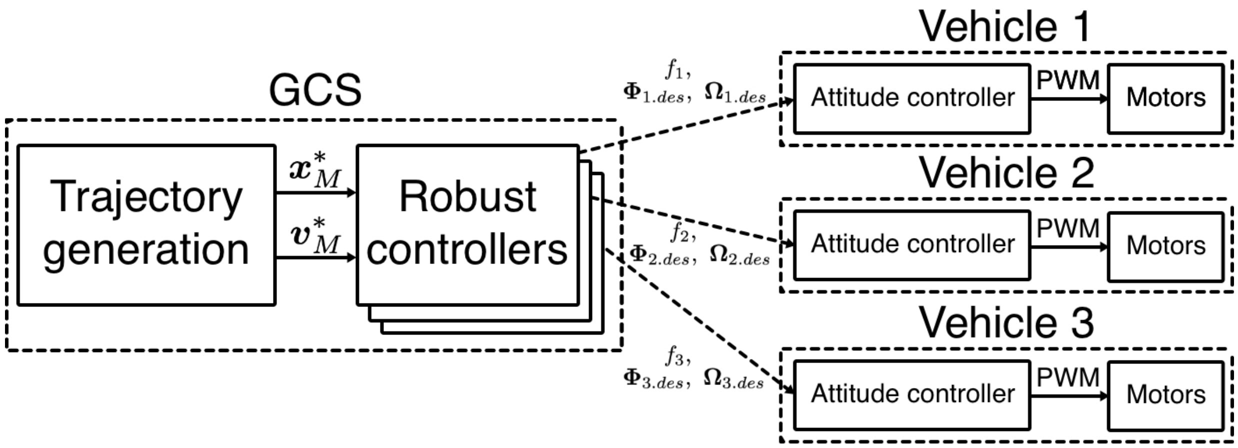

GCS
Trajectory generation
$\boldsymbol{x}_M^*$
$\boldsymbol{v}_M^*$
Robust controllers
f_1, $\boldsymbol{\Phi}_{1.des}$, $\boldsymbol{\Omega}_{1.des}$
f_2, $\boldsymbol{\Phi}_{2.des}$, $\boldsymbol{\Omega}_{2.des}$
f_3, $\boldsymbol{\Phi}_{3.des}$, $\boldsymbol{\Omega}_{3.des}$
Vehicle 1
Vehicle 2
Vehicle 3
Attitude controller
PWM
Motors

Figure 4.17: The control structure for the experiment.

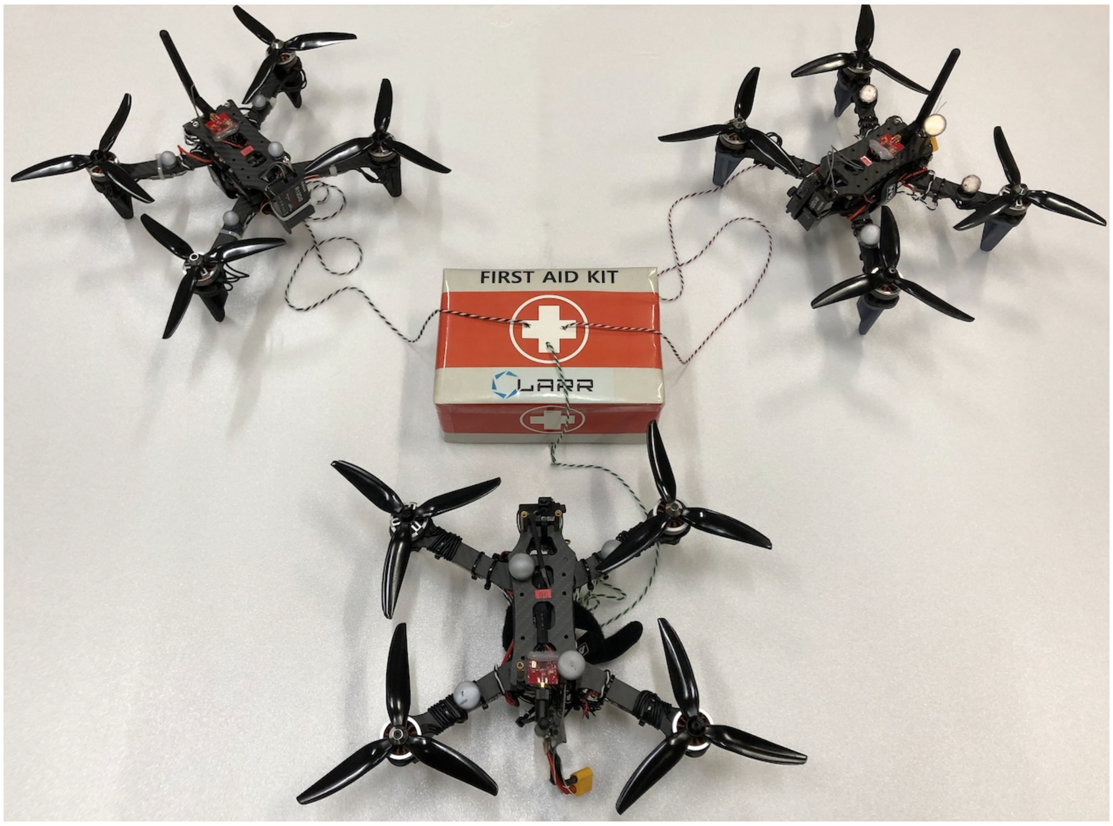


Figure 4.18: The multi-rotors and the suspended load used for the experiment.

4.2.3 Experiment Results and Discussion

To validate the proposed algorithm, a trajectory tracking experiment is conducted. In the following figures, the generated trajectories are denoted in translucent lines while the experiment data is shown in solid lines. The vehicle trajectories are drawn in RGB corresponding to each multi-rotor, and the suspended load is drawn in black.

The experiment snapshot is shown in Fig. 4.19 where three multi-rotors cooperatively transport a suspended load. The top and perspective views of the tracking result are shown in Fig. 4.20. It can be easily checked that the multi-rotors and the suspended load accurately follow the generated trajectory without any collision.

Fig. 4.21 and Fig. 4.22 show the tracking error of the tracking experiment for the multi-rotors and load, respectively. The RMSE for each translational state is listed in Table 4.5 and Table 4.6. The load states also show small tracking error although it is passively controlled. Since the maximum acceleration of the load, maximum tension, and minimum tension of the cables are considered in the trajectory optimization, the generated result is dynamically feasible, which results in good tracking results. Fig. 4.23 shows that the attitudes of the vehicles are not zeros since they should generate horizontal forces to compensate the cable tensions.

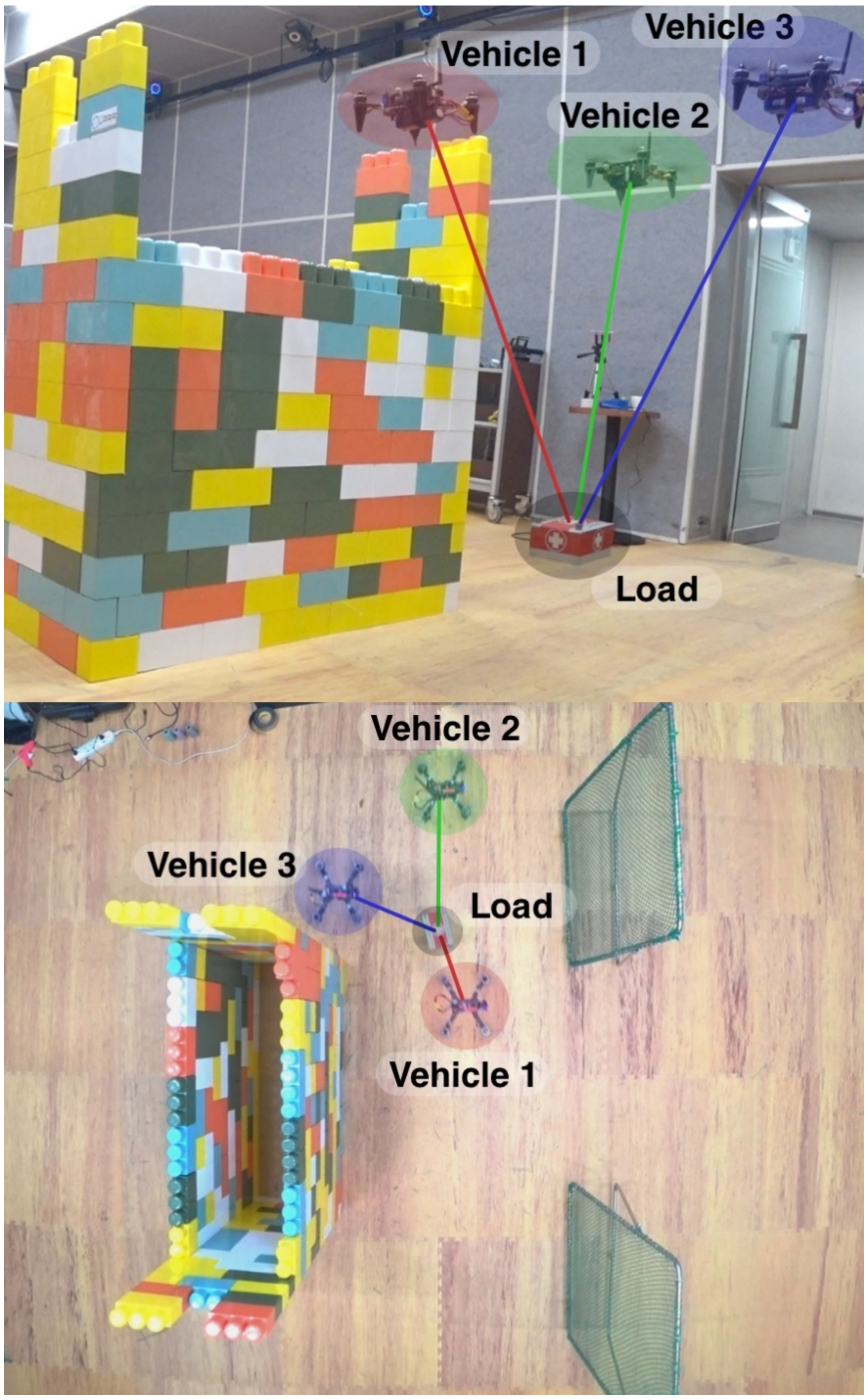


Figure 4.19: Multi-rotors transporting a suspended load cooperatively.

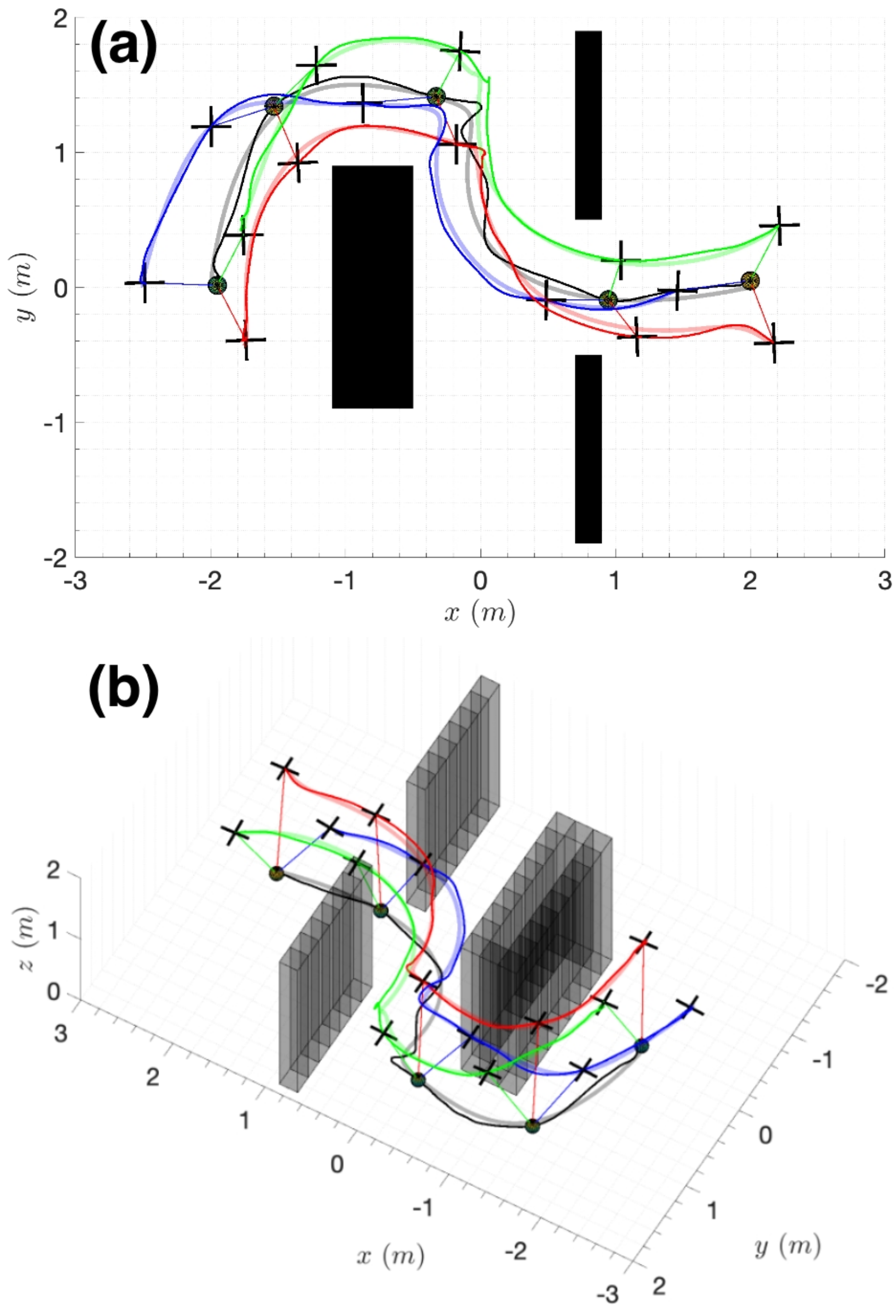


Figure 4.20: The tracking result of the experiment. Translucent lines are the generated desired trajectories, and the solid lines are the experiment result. (a) Top view of the experiment. (b) Perspective view of the experiment.

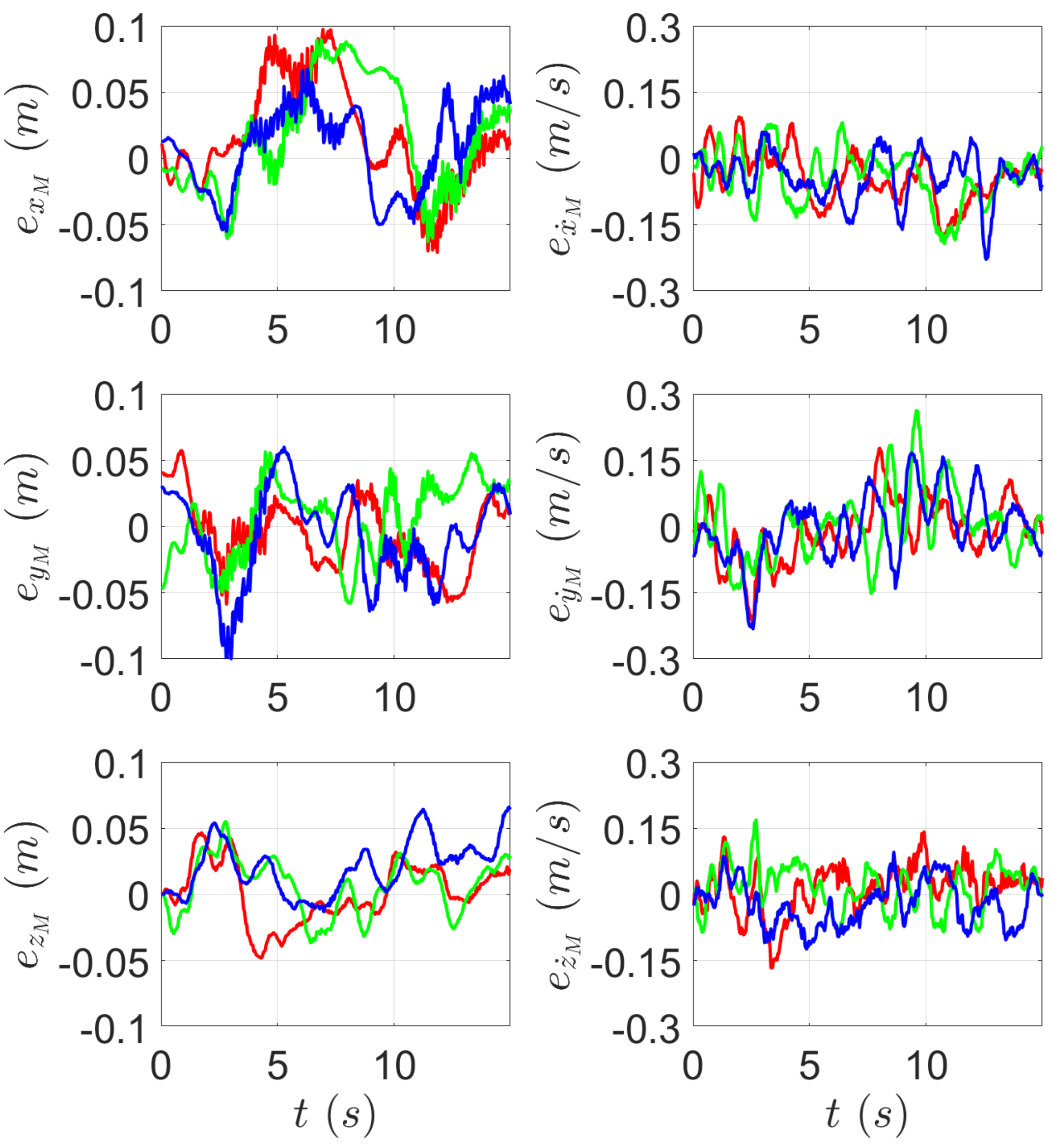


Figure 4.21: The tracking error result of the multi-rotors for the translational states.

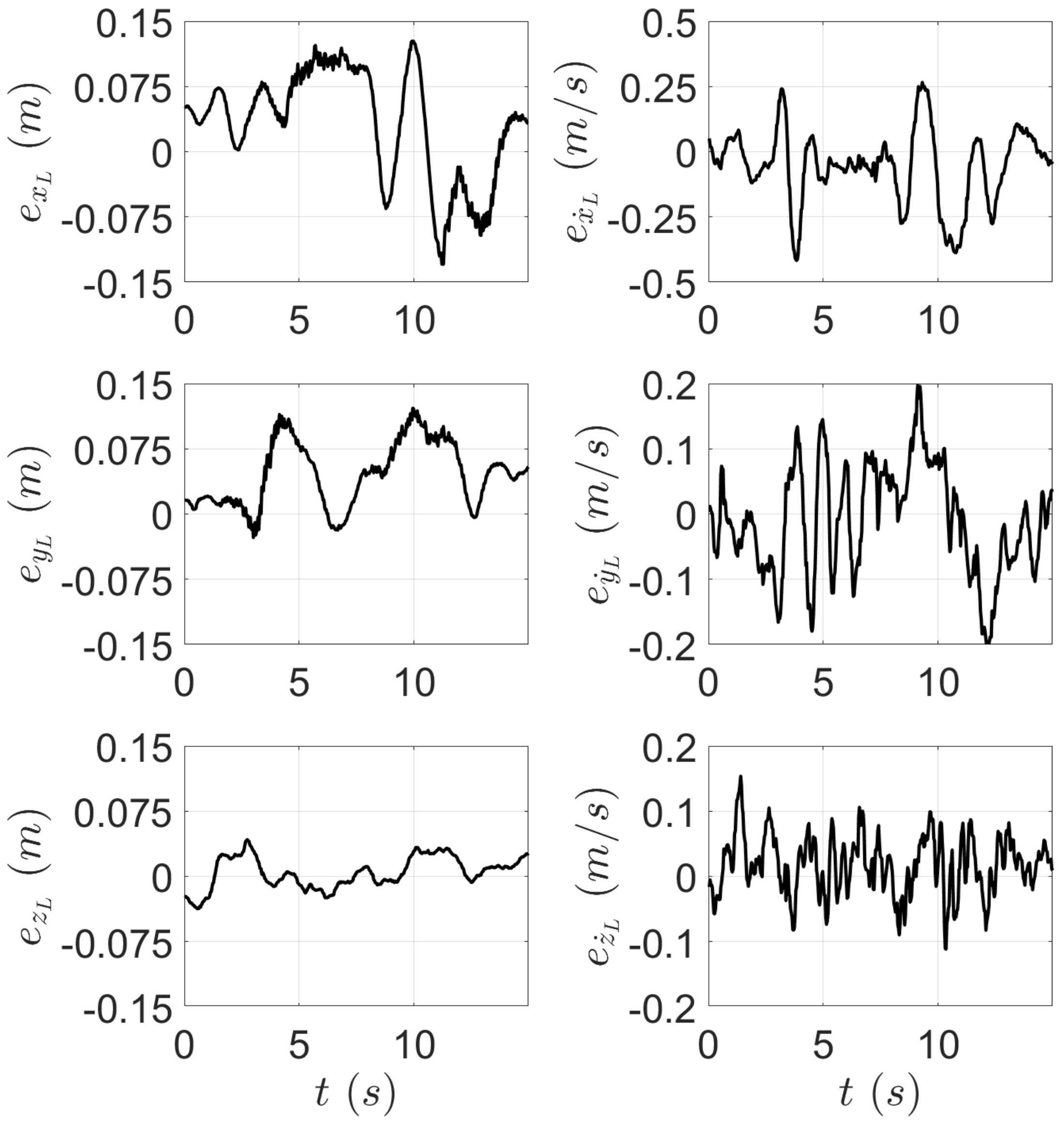


Figure 4.22: The tracking error result of the suspended load.

Table 4.5: RMSE of the trajectory tracking experiment for the multi-rotors.

	State	RMSE (m)	State	RMSE (m)	State	RMSE (m)
Position	x_1	0.042	x_2	0.044	x_3	0.033
	y_1	0.027	y_2	0.029	y_3	0.035
	z_1	0.022	z_2	0.022	z_3	0.029
	State	RMSE (m/s)	State	RMSE (m/s)	State	RMSE (m/s)
Velocity	$\dot{x}_1$	0.071	$\dot{x}_2$	0.074	$\dot{x}_3$	0.067
	$\dot{y}_1$	0.069	$\dot{y}_2$	0.079	$\dot{y}_3$	0.074
	$\dot{z}_1$	0.055	$\dot{z}_2$	0.053	$\dot{z}_3$	0.055

Table 4.6: RMSE of the trajectory tracking experiment for the load.

	State	RMSE (m)
Position	x_L	0.071
	y_L	0.059
	z_L	0.059
	State	RMSE (m/s)
Velocity	$\dot{x}_L$	0.152
	$\dot{y}_L$	0.084
	$\dot{z}_L$	0.048

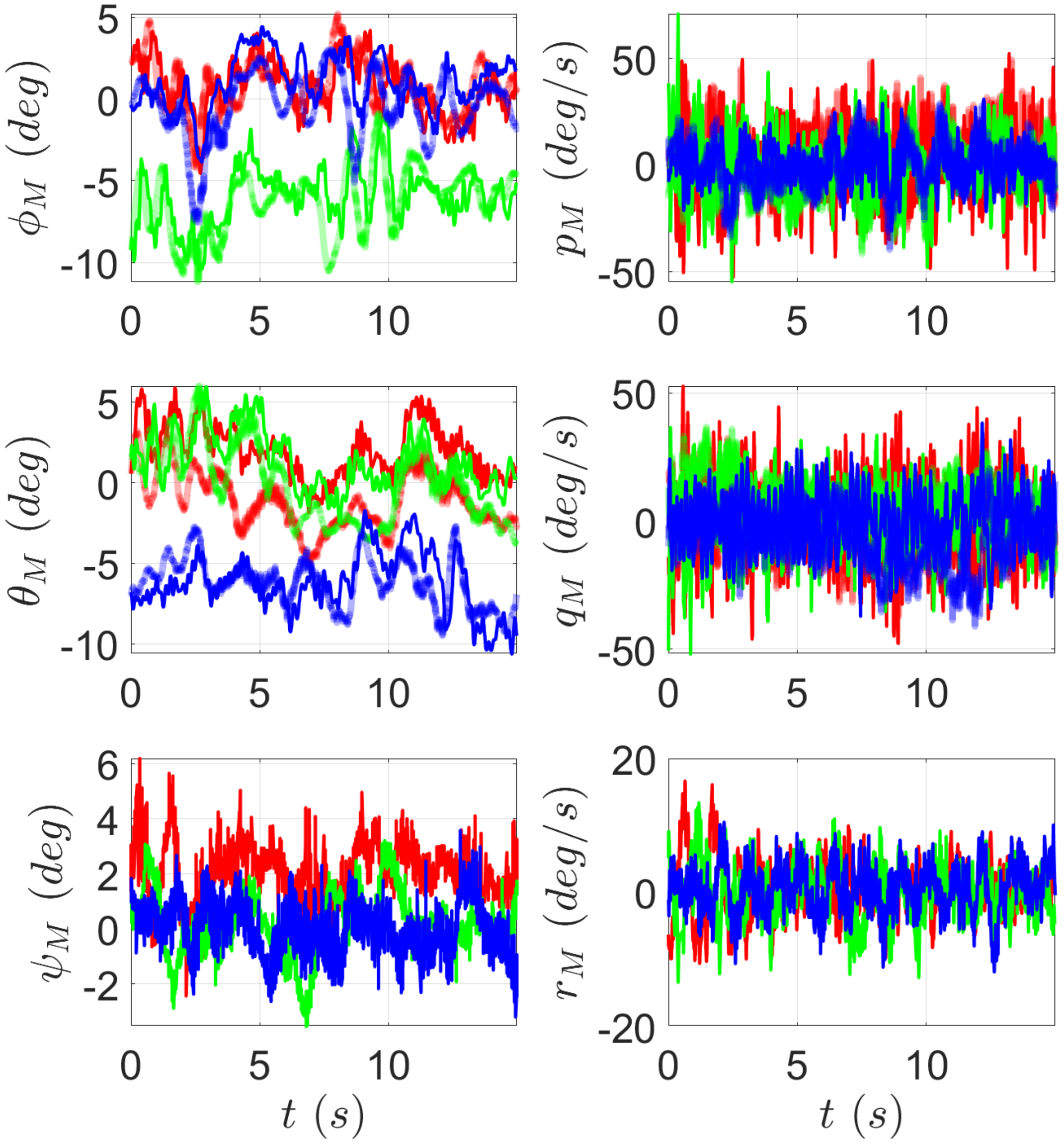


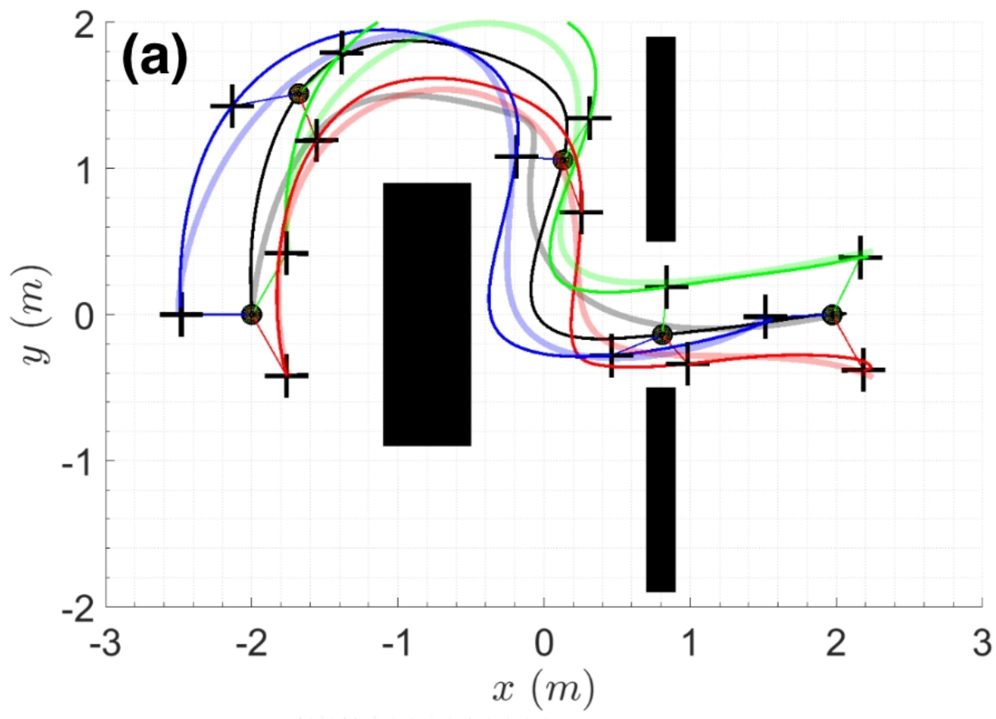
Figure 4.23: The tracking result of the multi-rotors for the rotational states. Translucent lines are the generated desired trajectories, and the solid lines are the experiment result.

4.2.4 Comparison with formation control

The proposed algorithm requires relatively high computational load compared to the formation control algorithms since several constraints are considered in addition to the dynamic model. As a comparison work, [45] is used which generates trajectories of the vehicles without the consideration of the dynamics. The work generates relative safe flight corridors for convexification of the non-convex collision-avoidance constraints.

The optimal load trajectory generated in Problem 2 is used identically, and [45] generates collision-free trajectories of each vehicle for the $x-$ and $y-$axis. While computing each trajectory, an additional constraint is imposed to restrict the distance of the vehicles from the load less than the radius of the cable length. After optimization, the altitudes of the vehicles are decided using the cable length constraints.

A simulation is performed to track the generated trajectories. As explained Section 4.2.1, each vehicle runs a trajectory tracking controller individually. The tracking result is shown in Fig. 4.24. Fig. 4.25 and Fig. 4.26 show the tracking error for the simulation. As shown in the figures, the tracking error becomes very large compared to the proposed algorithm. Since the tension of the cable and the overall dynamics for the load and the vehicles are not considered in the referred work, the generated trajectories are infeasible to be tracked accurately. This inaccurate tracking performance may result in collision although the generated trajectories are collision-free.



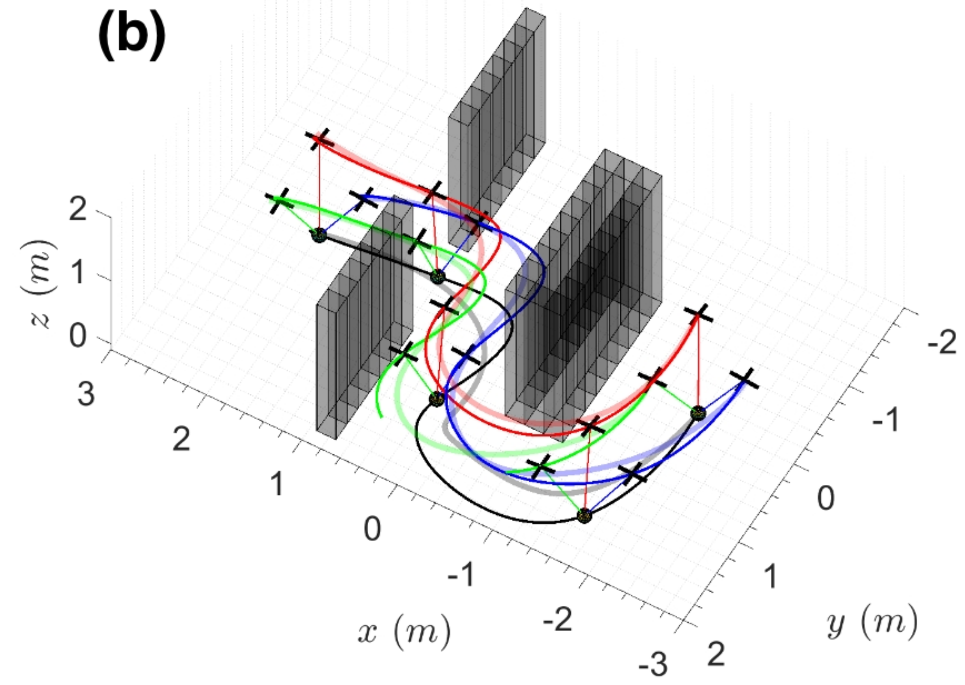


Figure 4.24: The tracking result for the formation control. Translucent lines are the generated desired trajectories, and the solid lines are the experiment result. (a) Top view of the experiment. (b) Perspective view of the experiment.

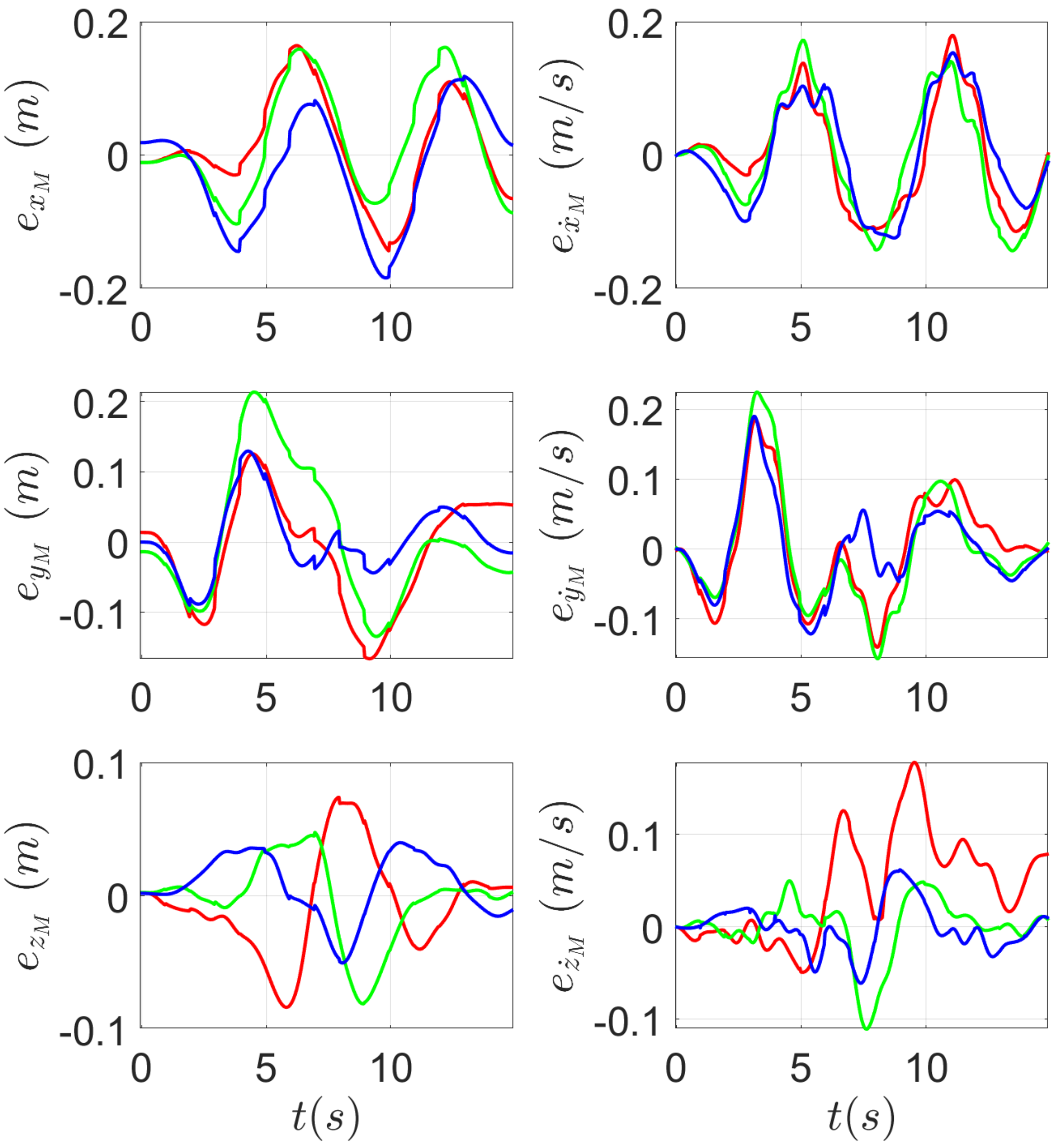


Figure 4.25: The tracking error result of the multi-rotors for the translational states.

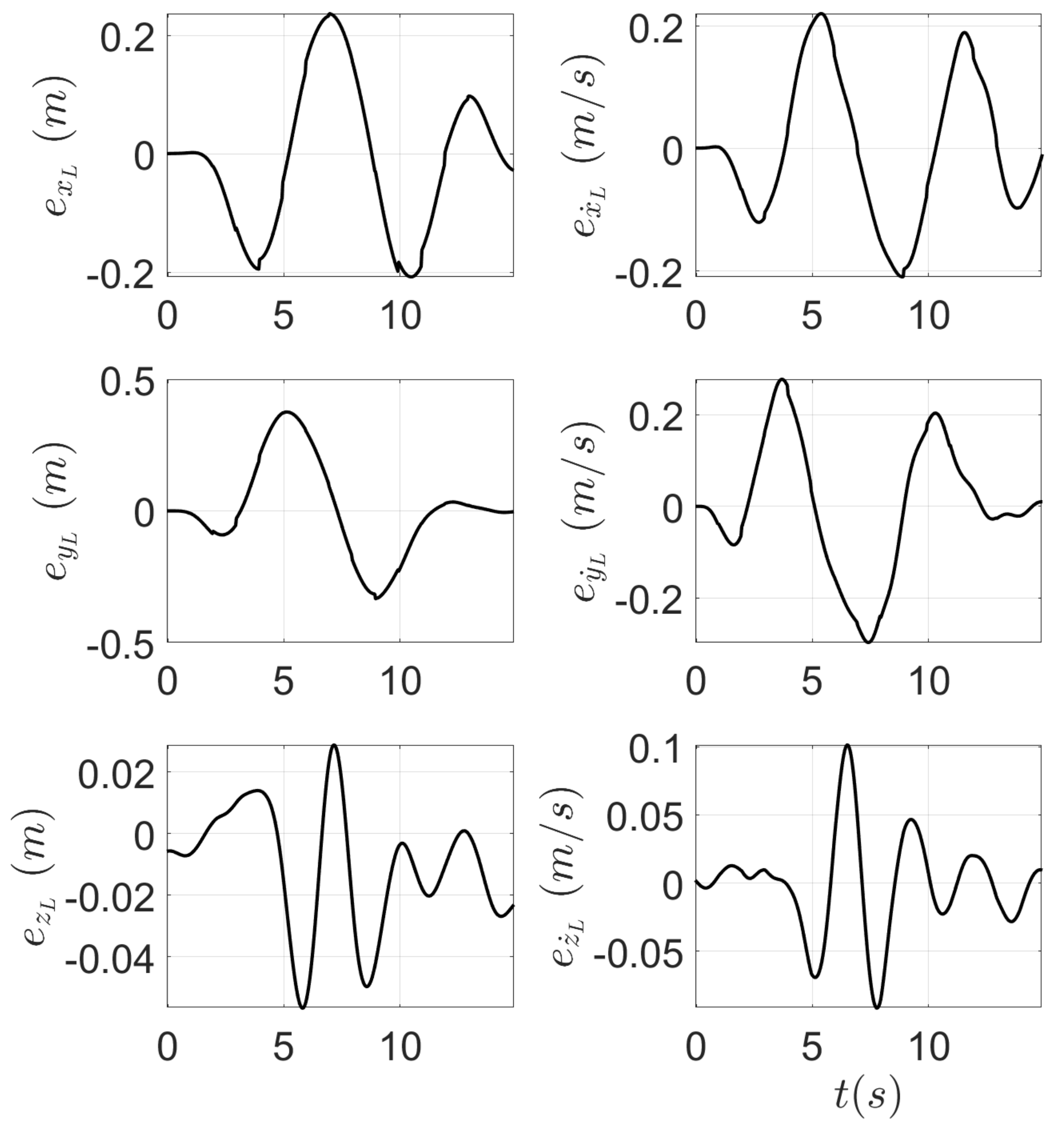


Figure 4.26: The tracking error result of the suspended load.

5 Conclusion

This dissertation presents trajectory generation and control methods for multi-rotor with a suspended load. Since the dynamic model becomes more complex as cables are attached compared with the bare multi-rotor, efficient algorithms are required for operation of the system.

For a single multi-rotor with a suspended load, the dynamic model is simplified with the consideration of control delay, and it is used in the proposed real-time planning and optimal control algorithms. The entire system is considered in a less-conservative manner using ellipsoids for collision avoidance. Performance of the planner is validated with two difficult scenarios where safety cannot be assured without collision avoidance. To track the generated trajectories, model predictive control is used, and flight experiments successfully demonstrate agile maneuvers without losing safety and real-time performance.

The operation becomes more difficult when multiple multi-rotors transport a suspended load cooperatively as the dimension of the system increase proportionally to the number of the vehicles. Thanks to the differential flatness, the optimization variables are changed to ones with linear dynamic model, and the number of them is reduced. Furthermore, to compute the continuous trajectory using discrete optimization variables efficiently, the trajectories are parameterized using Bernstein polynomials whose convex hull property allows to consider various collision avoidance and dynamic constraints. Since solving the entire optimization is intractable due to high non-linearity and the large number of the optimization variables, the entire problem is decomposed into two sub-problems, which are load optimization and tension optimization, respectively. In the former sub-problem, collision avoidance of the load is guaranteed using SFC, and the clearance for the multi-rotors is also considered. In the latter sub-problem, using the proposed SFS and RSFS construction method, collision avoidance and inter-agent collision avoidance are guaranteed. The feasibility of the generated trajectory is validated through the successful collision avoidance experiment with little tracking error.

The possible future works of this dissertation will be increasing the generality. Since only a point mass load is considered, there exist room for development of the trajectory generation and control required to transport a rigid-body load. Moreover, when vehicles have cables with different lengths, the construction RSFS can be revised to generate a more tight safe set as the required minimum angle between two vehicles decreases. While the real-time performance is validated for a single slung load system, the proposed method for multiple multi-rotors is required to be validated to replan trajectories fast. Finally, since the size and position of the obstacles are assumed to be known, vision-based obstacle detection algorithms can increase practicality in operation of the system.

A
Detailed Derivation of Differential Flatness

This section shows a detailed derivation of the differential flatness of the system composed of n_M multi-rotors and a suspended point-mass load. While the differential flatness was proved in [24], explicit equations were omitted. The following steps are based on the derivations for a multi-rotor without a cable [35].

The tension vector for the first multi-rotor is not included in the flat outputs, and it can be computed using the fourth line of (3.1),

$$T_1\boldsymbol{q}_1 = m_L\ddot{\boldsymbol{x}}_L - \sum_{i=2}^{n_M} T_i\boldsymbol{q}_i + m_L g\boldsymbol{e}_3. \tag{A.1}$$

The direction vectors for the cables can be computed by normalizing the tension vectors as follows:

$$\boldsymbol{q}_i = T_i\boldsymbol{q}_i / \left\|T_i\boldsymbol{q}_i\right\|. \tag{A.2}$$

The position, velocity, and acceleration of the i^{th} multi-rotor can be expressed using the kinematic relationship as follows:

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{x}_L + l\boldsymbol{q}_i, \\ \dot{\boldsymbol{x}}_i &= \dot{\boldsymbol{x}}_L + l\dot{\boldsymbol{q}}_i, \\ \ddot{\boldsymbol{x}}_i &= \ddot{\boldsymbol{x}}_L + l\ddot{\boldsymbol{q}}_i. \end{aligned} \tag{A.3}$$

Note that the derivatives of $\boldsymbol{q}_i$ can be computed by differentiating $T_i\boldsymbol{q}_i$ with the same order.

The rotation matrix for the i^{th} multi-rotor, $R_i = [\boldsymbol{x}_{\mathcal{B}_i}\ \boldsymbol{y}_{\mathcal{B}_i}\ \boldsymbol{z}_{\mathcal{B}_i}]$, can be computed using the following equations:

$$\begin{aligned} \boldsymbol{z}_{\mathcal{B}_i} &= R_i\boldsymbol{e}_3 = \frac{f_iR_i\boldsymbol{e}_3}{\|f_iR_i\boldsymbol{e}_3\|}, \\ \boldsymbol{y}_{\mathcal{B}_i} &= \frac{\boldsymbol{z}_{\mathcal{B}_i} \times \boldsymbol{x}_{\mathcal{C}_i}}{\|\boldsymbol{z}_{\mathcal{B}_i} \times \boldsymbol{x}_{\mathcal{C}_i}\|}, \\ \boldsymbol{x}_{\mathcal{B}_i} &= \boldsymbol{y}_{\mathcal{B}_i} \times \boldsymbol{z}_{\mathcal{B}_i}, \end{aligned} \tag{A.4}$$

where $\boldsymbol{x}_{\mathcal{C}_i} = [cos\psi_i\ sin\psi_i\ 0]^\top$ is the x axis of the frame $\mathcal{C}_i$. The frame $\mathcal{C}_i$ is an intermediate frame computed after rotating the frame $\mathcal{I}$ with ψ_i angle with respect to the $\boldsymbol{z}_{\mathcal{I}}$ axis. Note that the y axis of the frame $\mathcal{C}_i$ is $\boldsymbol{y}_{\mathcal{C}_i} = [sin\psi_i\ \ -cos\psi_i\ 0]^\top$

By projecting both sides of the second line of (3.1) to $R_i\boldsymbol{e}_3 = \boldsymbol{z}_{\mathcal{B}_i}$, the thrust input for the i^{th} multi-rotor can be computed as follows:

$$\begin{aligned} m_i\ddot{\boldsymbol{x}}_i \cdot (R_i\boldsymbol{e}_3) &= f_i - m_ig\boldsymbol{e}_3 \cdot (R_i\boldsymbol{e}_3) - T_i\boldsymbol{q} \cdot (R_i\boldsymbol{e}_3), \\ f_i &= \big(m_i\ddot{\boldsymbol{x}}_i + m_ig\boldsymbol{e}_3 + T_i\boldsymbol{q}\big) \cdot \boldsymbol{z}_{\mathcal{B}_i}. \end{aligned} \tag{A.5}$$

To compute the rotational velocities of the i^{th} multi-rotor, $\boldsymbol{\Omega}_{\mathcal{B}_i} = [p_i\ q_i\ r_i]$, the second line of (3.1) is differentiated.

$$\begin{aligned} m_i\dddot{\boldsymbol{x}}_i &= \dot{f}_iR_i\boldsymbol{e}_3 + f_i\dot{R}_i\boldsymbol{e}_3 - (\dot{T_i\boldsymbol{q}_i}) \\ &= \dot{f}_iR_i\boldsymbol{e}_3 + f_iR_i\hat{\boldsymbol{\Omega}}_{\mathcal{B}_i}\boldsymbol{e}_3 - (\dot{T_i\boldsymbol{q}_i}). \end{aligned} \tag{A.6}$$

After projecting both sides of (A.6) to $\boldsymbol{x}_{\mathcal{B}_i}$, the following equation can be computed:

$$m_i \boldsymbol{x}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i = \dot{f}_i [1\ 0\ 0] \boldsymbol{e}_3 + f_i [1\ 0\ 0] \begin{bmatrix} q_i \\ -p_i \\ 0 \end{bmatrix} - \boldsymbol{x}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}). \tag{A.7}$$

Then, q_i can be computed as follows:

$$q_i = \frac{1}{f_i} \left(m_i \boldsymbol{x}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i + \boldsymbol{x}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}) \right). \tag{A.8}$$

For computation of p_i, both sides of (A.6) are projected to $\boldsymbol{y}_{\mathcal{B}_i}$:

$$m_i \boldsymbol{y}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i = \dot{f}_i [0\ 1\ 0] \boldsymbol{e}_3 + f_i [0\ 1\ 0] \begin{bmatrix} q_i \\ -p_i \\ 0 \end{bmatrix} - \boldsymbol{y}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}). \tag{A.9}$$

Then, p_i can be computed as follows:

$$p_i = -\frac{1}{f_i} \left(m_i \boldsymbol{y}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i + \boldsymbol{y}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}) \right). \tag{A.10}$$

The first required equation for the computation of r_i can be computed by expanding the fifth line of (3.1) after projecting both sides to $\boldsymbol{y}_{\mathcal{B}_i}$ as follows:

$$\begin{gathered} \boldsymbol{y}_{\mathcal{B}_i}^\top [\dot{\boldsymbol{x}}_{\mathcal{B}_i}\ \dot{\boldsymbol{y}}_{\mathcal{B}_i}\ \dot{\boldsymbol{z}}_{\mathcal{B}_i}] = [r_i\ 0\ \ -p_i], \\ r_i = \boldsymbol{y}_{\mathcal{B}_i}^\top \dot{\boldsymbol{x}}_{\mathcal{B}_i}. \end{gathered} \tag{A.11}$$

The second required equation is the derivative of $\boldsymbol{x}_{\mathcal{B}_i}$ which can be computed as follows:

$$\begin{aligned} \boldsymbol{x}_{\mathcal{B}_i} &= \frac{\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}}{\|\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\|} = \frac{\tilde{\boldsymbol{x}}_{\mathcal{B}_i}}{\|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\|}, \\ \dot{\boldsymbol{x}}_{\mathcal{B}_i} &= \frac{\dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i}}{\|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\|} - \frac{\tilde{\boldsymbol{x}}_{\mathcal{B}_i} (\tilde{\boldsymbol{x}}_{\mathcal{B}_i}^\top \dot{\boldsymbol{x}}_{\mathcal{B}_i})}{\|\dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i}\|^3}. \end{aligned} \tag{A.12}$$

By substituting (A.12) to (A.11) and using the condition that $\boldsymbol{y}_{\mathcal{B}_i}^\top \tilde{\boldsymbol{x}}_{\mathcal{B}_i} = 0$, the last angular velocity component can be computed as follows:

$$\begin{aligned}
\dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i} &= \dot{\boldsymbol{y}}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i} + \boldsymbol{y}_{\mathcal{C}_i} \times \dot{\boldsymbol{z}}_{\mathcal{B}_i} \\
&= (-\dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}) \times \boldsymbol{z}_{\mathcal{B}_i} + \boldsymbol{y}_{\mathcal{C}_i} \times (q_i \boldsymbol{x}_{\mathcal{B}_i} - p_i \boldsymbol{y}_{\mathcal{B}_i}), \\
r_i &= \frac{\boldsymbol{y}_{\mathcal{B}_i}^\top \dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i}}{\|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\|} = \frac{\dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + q_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i}}{\|\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\|}.
\end{aligned} \tag{A.13}$$

Lastly, the angular acceleration vector needs to be computed. The first thing to be found is the derivative of the thrust input by differentiating the second line of (3.1).

$$\begin{aligned}
m_i \dddot{\boldsymbol{x}}_i &= \dot{f}_i R_i \boldsymbol{e}_3 + f_i R_i \hat{\boldsymbol{\Omega}}_{\mathcal{B}_i} \boldsymbol{e}_3 - (\dot{T_i \boldsymbol{q}_i}) \\
&= \dot{f}_i R_i \boldsymbol{e}_3 + f_i (q_i \boldsymbol{x}_{\mathcal{B}_i} - p_i \boldsymbol{y}_{\mathcal{B}_i}) - (\dot{T_i \boldsymbol{q}_i}).
\end{aligned} \tag{A.14}$$

By projecting (A.14) to $\boldsymbol{z}_{\mathcal{B}_i}$, the derivative of the thrust input can be expressed as follows:

$$\dot{f}_i = \big(m_i \dddot{\boldsymbol{x}}_i + (\dot{T_i \boldsymbol{q}_i})\big) \cdot \boldsymbol{z}_{\mathcal{B}_i}. \tag{A.15}$$

For computation of $\dot{q}_i$, $f_i q_i$ is differentiated based on (A.8):

$$\dot{f}_i q_i + f_i \dot{q}_i = m_i (\dot{\boldsymbol{x}}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i + \boldsymbol{x}_{\mathcal{B}_i}^\top \dddot{\boldsymbol{x}}_i) + \dot{\boldsymbol{x}}_{\mathcal{B}_i}^\top (T_i \boldsymbol{q}_i) + \boldsymbol{x}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}). \tag{A.16}$$

Note that the one unrecognized term $\dot{\boldsymbol{x}}_{\mathcal{B}_i}$ can be expressed using the attitude and angular the velocity of the i^{th} vehicle as follows:

$$\dot{\boldsymbol{x}}_{\mathcal{B}_i} = (\boldsymbol{\Omega}_{\mathcal{I}_i} \boldsymbol{x}_{\mathcal{B}_i}) = (R_i \hat{\boldsymbol{\Omega}}_{\mathcal{B}_i} R_i^\top \boldsymbol{x}_{\mathcal{B}_i}). \tag{A.17}$$

Accordingly, $\dot{q}_i$ can be computed by the flat outputs and their derivatives using (A.16). Following the similar procedure starting by differentiating (A.10), the following equation can be computed to find $\dot{p}_i$:

$$-\dot{f}_i p_i - f_i \dot{p}_i = m_i(\dot{\boldsymbol{y}}_{\mathcal{B}_i}^\top \ddot{\boldsymbol{x}}_i + \boldsymbol{y}_{\mathcal{B}_i}^\top \dddot{\boldsymbol{x}}_i) + \dot{\boldsymbol{y}}_{\mathcal{B}_i}^\top (\dot{T_i \boldsymbol{q}_i}) + \boldsymbol{y}_{\mathcal{B}_i}^\top (\ddot{T_i \boldsymbol{q}_i}). \tag{A.18}$$

Similarly, the one unrecognized term $\dot{\boldsymbol{x}}_{\mathcal{B}_i}$ can be expressed as follows:

$$\dot{\boldsymbol{y}}_{\mathcal{B}_i} = (\boldsymbol{\Omega}_{\mathcal{I}_i} \boldsymbol{y}_{\mathcal{B}_i}) = (R_i \hat{\boldsymbol{\Omega}}_{\mathcal{B}_i} R_i^\top \boldsymbol{y}_{\mathcal{B}_i}). \tag{A.19}$$

The last component of the angular acceleration, $\dot{r}_i$, needs to be computed, the first step of which is computing derivative of r_i expressed in (A.13):

$$\begin{aligned}\frac{d}{dt} \|\boldsymbol{x}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\| r_i &+ \|\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\| \dot{r}_i \\ &= \ddot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + \dot{\psi}_i \dot{\boldsymbol{x}}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + \dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \dot{\boldsymbol{x}}_{\mathcal{B}_i} \\ &\qquad + \dot{q}_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} + q_i \dot{\boldsymbol{y}}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} + q_i \boldsymbol{y}_{\mathcal{C}_i}^\top \dot{\boldsymbol{z}}_{\mathcal{B}_i}.\end{aligned} \tag{A.20}$$

The $\frac{d}{dt} \|\boldsymbol{x}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\|$ term in the left-hand side of (A.20) needs to be expressed by the flat outputs as follows:

$$\begin{aligned}\frac{d}{dt} \|\boldsymbol{x}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\| = \frac{d}{dt} \|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\| &= \frac{d}{dt} \sqrt{\tilde{\boldsymbol{x}}_{\mathcal{B}_i}^\top \tilde{\boldsymbol{x}}_{\mathcal{B}_i}} \\ &= \frac{\tilde{\boldsymbol{x}}_{\mathcal{B}_i}^\top \dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i}}{\|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\|} = \boldsymbol{x}_{\mathcal{B}_i}^\top \dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i},\end{aligned} \tag{A.21}$$

using that $\tilde{\boldsymbol{x}}_{\mathcal{B}_i} / \|\tilde{\boldsymbol{x}}_{\mathcal{B}_i}\| = \boldsymbol{x}_{\mathcal{B}_i}$. Using the properties, $\dot{\boldsymbol{y}}_{\mathcal{C}_i} = -\dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}$ and $\dot{\boldsymbol{z}}_{\mathcal{B}_i} = -p_i \boldsymbol{y}_{\mathcal{B}_i}$, (A.20) can be further simplified as follows:

$$\begin{aligned}\boldsymbol{x}_{\mathcal{B}_i}^\top \dot{\tilde{\boldsymbol{x}}}_{\mathcal{B}_i} &= -\dot{\psi}_i \boldsymbol{x}_{\mathcal{B}_i}^\top (\boldsymbol{x}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}) - p_i \boldsymbol{x}_{\mathcal{B}_i}^\top (\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{y}_{\mathcal{B}_i}) \\ &= \dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top (\boldsymbol{x}_{\mathcal{B}_i} \times \boldsymbol{z}_{\mathcal{B}_i}) + p_i \boldsymbol{y}_{\mathcal{C}_i}^\top (\boldsymbol{x}_{\mathcal{B}_i} \times \boldsymbol{y}_{\mathcal{B}_i}) \\ &= -\dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{y}_{\mathcal{B}_i} + p_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i}.\end{aligned} \tag{A.22}$$

The right-hand side of (A.20) can be simplified using the properties, $\dot{\boldsymbol{x}}_{\mathcal{C}_i} = \dot{\psi}_i \boldsymbol{y}_{\mathcal{B}_i}$ and

$\boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} = 0$, as follows:

$$\begin{aligned}
&\ddot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + \dot{\psi}_i \dot{\boldsymbol{x}}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + \dot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \dot{\boldsymbol{x}}_{\mathcal{B}_i} \\
&\qquad + \dot{q}_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} + q_i \dot{\boldsymbol{y}}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} + q_i \boldsymbol{y}_{\mathcal{C}_i}^\top \dot{\boldsymbol{z}}_{\mathcal{B}_i} \\
&= \ddot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + \dot{\psi}_i r_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{y}_{\mathcal{B}_i} - 2\dot{\psi}_i q_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} \\
&\qquad + \dot{q}_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} - p_i q_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{y}_{\mathcal{B}_i}.
\end{aligned} \tag{A.23}$$

Using the results in (A.20), (A.22) and (A.23), $\dot{r}_i$ can be computed as follows:

$$\begin{aligned}
\dot{r}_i = \frac{1}{\|\boldsymbol{y}_{\mathcal{C}_i} \times \boldsymbol{z}_{\mathcal{B}_i}\|} \Big(&\ddot{\psi}_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{x}_{\mathcal{B}_i} + 2\dot{\psi}_i r_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{y}_{\mathcal{B}_i} \\
&- 2\dot{\psi}_i q_i \boldsymbol{x}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} - p_i q_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{y}_{\mathcal{B}_i} \\
&- p_i r_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} + \dot{q}_i \boldsymbol{y}_{\mathcal{C}_i}^\top \boldsymbol{z}_{\mathcal{B}_i} \Big).
\end{aligned} \tag{A.24}$$

In summary, it is shown that the states and the inputs can be shown as functions of the flat outputs and their derivatives. To compute p_i and q_i as in (A.16) and (A.18), the fourth-order derivatives of the multi-rotor positions are required. The derivatives of the multi-rotor positions can be computed from the derivatives of the load positions as shown in (A.3) except that the position of the first multi-rotor requires the second-order derivative of the load position as shown in (A.1). Therefore, the highest differentiation order of $\boldsymbol{x}_L$ and $\boldsymbol{T}_i$ for the computation is six and four, i.e. $\eta_L = 6$ and $\eta_T = 4$. On the other hand, the highest derivative order of ψ_i is two as shown in (A.20).

B

Preliminaries of Bernstein Polynomials

B.1 Definition of a Bernstein Polynomial

A Bernstein polynomial of degree d_B can be expressed with control points and Bernstein basis polynomials of degree d_B which are c_j and $B_{d_B.j}(t)$ for $j = 1, 2, \cdots, d_B{+}1$, respectively. Bernstein basis polynomials are defined as follows:

$$B_{d_B.j}(t) = \binom{d_B}{j-1} t^{j-1}(1-t)^{d_B+1-j} \;\; for \;\; t \in [0,1]\,. \tag{B.1}$$

Then, a Bernstein polynomial of degree d_B with the control points is expressed as follows:

$$p(t) = \sum_{j=1}^{d_B+1} c_j B_{d_B.j}(t). \tag{B.2}$$

For more flexibility with a longer duration, a trajectory can be composed of several Bernstein polynomials occupying each time segment. The trajectory with n_B segments can be expressed as follows:

$$p(t) = \begin{cases} \sum_{j=1}^{d_B+1} c_j^1 B_{d_B \cdot j}(\tau_1) & for\ t \in [t_0, t_1] \\ \sum_{j=1}^{d_B+1} c_j^2 B_{d_B \cdot j}(\tau_2) & for\ t \in [t_1, t_2] \\ \vdots & \vdots \\ \sum_{j=1}^{d_B+1} c_j^{n_B} B_{d_B \cdot j}(\tau_{n_B}) & for\ t \in [t_{n_B-1}, t_{n_B}] \, , \end{cases} \tag{B.3}$$

where $\tau_k = \frac{t-t_{k-1}}{t_k-t_{k-1}}$, c_j^k is the j^{th} control point of the k^{th} segment of the trajectory $p(t)$. t_{k-1} and t_k are the start and end time of the k^{th} segment.

B.2 Convex hull property of a Bernstein Polynomial

The most interesting characteristic of a Bernstein polynomial is a convex hull property. The property means that the polynomial resides in the convex hull built from the control points. As an example, two dimensional Bernstein polynomials of degree two can be expressed using (B.2) as follows:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{3} c_{x.j} B_{2.j}(t) \\ \sum_{j=1}^{3} c_{y.j} B_{2.j}(t) \end{bmatrix}, \tag{B.4}$$

where $c_{x.j}$ and $c_{y.j}$ are the control points of the x and y axes polynomials, respectively. It can easily computed that $\sum_{j=1}^{3} B_{2.j}(t) = 1$ and $B_{2.j}(t) \geq 0$ for $t \in [0,1]$. Then, the polynomials on the right-hand side of (B.4) are convex combinations of the control points, $c_{x.j}$ and $c_{y.j}$, which means the minimum and maximum values of the continuous polynomials can be controlled by limiting the control points as follows:

$$\begin{bmatrix} min\left\{\boldsymbol{c}_x\right\} \\ min\left\{\boldsymbol{c}_y\right\} \end{bmatrix} \preceq \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \preceq \begin{bmatrix} max\left\{\boldsymbol{c}_x\right\} \\ max\left\{\boldsymbol{c}_y\right\} \end{bmatrix}, \tag{B.5}$$

where $\boldsymbol{c}_x = [c_{x.1}\ c_{x.2}\ c_{x.3}]^\top$ and $\boldsymbol{c}_y = [c_{y.1}\ c_{y.2}\ c_{y.3}]^\top$.

B.3 Representation of a General Polynomial with Bernstein Basis Polynomials

This subsection addresses a method to express a general polynomial as a Bernstein polynomial. A general polynomial of degree d_B can be expressed as follows:

$$\begin{aligned} q(t) &= [s_q^{d_B}\ s_q^{d_B-1}\ \cdots\ s_q^1\ s_q^0][t^{d_B}\ t^{d_B-1}\ \cdots\ t\ 1]^\top \\ &= \boldsymbol{s}_q^\top \boldsymbol{t}_{d_B}, \end{aligned} \tag{B.6}$$

where $\boldsymbol{s}_q \in \mathbb{R}^{d_B+1}$ is the coefficient vector of $q(t)$. The Bernstein polynomial (B.2) is expressed in a different way:

$$\begin{gathered} p(t) = \left(S_{B_{d_B}} \boldsymbol{c}\right)^\top \boldsymbol{t}_{d_B}, \\ \boldsymbol{c} = [c_1\ c_2\ \cdots\ c_{d_B}\ c_{d_B+1}]^T, \\ S_{B_{d_B}} = [\boldsymbol{s}_{B_1}\ \boldsymbol{s}_{B_2}\ \cdots\ \boldsymbol{s}_{B_{d_B}}\ \boldsymbol{s}_{B_{d_B+1}}], \\ \boldsymbol{s}_{B_{d_B \cdot j}} = [s_{B_{d_B \cdot j, d_B}}\ s_{B_{d_B \cdot j, d_B-1}}\ \cdots\ s_{B_{d_B \cdot j,1}}\ s_{B_{d_B \cdot j,0}}]^T. \end{gathered} \tag{B.7}$$

$\boldsymbol{s}_{B_{d_B \cdot j}}$ includes the coefficients of the j^{th} Bernstein basis polynomial, $B_{d_B \cdot j}$, in a descending order. By computing (B.1), it can be easily found that $S_{B_{d_B}}$ is symmetric. Accordingly, the control points for representation of $q(t)$ can be computed as follows:

$$\boldsymbol{c} = S_{B_{d_B}}^{-1} \boldsymbol{s}_q. \tag{B.8}$$

B.4 Representation of the Derivative of a Bernstein Polynomial with Bernstein Basis Polynomials

When a Bernstein polynomial of degree d_B, (B.2), is differentiated r times, the derivative polynomial is expressed as follows:

$$p^{(\eta)}(t) = \sum_{j=1}^{d_B+1} c_j B^{(\eta)}_{d_B.j}(t) = (S_{B^{(\eta)}_{d_B}} \boldsymbol{c})^T \boldsymbol{t}_{d_B-\eta}, \tag{B.9}$$

where $S_{B^{(\eta)}_{d_B}} \in \mathbb{R}^{(d_B+1-\eta)\times(d_B+1)}$ is a coefficient matrix of $B^{(\eta)}_{d_B}(t)$ similarly defined as (B.7). Since the differentiated polynomials, $B^{(\eta)}_{d_B}(t)$, are not Bernstein basis polynomials, the derivative polynomial no longer has the convex hull property. To generate various constraints for the derivative polynomials, they are expressed with Bernstein basis polynomials of an order $d_B - \eta$ as follows:

$$\begin{gathered}(S_{B^{(\eta)}_{d_B}} \boldsymbol{c})^T \boldsymbol{t}_{d_B-\eta} = \left(S_{B_{d_B-\eta}} \hat{\boldsymbol{c}}\right)^T \boldsymbol{t}_{d_B-\eta}, \\ \hat{\boldsymbol{c}} = (S_{B_{d_B-\eta}})^{-1} S_{B^{(\eta)}_{d_B}} \boldsymbol{c},\end{gathered} \tag{B.10}$$

where $S_{B_{d_B-\eta}} \in \mathbb{R}^{(d_B-\eta+1)\times(d_B-\eta+1)}$ and $\hat{\boldsymbol{c}} \in \mathbb{R}^{d_B-\eta+1}$ are coefficient matrix of $B_{d_B-\eta.j}(t)$ and control points of a new Bernstein polynomial of an order $d_B - \eta$, respectively. Accordingly, the equation (B.10) means that the η^{th} order derivative of a Bernstein polynomial of an order d_B can be expressed as another Bernstein polynomial of an order $d_B - \eta$.

References

[1] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2011, pp. 2668–2673.

[2] H. Seo, S. Kim, and H. J. Kim, "Locally optimal trajectory planning for aerial manipulation in constrained environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2017, pp. 1719–1724.

[3] D. Lee, H. Seo, I. Jang, S. J. Lee, and H. J. Kim, "Aerial manipulator pushing a movable structure using a dob-based robust controller," *IEEE Robotics and Automation Letters*, 2020.

[4] C. Y. Son, H. Seo, T. Kim, and H. J. Kim, "Model predictive control of a multi-rotor with a suspended load for avoiding obstacles," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 5233–5238.

[5] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time optimal planning and model predictive control of a multi-rotor with a suspended load," in *2019 International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 5665–5671.

[6] S. Tang, V. Wüest, and V. Kumar, "Aggressive flight with suspended payloads using vision-based control," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1152–1159, 2018.

[7] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC).* IEEE, 2010, pp. 5420–5425.

[8] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system," in *2013 International Conference on Robotics and Automation (ICRA).* IEEE, 2013, pp. 4888–4895.

[9] C. Y. Son, H. Seo, D. Jang, and H. J. Kim, "Real-time optimal trajectory generation and control of a multi-rotor with a suspended load for obstacle avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1915–1922, 2020.

[10] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on.* IEEE, 2016, pp. 139–146.

[11] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 344–351.

[12] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1398–1404.

[13] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açikmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on.* IEEE, 2017, pp. 4862–4868.

[14] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4063–4069, 2017.

[15] S. Tang and V. Kumar, "Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015, pp. 2216–2222.

[16] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload." in *Robotics: Science and Systems*, 2017, pp. 1–10.

[17] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear control with swing damping of a multirotor uav with suspended load," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 379–394, 2017.

[18] M. Bisgaard, A. la Cour-Harbo, and J. D. Bendtsen, "Adaptive control system for autonomous helicopter slung load operations," *Control Engineering Practice*, vol. 18, no. 7, pp. 800–811, 2010.

[19] G. V. Raffo and M. M. de Almeida, "Nonlinear robust control of a quadrotor uav for load transportation with swing improvement," in *2016 American Control Conference (ACC).* IEEE, 2016, pp. 3156–3162.

[20] G. Yu, D. Cabecinhas, R. Cunha, and C. J. Silvestre, "Nonlinear backstepping control of a quadrotor slung load system," *IEEE/ASME Transactions on Mechatronics*, 2019.

[21] P. J. Cruz and R. Fierro, "Cable-suspended load lifting by a quadrotor uav: hybrid model, trajectory generation, and control," *Autonomous Robots*, pp. 1–15, 2017.

[22] J. E. Trachte, L. F. G. Toro, and A. McFadyen, "Multi-rotor with suspended load: System dynamics and control toolbox," in *2015 IEEE Aerospace Conference.* IEEE, 2015, pp. 1–9.

[23] S. Notter, A. Heckmann, A. Mcfadyen, and F. Gonzalez, "Modelling, simulation and flight test of a model predictive controlled multirotor with heavy slung load," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 182–187, 2016.

[24] K. Sreenath and V. Kumar, "Dynamics control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," *rn*, vol. 1, no. r2, p. r3, 2013.

[25] F. A. Goodarzi and T. Lee, "Dynamics and control of quadrotor uavs transporting a rigid body connected via flexible cables," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 4677–4682.

[26] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 324–334, 2011.

[27] B. E. Jackson, T. A. Howell, K. Shah, M. Schwager, and Z. Manchester, "Scalable cooperative transport of cable-suspended loads with uavs using distributed trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3368–3374, 2020.

[28] M. Gassner, T. Cieslewski, and D. Scaramuzza, "Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors," in *ICRA*, no. EPFL-CONF-228458, 2017.

[29] G. Tartaglione, E. D?Amato, M. Ariola, P. S. Rossi, and T. A. Johansen, "Model predictive control for a multi-body slung-load system," *Robotics and Autonomous Systems*, vol. 92, pp. 1–11, 2017.

[30] T. Lee, "Geometric control of quadrotor uavs transporting a cable-suspended rigid body," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 255–264, 2017.

[31] S. Kim, S. Choi, H. Kim, J. Shin, H. Shim, and H. J. Kim, "Robust control of an equipment-added multirotor using disturbance observer," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1524–1531, 2017.

[32] M. Bisgaard, J. D. Bendtsen, and A. la Cour-Harbo, "Modeling of generic slung load system," *Journal of guidance, control, and dynamics*, vol. 32, no. 2, pp. 573–585, 2009.

[33] K. K. Dhiman, A. Abhishek, and M. Kothari, "Cooperative load control and transportation," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 0895.

[34] A. Franchi and A. Mallet, "Adaptive closed-loop speed control of bldc motors with applications to multi-rotor aerial vehicles," in *2017 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2017, pp. 5203–5208.

[35] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018.

[36] S. J. Lee, S. H. Kim, and H. J. Kim, "Robust translational force control of multi-rotor uav for precise acceleration tracking," *IEEE Transactions on Automation Science and Engineering*, 2019.

[37] A. Koubâa, *Robot Operating System (ROS).* Springer, 2017.

[38] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications.* Springer Science & Business Media, 2012.

[39] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2017, pp. 5674–5680.

[40] H. Seo, D. Lee, C. Y. Son, C. J. Tomlin, and H. J. Kim, "Robust trajectory planning for a multirotor against disturbance based on hamilton-jacobi reachability analysis," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE (accepted), 2019.

[41] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[42] K. Sreenath, T. Lee, and V. Kumar, “Geometric control and differential flatness of a quadrotor uav with a cable-suspended load,” in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 2269–2274.

[43] F. Gao, W. Wu, Y. Lin, and S. Shen, “Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 344–351.

[44] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.1,” http://cvxr.com/cvx, Mar. 2014.

[45] J. Park, J. Kim, I. Jang, and H. J. Kim, “Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 434–440.

국 문 초 록

경로 계획과 제어는 안전하고 안정적으로 멀티로터를 운용하기 위해서 필수적인 요소이다. 충돌을 회피하며 효율적인 경로를 생성하고 이를 실제로 추종하기 위해서는 동역학 모델이 고려되어야 한다. 일반 멀티로터의 동역학 모델은 높은 차원을 가진 비선형식으로 표현되는데, 현수 운송 물체를 추가할 경우 계산이 더욱 복잡해진다. 본 논문은 멀티로터를 이용한 현수 운송에 있어 경로 계획과 제어에 대한 효율적인 기법을 제안한다.

첫 번째로 단일 멀티로터를 이용한 현수 운송을 다룬다. 물체가 별도의 엑츄에이터 없이 운송될 경우 물체는 기체의 움직임에 의해서만 제어가 가능하다. 하지만, 동역학식의 높은 비선형성으로 운용에 어려움이 존재한다. 이를 경감시키기 위해서 회전 동역학식의 비선형성을 줄이고 자세 제어에 존재하는 시간 지연을 고려하여 동역학식을 간소화한다. 경로 계획에 있어서는 충돌 회피를 위해 기체, 케이블, 그리고 운송 물체를 다른 크기와 모양을 가진 타원체들로 감싸며, 효과적이면서도 덜 보수적인 방식으로 충돌 회피 구속조건을 부과한다. Augmented Lagrangian 방법을 이용하여 비선형 구속조건이 부과된 비선형 문제를 실시간 최적화하여 경로를 생성한다. 생성된 경로를 추종하기 위해서 Sequential linear quadratic 솔버를 이용한 모델 예측 제어기로 최적 제어 입력을 계산한다. 제안된 기법은 여러 시뮬레이션과 실험을 통해 검증한다.

다음으로, 다중 멀티로터를 이용한 협업 현수 운송 시스템을 다룬다. 해당 시스템의 상태 변수나 동역학식에서 연결된(coupled) 항의 개수는 기체의 수에 비례하여 증가하기 때문에, 효과적인 기법 없이는 최적화에 많은 시간이 소요된다. 높은 비선형성을 가진 동역학식의 복잡성을 낮추기 위하여 미분 평탄성을 사용한다. 경로 또한 piece-wise Bernstein 다항식을 이용하여 매개변수화하여 최적화 변수의 개수를 줄인다. 최적화 문제를 분해하고 충돌 회피 구속조건들에 대해 볼록화(convexification)를 수행하여 운송 물체의 경로와 장력의 경로에 대한 볼록한(convex) 하위문제들이 만들어진다. 첫 번째 하위문제인 물체 경로 생성에서는, 장애물 회피와 멀티로터의 공간을 확보하기 위하여 안전 비행 통로(safe flight corridor, SFC)와 여유 간격 구속조건을 고려하여 최적화한다. 다음으로, 장력 벡터들의 경로는 장애물

회피와 상호 충돌을 방지하기 위하여 안전 비행 섹터(safe flight sector, SFS)와 상대 안전 비행 섹터(relative safe flight sector, RSFS) 구속조건을 부과하여 최적화한다. 시뮬레이션과 실험으로 복잡한 환경에서 효율적인 경로 계획 기법을 시연하며 검증한다.

주요어: 공중 조작, 움직임 및 경로 계획, 최적화 및 최적 제어, 다개체

학 번: 2015-22735