EE5003 - PROJECT REPORT

NATIONAL UNIVERSITY OF SINGAPORE

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Distributed Learning and Control of Cooperative Quadrotor Load Transportation

by:

Gao Yichao (ID: A0298755E)

yichao_gao@u.nus.edu

Supervisor: Dr. Zhao Lin

Examiner: Prof. Ge Shuzhi Sam

April 2025

Abstract

Multi-lift aerial transportation, in which multiple quadrotors collaboratively carry a common payload via cables, has been extensively explored to enhance load-carrying capacity and increase system redundancy. However, existing simulation platforms and software pipelines have not matured to meet the complex requirements of such tasks. In particular, there is currently no comprehensive simulator that seamlessly integrates all the necessary components for multi-lift research: Isaac Sim for high-fidelity physics, PX4 for low-level flight control, ROS for communication and orchestration, and self-defined reinforcement learning (RL) algorithms for high-level policy training. The absence of such toolchain not only limits the ability to test advanced control and learning strategies but also makes it difficult to bridge the gap between simulated performance and real-world behavior.

In this work, we develop a high-fidelity software-in-the-loop (SITL) framework for multi-lift aerial transportation that addresses these challenges. Building on Isaac Sim's Temporal Gauss–Seidel (TGS) solver, we introduce a mass–spring–damper (MSD) cable model capable of handling highly stiff tethers under real-time constraints. This ensures a stable and accurate representation of cable dynamics in various cooperative load-sharing scenarios. Meanwhile, we integrate PX4-based flight controllers and ROS 2 communication layers to achieve minimal-latency data exchange and synchronized state estimation. A geometric controller with decoupled yaw regulation is implemented as a robust low-level flight-control strategy, while the system architecture accommodates self-defined RL modules for distributed learning of high-level control policies.

Through extensive tests—ranging from basic cable-dynamics validation to cooperative trajectory tracking with multiple UAVs—the proposed platform demonstrates reliable real-time operation and consistent tracking accuracy. In addition, the design naturally accommodates wind disturbances and varying payload conditions, thus expanding the scope of scenarios that can be evaluated. While this framework lays a strong foundation for the simulation of multi-lift systems, future efforts will focus on refining aerodynamic models (e.g., rotor wash, turbulence) and incorporating hardware-in-the-loop components for more seamless transitions to physical flight tests. Overall, the presented SITL environment provides a long-awaited solution for validating advanced learning and control algorithms in multi-lift tasks and paves the way for safer, more efficient aerial cargo transport.

Acknowledgement

My time as a Master's student at the NUS - ECE has been both a challenging and enriching experience. Over the course of this journey, I've had the privilege to engage in academic research, explore new areas within engineering, and develop both personally and intellectually. This report is a reflection of that journey, which is shaped not only by my efforts but also by the generous support, guidance, and encouragement I've received from many along the way.

Foremost among those I wish to acknowledge is my supervisor, Dr. Zhao Lin, whose exceptional mentorship has been profoundly influential throughout my academic journey. Prof. Zhao has provided invaluable guidance that has significantly shaped the direction and depth of my research. His insightful feedback and unwavering support have greatly enriched my understanding of these complex fields and have been instrumental in my academic and personal development.

I would also like to extend my thanks to Dr. Wang Bingheng for his support in both academic research and coursework. Dr. Wang is a dedicated scholar, whose seriousness and unwavering commitment to scientific inquiry have profoundly influenced me.

In addition, I would like to express my gratitude for the support of many researchers and friends, including but not limited to Dr. He Lei, Sun Tianchen, Sima Kuankuan, Huang Rui and Chen Xin. Their advice and insightful perspectives on academic topics have significantly contributed to my progress in research. Special thanks go to Tang Longbin, for making both coursework and research endeavors productive and enjoyable.

Lastly, I extend my deepest gratitude to my family—my parents and my sister—for their unwavering love and support. Their encouragement and belief in me have been my foundation throughout this journey, giving me the strength and confidence to pursue my goals.

Declaration

I hereby declare that this report is my original work and it has been written by me in its entirety.
I have duly acknowledged all the sources of information which have been used in the report.

April 1st, 2025

Contents

List of Figures

1 Introduction

Aerial transportation using multiple quadrotors has garnered substantial interest in the robotics and unmanned aerial vehicle (UAV) communities due to its enhanced load-carrying capacity and improved robustness compared to single-vehicle systems. By attaching the payload via lightweight cables, multi-lift configurations can accommodate large and potentially heavy loads. However, these systems are inherently complex: each quadrotor's motion is constrained by cable length when taut, and dynamic coupling through the cables introduces intricate force interactions among vehicles and the load. Additionally, the system exhibits hybrid dynamics owing to transitions between slack and taut states, rendering motion planning and control significantly more challenging. Achieving safe and effective coordination requires addressing cable slack avoidance, maintaining inter-vehicle distances, managing control limits, and ensuring scalability with an increasing number of UAVs.

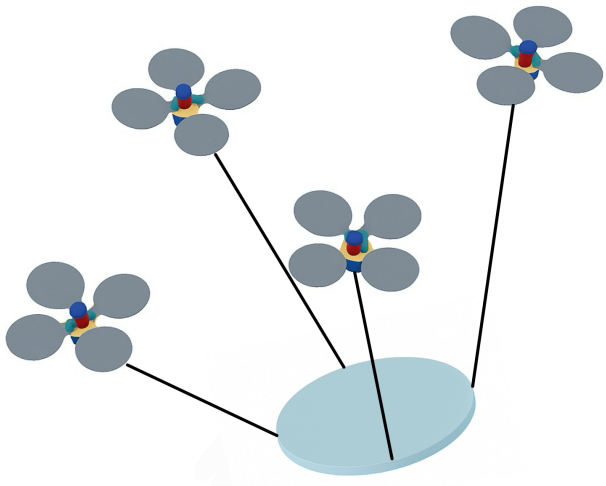

Figure 1: Illustration of Multilift Problems

Early investigations into multi-lift aerial transport often simplified the payload as a point mass, wherein cable tensions were treated as external disturbances acting on the quadrotors [1],[2]. Although these models facilitated initial theoretical and experimental studies, they limited achievable maneuvers to near-quasistatic or slow trajectories. Subsequent research endeavored to capture the dynamic coupling between quadrotors and payload more accurately. In particular, [3] introduced a geometric control framework that explicitly accounts for the payload's dynamics as a point mass. Further enhancements included modeling the payload as a full six-degrees-of-freedom rigid body, enabling higher-fidelity control but demanding at least three quadrotors for complete attitude and position regulation [4],[5].

More recent work has pushed toward agile and robust control of cable-suspended loads.

Sun [6] proposed a model predictive control (MPC)-based solution for high-acceleration payload transport without onboard sensors, addressing dynamic coupling and load uncertainties. However, the reliance on carefully tuned MPC parameters remained a practical challenge. Wang [7] introduced a distributed, learning-based approach that enables real-time trajectory planning and control in cluttered environments, though this method simplifies the payload to a point mass and neglects its full dynamic behavior.

Auto-Multilift [8] offers a distributed closed-loop learning framework to automatically adapt MPC hyperparameters for multi-lift systems. At its core, a Distributed Sensitivity Propagation (DSP) algorithm is employed to compute closed-loop state sensitivities in parallel, informing a distributed policy gradient method that trains deep neural networks using tracking error feedback. This approach leverages the Safe-PDP-based MPC gradient solver, improving scalability and ensuring robust trajectory tracking compared to open-loop MPC parameter tuning.

To investigate the stability and practical feasibility of Auto-Multilift, a high-fidelity simulator that accurately captures multi-lift aerial systems is essential. Hence, this work extends Isaac Sim and Pegasus [9] with refined quadrotor dynamics and flexible cable modeling. In particular, the cable model is formulated as a mass-spring-damper (MSD) system with multiple joints, capturing the high stiffness required in real-world tethering scenarios. Traditional simulators, such as Gazebo and MuJoCo, often struggle to simulate such highly stiff, multi-joint connections in real time. By leveraging the high computational power and stability of PhysX through the fast TGS (Temporal Gauss-Seidel) numerical solver, Isaac Sim can now operate in real time while accurately reproducing the physics of complex cable dynamics.

A ROS-based communication pipeline is also developed to facilitate software-in-the-loop (SITL) simulation with PX4, ensuring precise time synchronization and reliable exchange of states between the flight controller, ground station, and simulation environment. Additionally, a geometric controller with decoupled yaw-axis regulation is integrated as the low-level controller for individual UAV stabilization and trajectory tracking.

The main contributions of this work are as follows:

1. **Development of a high-fidelity simulation environment for multi-lift aerial systems**: This environment, built upon Isaac Sim and Pegasus, incorporates improved quadrotor and cable dynamics through an MSD model with multiple joints.
2. **Design of a ROS-based SITL pipeline**: A specialized communication framework is implemented to ensure consistent time synchronization and robust state exchange among PX4, the ground control station, and the simulator.
3. **Integration of a geometric low-level controller**: A geometric controller with decoupled yaw-axis dynamics is presented, enabling precise single-UAV stabilization and control in multi-lift operations.

2 Simulator Design

2.1 Quadrotor Dynamics

In this subsection, we present a comprehensive overview of the quadrotor's translational and rotational equations of motion. We then highlight the improvements introduced in the Pegasus Simulator, which result in a more accurate representation of quadrotor dynamics. Notably, all rotational quantities are expressed on the special orthogonal group SO(3), ensuring a globally valid description of the quadrotor's attitude.

Define an inertial frame $\{\mathcal{I}\}$ and a body frame $\{\mathcal{B}\}$ fixed to the quadrotor's center of mass. Let $\mathbf{p} \in \mathbb{R}^3$ denote the position of the quadrotor's center of mass expressed in the inertial frame, and let $\mathbf{R} \in \mathrm{SO}(3)$ describe the orientation of the body frame with respect to the inertial frame. The time derivative $\dot{\mathbf{p}} \in \mathbb{R}^3$ denotes the linear velocity, and the angular velocity in the body frame is $\boldsymbol{\Omega} \in \mathbb{R}^3$. The vehicle has mass $m > 0$, a (typically diagonal) moment of inertia matrix $\mathbf{I} \in \mathbb{R}^{3\times3}$, and experiences gravity $g > 0$ acting in the $-z$ direction of the inertial frame.

Translational Dynamics. Newton's second law gives the translational dynamics of the quadrotor's center of mass. The primary external forces are gravity and the total thrust generated by the four rotors. Let $T \geq 0$ denote the total thrust, which acts along the body's z-axis. Transformed into the inertial frame, the thrust vector becomes $\mathbf{R}\,\mathbf{e}_3\,T$, where $\mathbf{e}_3 = [\,0 \;\; 0 \;\; 1\,]^\top$. Consequently, the translational dynamics are

$$m\,\ddot{\mathbf{p}} \;=\; -m\,g\,\mathbf{e}_3 \;+\; T\,\mathbf{R}\,\mathbf{e}_3. \tag{1}$$

Rotational Kinematics. The evolution of the orientation matrix $\mathbf{R}$ is governed by

$$\dot{\mathbf{R}} \;=\; \mathbf{R}\,\widehat{\boldsymbol{\Omega}}, \tag{2}$$

where $\widehat{\boldsymbol{\Omega}}$ is the skew-symmetric matrix associated with the cross product in $\mathbb{R}^3$, namely

$$\widehat{\boldsymbol{\Omega}} \;=\; \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix}. \tag{3}$$

Rotational Dynamics. The rotational motion obeys Euler's equation in the body frame:

$$\mathbf{I}\,\dot{\boldsymbol{\Omega}} \;+\; \boldsymbol{\Omega} \times \Big(\mathbf{I}\,\boldsymbol{\Omega}\Big) \;=\; \boldsymbol{\tau}, \tag{4}$$

where $\boldsymbol{\tau} \in \mathbb{R}^3$ is the net torque in the body frame. For a symmetric quadrotor, $\mathbf{I} = \mathrm{diag}(I_x, I_y, I_z)$ and the cross-product terms simplify, but we retain the general form in Eq. (4) for completeness.

Thrust Allocation. Each rotor generates a thrust force proportional to the square of its rotational speed. Let ω_i be the angular speed of rotor i, and let $b > 0$ denote the thrust coefficient. Then the thrust of rotor i is

$$f_i = b\,\omega_i^2, \quad i = 1, \ldots, 4. \tag{5}$$

Hence the total thrust T is

$$T \;=\; \sum_{i=1}^{4} f_i \;=\; b\Big(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\Big). \tag{6}$$

Additionally, each rotor creates a torque about its own axis, often modeled as

$$\tau_{\text{drag},i} = d\,\omega_i^2, \tag{7}$$

where $d > 0$ is the drag coefficient. The *net* torque in the body frame arises from the combination of torques due to (1) the lateral distance of each rotor thrust from the center of mass, and (2) the drag torques induced by rotor spinning. Labeling the four rotors such that ω_1 and ω_3 spin in one direction, and ω_2 and ω_4 spin in the opposite direction, one obtains the commonly used expressions for roll, pitch, and yaw torques:

$$\tau_\phi \;=\; \ell\, b\,(\omega_2^2 - \omega_4^2), \qquad \tau_\theta \;=\; \ell\, b\,(\omega_3^2 - \omega_1^2), \qquad \tau_\psi \;=\; d\,(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2), \tag{8}$$

where ℓ is the length of each rotor arm. Stacking these into the vector $\boldsymbol{\tau} = [\,\tau_\phi \;\; \tau_\theta \;\; \tau_\psi\,]^\top$ yields

$$\boldsymbol{\tau} \;=\; \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{9}$$

A common approach is to collect the force and torque components into a single control vector

$$\mathbf{u} \;=\; \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}, \quad \text{and define} \quad \boldsymbol{\omega}^2 \;=\; \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \tag{10}$$

Then,

$$\mathbf{u} \;=\; \begin{bmatrix} b & b & b & b \\ 0 & \ell\, b & 0 & -\ell\, b \\ -\ell\, b & 0 & \ell\, b & 0 \\ d & -d & d & -d \end{bmatrix} \boldsymbol{\omega}^2, \tag{11}$$

where the 4×4 matrix on the right-hand side is referred to as the *allocation matrix*. By inverting or pseudo-inverting this matrix, one can obtain the required rotor speeds (or squared speeds) to achieve a desired thrust and torque vector.

The Pegasus Simulator incorporates detailed quadrotor dynamics within the simulation environment. However, it does not account for wind effects or aerodynamic drag [10], which can lead to inaccuracies and contribute to a significant sim-to-real gap. To further reduce this gap, we extend the simulator by incorporating wind disturbances and rolling moments, effectively modeling aerodynamic drag and external disturbances more realistically.

$$\mathbf{I}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \left(\mathbf{I}\boldsymbol{\Omega}\right) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{dist}}, \tag{12}$$

where

$$\boldsymbol{\tau}_{\text{dist}} = \begin{bmatrix} \tau_{\phi_d} & 0 & 0 \end{bmatrix}^\top, \tag{13}$$

represents a wind-induced moment acting primarily about the roll axis, $\tau_{\phi_d} \in \mathbb{R}$. By modeling both aerodynamic drag and rolling-moment disturbances linearly, the simulator achieves a more realistic approximation of quadrotor flight under varying wind conditions.

2.2 Cable Modeling

In this subsection, we introduce the mass–spring–damper (MSD) model to represent the dynamics of a flexible cable. This modeling approach is frequently employed in the simulation of slender mechanical systems under various loading and boundary conditions. The model is constructed by discretizing the cable into a finite number of lumped masses connected by linear or nonlinear springs and dampers.

To outline the derivation, let us assume that the cable is subdivided into N segments, each of which is characterized by a small mass Δm. We denote the displacement of each mass i at time t by $\mathbf{x}_i(t) \in \mathbb{R}^3$. The governing equations arise from applying Newton's second law to each segment, taking into account both the internal elastic and damping forces between adjacent masses.

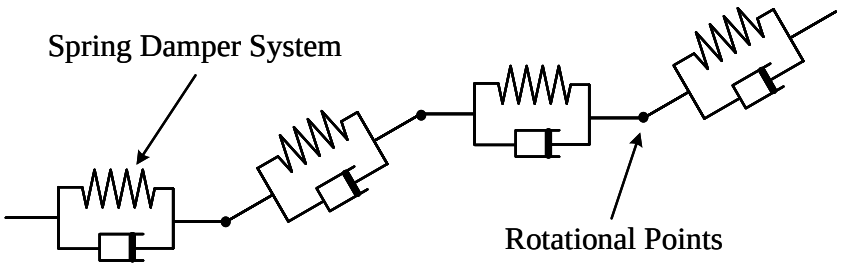

Figure 2: Illustration of MSD model for cables.

Spring forces In the simplest linear case, each segment is modeled using Hooke's law, where the spring force $\mathbf{F}^{(s)}$ between two adjacent masses i and $i+1$ is given by

$$\mathbf{F}^{(s)}_{i,i+1} = k\Big(\|\mathbf{x}_{i+1} - \mathbf{x}_i\| - \ell_0\Big)\hat{\mathbf{r}}_{i,i+1}, \tag{14}$$

where

- k is the tensile stiffness of the cable,
- ℓ_0 is the rest length of the segment,
- $\hat{\mathbf{r}}_{i,i+1} = \dfrac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|}$ is the unit vector pointing from mass i to mass $i+1$.

This formulation represents the elastic restoring force along the direction of the segment.

Damping forces To account for energy dissipation due to internal friction and relative motion, a linear viscous damping force is included. The damping force $\mathbf{F}^{(d)}$ between masses i and $i+1$ is given by

$$\mathbf{F}^{(d)}_{i,i+1} = c\Big[\Big(\dot{\mathbf{x}}_{i+1} - \dot{\mathbf{x}}_i\Big) \cdot \hat{\mathbf{r}}_{i,i+1}\Big]\hat{\mathbf{r}}_{i,i+1}, \tag{15}$$

where

- c is the damping coefficient,
- $\dot{\mathbf{x}}_{i+1}$ and $\dot{\mathbf{x}}_i$ are the velocities of the adjacent masses.

The damping force acts purely along the segment and is proportional to the **relative velocity** in that direction.

Equation of motion By summing the forces acting on each mass i and applying Newton's second law, the equation of motion becomes

$$\Delta m\, \ddot{\mathbf{x}}_i = \sum_{j \in \mathcal{N}(i)} \Big(\mathbf{F}^{(s)}_{i,j} + \mathbf{F}^{(d)}_{i,j}\Big) + \mathbf{F}_{\text{ext}}, \tag{16}$$

where

- Δm is the mass associated with each segment (or each node),
- $\mathcal{N}(i)$ denotes the set of neighbors directly connected to mass i,
- $\mathbf{F}_{\text{ext}}$ includes any external forces (e.g., gravity, contact, or actuation).

Equations (14)–(16) constitute the core of the MSD model for flexible cable dynamics. This framework can be extended to incorporate geometric nonlinearities, bending stiffness, or more complex damping models. Nonetheless, the fundamental principle remains the same: each discrete segment contributes its own inertia, elastic restoring force, and velocity-dependent damping, which together capture the global dynamic response of the cable.

The configuration of the cable within the simulator is illustrated in Figure 3. The rigid body link primarily represents the mass component in the physical model, while the D6 joint corresponds to a six-degree-of-freedom (6-DOF) joint, allowing translational and rotational motion along all axes. The elastic and damping properties of the system are defined by specifying appropriate stiffness and damping coefficients. By connecting the rigid body link and the D6 joint in series, the model effectively captures the cable's flexibility and elasticity.

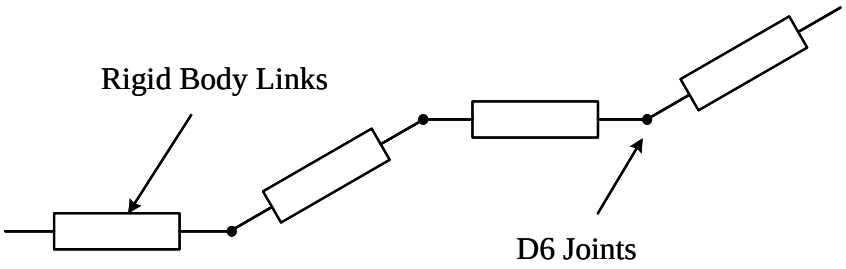

Figure 3: Implementation of MSD models in the simulator.

Although the mass–spring–damper (MSD) model for cable dynamics is conceptually straightforward and relatively simple to implement, numerical challenges can arise in practical simulators. In many physics engines, the internal solver for dynamics relies on iterative methods such as the Gauss–Seidel (GS) algorithm to solve systems of equations arising from joint or constraint forces. For example, consider a linear system of the form

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}, \tag{17}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a coefficient matrix, $\mathbf{x} \in \mathbb{R}^n$ is the unknown vector, and $\mathbf{b} \in \mathbb{R}^n$ is the right-hand side vector. The Gauss–Seidel method solves the equation (17) iteratively by updating one component of $\mathbf{x}$ at a time, using the most recent updates as they become available:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\,x_j^{(k+1)} - \sum_{j>i} a_{ij}\,x_j^{(k)}\right), \quad i = 1, 2, \ldots, n, \tag{18}$$

where $x_i^{(k)}$ denotes the i-th component of the vector $\mathbf{x}$ at iteration k. The iteration sweeps through $i = 1$ to n, then repeats until convergence (or failure to converge).

In an MSD system, each cable or joint often introduces large entries in the global stiffness matrix $\mathbf{A}$, particularly when the spring constant (k) or damping coefficient (c) is

high. While robust time-integration schemes and solver strategies can handle moderate stiffness, when k or c become excessively large, the matrix $\mathbf{A}$ can become ill-conditioned or exhibit very high condition numbers. This leads to the following potential issues in a Gauss–Seidel solver:

1. **Slow convergence or divergence.** If $\mathbf{A}$ is poorly conditioned, the error reduction per iteration of Gauss–Seidel can become extremely slow. In some cases, numerical round-off or small perturbations in the matrix may cause the solver to diverge rather than converge.
2. **Stiffness-induced instability.** In real-time physics simulations, large stiffness or damping forces can make the system "stiff," requiring very small time steps for stability. An iterative solver like Gauss–Seidel may not converge within the allotted iteration budget if the local coupling is too strong.
3. **Limited numerical accuracy.** With large entries on the diagonal (due to high stiffness/damping), errors in floating-point arithmetic can accumulate, degrading precision and further hindering the iterative solver's convergence.

When a system is too stiff, one can employ smaller time-step sizes or add numerical damping schemes to mitigate high-frequency effects. However, all of these strategies may increase computational cost and complexity, thus demanding a careful balance between physical realism, stability, and performance in the simulation of highly stiff or heavily damped MSD cable models.

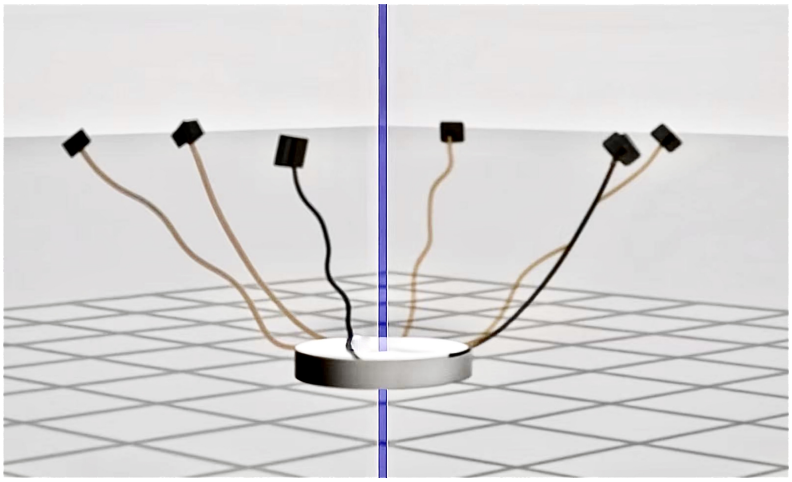

Figure 4: Cable unstable oscillation resulting from Gauss-Seidel numerical failure

To address the aforementioned issues and ensure both the stability and real-time capability of the simulation, two primary strategies were adopted. First, the Temporal Gauss-Seidel (TGS) solver was employed in place of the conventional Gauss-Seidel (GS)

method, offering improved numerical stability under stiff conditions. Second, soft constraints were applied to the stiffness and damping parameters, rather than enforcing them as hard constraints. The following sections provide a detailed discussion of these methods.

Temporal Gauss–Seidel (TGS) Solver Classical Gauss–Seidel (GS) and Projected Gauss–Seidel (PGS) methods often struggle with slow convergence or numerical instability in mass–spring–damper (MSD) systems characterized by high stiffness and damping. However, a *Temporal Gauss–Seidel* (TGS) solver augments position-based updates with *velocity-level* corrections at each iteration. This additional velocity damping better controls the rate of energy injection in the system, yielding improved stability and faster convergence for real-time applications.

In TGS, the constraint equations are formulated in velocity space. A generic velocity-based constraint has the form

$$\mathbf{J}\,\mathbf{v} + \mathbf{b} = 0, \tag{19}$$

where $\mathbf{J}$ is the constraint Jacobian, $\mathbf{v}$ is the vector of generalized velocities, and $\mathbf{b}$ includes bias terms for Baumgarte stabilization or error reduction. At each TGS iteration, incremental impulses $\Delta\boldsymbol{\lambda}$ are computed and used to update velocities:

$$\Delta\mathbf{v} = \mathbf{M}^{-1}\,\mathbf{J}^{\mathsf{T}}\,\Delta\boldsymbol{\lambda}, \tag{20}$$

where $\mathbf{M}^{-1}$ is the inverse mass-inertia matrix. The velocities and positions are then updated:

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \Delta\mathbf{v}, \tag{21}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta t\,\mathbf{v}^{(k+1)}, \tag{22}$$

where Δt is the timestep.

Compared to standard GS or PGS, TGS explicitly handles velocity-level corrections throughout each iteration. As a result, it damps out spurious high-frequency oscillations more effectively, allowing the solver to remain stable under large stiffness or damping constants. Moreover, TGS generally requires fewer iterations to converge to a physically plausible state in real-time contexts, reducing the computational burden while preserving numerical robustness.

Soft Constraints In mass–spring–damper (MSD) cable models, imposing extremely large stiffness and damping parameters can lead to unbounded force spikes if treated as rigid constraints. In NVIDIA PhysX, each joint constraint typically comprises a *driver*—which applies a proportional-derivative (PD) law to move the joint toward a target—and a *limit* that enforces a positional boundary. The TGS solver iterates over these

constraints at each timestep, updating velocities and positions so that the overall system converges to a physically plausible state.

Concretely, for a translational degree of freedom, the *driver* constraint is modeled by

$$F_{\text{drive}} = k_{\text{drive}}\left(p_{\text{target}} - p\right) + c_{\text{drive}}\left(v_{\text{target}} - v\right), \tag{23}$$

where p and v are the current position and velocity, p_{target} and v_{target} are the desired position and velocity, and k_{drive}, c_{drive} are the driver's stiffness and damping coefficients. The *limit* constraint remains inactive unless p exceeds a boundary $p_{\max}$. When that occurs, the solver applies

$$F_{\text{limit}} = -k_{\text{limit}}\left(p - p_{\max}\right) - c_{\text{limit}}\, v, \quad \text{for } p > p_{\max}, \tag{24}$$

to clamp motion at the boundary. During TGS iteration, these constraints are evaluated in parallel. If $p \le p_{\max}$, only the driver is active; if p attempts to exceed $p_{\max}$, the limit constraint supplements the driver to keep the segment within range.

When k_{drive} grows excessively large (e.g., on the order of 10^5 or more) without a corresponding limit stiffness, TGS can struggle to converge. From the solver's perspective, any elongation beyond $p_{\max}$ must be corrected solely by very large drive forces, leading to:

- No explicit boundary constraint to resist further elongation, effectively creating a "soft infinity" for $p > p_{\max}$.
- Large, rapidly changing velocities as TGS attempts to satisfy a single extreme-stiffness constraint.
- Ill-conditioned equations that amplify minor numerical errors, resulting in jitter or divergence.

Conversely, if a limit stiffness k_{limit} is configured, then once p exceeds $p_{\max}$, the solver imposes a stable spring–damper force (24) rather than relying on drive forces alone. Splitting the total load between the drive (maintaining its target) and the limit constraint (clamping elongation) greatly improves numerical conditioning and convergence.

By combining TGS with thoughtfully chosen driver parameters and an explicit limit constraint, one can achieve real-time MSD cable simulations that offer improved numerical stability, reduced oscillations, and physically consistent behavior.

3 Multilift Algorithm Implementation

3.1 Low-level Controller

A geometric controller[11] operates directly on the configuration space SO(3) to ensure robust and globally valid rotational error definitions. In the context of multirotor platforms, the position dynamics can be separated from the attitude dynamics by using geometric principles, while the yaw angle is decoupled to allow independent heading control[12]. This approach makes it possible to design a low-level controller that regulates both the thrust (for position tracking) and the body orientation (for attitude tracking) in a coherent framework.

Consider a multirotor with position $\mathbf{x} \in \mathbb{R}^3$ and orientation $R \in \mathrm{SO}(3)$. Let $\mathbf{x}_d$ be the desired position trajectory, and $\dot{\mathbf{x}}_d$, $\ddot{\mathbf{x}}_d$ be the desired velocity and acceleration. Let ψ_d be the desired yaw angle. We define the position and velocity errors as

$$\mathbf{e}_p = \mathbf{x} - \mathbf{x}_d, \quad \mathbf{e}_v = \dot{\mathbf{x}} - \dot{\mathbf{x}}_d. \tag{25}$$

A simple proportional-derivative-based desired acceleration command for position tracking can be formulated as

$$\mathbf{a}_d = \ddot{\mathbf{x}}_d - k_p\,\mathbf{e}_p - k_v\,\mathbf{e}_v + g\mathbf{e}_3, \tag{26}$$

where $\mathbf{e}_3 = [0 \;\; 0 \;\; 1]^\top$ represents the global vertical axis, g is the gravitational acceleration, and $k_p, k_v > 0$ are proportional and derivative gains.

The total thrust command f is then given by

$$f = m\|\mathbf{a}_d\|, \tag{27}$$

where m is the mass of the vehicle. To find the desired orientation R_d that orients the body z-axis toward $\mathbf{a}_d$ while simultaneously achieving the desired yaw ψ_d, we define

$$\hat{\mathbf{b}}_3 = \frac{\mathbf{a}_d}{\|\mathbf{a}_d\|}, \tag{28}$$

which is the desired body z-axis direction. The desired body x-axis $\hat{\mathbf{b}}_1$ is chosen to lie in the horizontal plane according to the desired yaw angle ψ_d. One way to obtain $\hat{\mathbf{b}}_1$ is

$$\hat{\mathbf{b}}_1^* = \begin{bmatrix} \cos(\psi_d) \\ \sin(\psi_d) \\ 0 \end{bmatrix}, \quad \hat{\mathbf{b}}_2 = \frac{\hat{\mathbf{b}}_3 \times \hat{\mathbf{b}}_1^*}{\|\hat{\mathbf{b}}_3 \times \hat{\mathbf{b}}_1^*\|}, \quad \hat{\mathbf{b}}_1 = \hat{\mathbf{b}}_2 \times \hat{\mathbf{b}}_3. \tag{29}$$

$$R_d = \begin{bmatrix} \hat{\mathbf{b}}_1 & \hat{\mathbf{b}}_2 & \hat{\mathbf{b}}_3 \end{bmatrix}. \tag{30}$$

This construction enforces the decoupling of yaw, because ψ_d is used explicitly to shape the desired heading, while the other two axes are defined by the thrust direction.

Attitude errors can be defined in a global, coordinate-free way using the fact that if $R_d \in \mathrm{SO}(3)$ is the desired orientation, then the orientation error e_R and angular velocity error e_Ω can be set as

$$e_R = \frac{1}{2}\left(R_d^T R - R^T R_d\right)^\vee, \quad e_\Omega = \Omega - R^T R_d \Omega_d, \tag{31}$$

where Ω is the body angular velocity, Ω_d is the desired angular velocity derived from the reference, and $(\cdot)^\vee$ maps a skew-symmetric matrix in $\mathfrak{so}(3)$ to $\mathbb{R}^3$. A typical feedback control law for the moment $\mathbf{M}$ is

$$\mathbf{M} = -k_R\, e_R - k_\Omega\, e_\Omega + \Omega \times J\,\Omega - J\left(\hat{\Omega} R^T R_d \Omega_d - R^T R_d\, \dot{\Omega}_d\right), \tag{32}$$

where $k_R, k_\Omega > 0$ are gains, J is the inertia matrix, and $\hat{\Omega}$ is the skew-symmetric matrix representation of Ω. This ensures that the actual orientation R follows R_d while allowing for an independent yaw specification.

In summary, the geometric controller with decoupled yaw control provides a unified way to track position and yaw independently by constructing the desired orientation matrix from the desired thrust direction and yaw angle. This approach leverages SO(3)-based error definitions to avoid singularities and discontinuities, yielding a smooth and globally valid low-level controller.

3.2 SITL System Design

Figure 5 depicts the software-in-the-loop (SITL) architecture that integrates the PX4 autopilot, Isaac Sim, ROS 2, and the Auto-Multilift algorithm into a cohesive simulation environment. Communication interface is established between the PX4 firmware and Isaac Sim through MAVLink, ensuring that simulated sensor data and flight commands are exchanged accurately in real time. The simulator provides high-fidelity sensor outputs, while PX4 responds with motor commands for attitude and position control.

On the ROS 2 side, several specialized nodes handle trajectory planning, high-level control, and data synchronization. The Geometric Control node processes state estimates from PX4 and generates reference attitudes and thrust commands to maintain stable flight. These references incorporate decoupled yaw regulation, thereby allowing the heading to be controlled independently of thrust and pitch–roll dynamics. Meanwhile, a separate Trajectory Planner node computes mission waypoints for multi-lift operations from the MPC Planner, where the Auto-Multilift method adjusted key parameters of MPC to improve tracking performance and robustness.

This SITL pipeline employs an additional time-synchronization mechanism to minimize latency and drift across multiple UAVs and ROS 2. A specialized node broadcasts clock updates, aligning measurement and control signals to a consistent temporal reference.

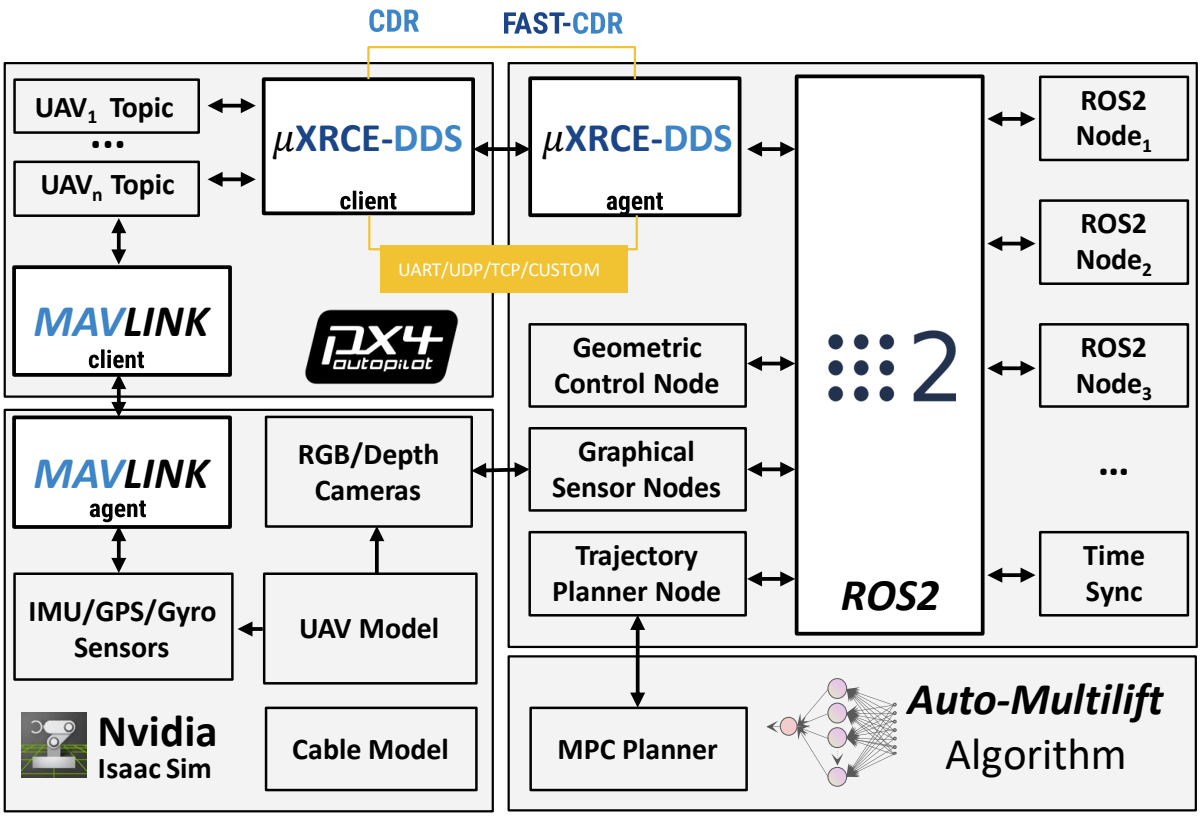

Figure 5: SITL framework for Auto-Multilift

Meanwhile, PX4 and Isaac Sim remain synchronized through a built-in lockstep procedure. By combining accurate multi-UAV load-transport simulations with flight-proven control software, the SITL setup provides a robust platform for validating the proposed methods before transitioning to physical flight tests.

4 Experiments & Results

4.1 Cable Dynamics Test

To verify that our cable simulation behaves as intended, we conducted a series of experiments in which the overall cable stiffness and damping were set to 10^5 N/m and 10^3 Ns/m, respectively. The primary goal of these experiments was to assess the accuracy of the cable's elasticity modeling and the real-time performance of the simulation. Throughout the experiments, the theoretical model of cables was considered as an ideal second-order system characterized by a stiffness of 10^5 N/m and a damping coefficient of 10^3 Ns/m along its extension. This theoretical model forms the basis for evaluating the cable's elongation accuracy.

We designed two main experiments. In the first experiment, weights ranging from 0.1 kg to 5 kg were suspended from a single rope and allowed to fall freely under gravity. Displacements, determined by the interplay of gravitational and tensile forces, were then measured. In the second experiment, a multi-rope configuration was employed, in which six ropes were arranged at angles 30° from the vertical. The displacement curves for the suspended weights were recorded and compared against theoretical predictions.

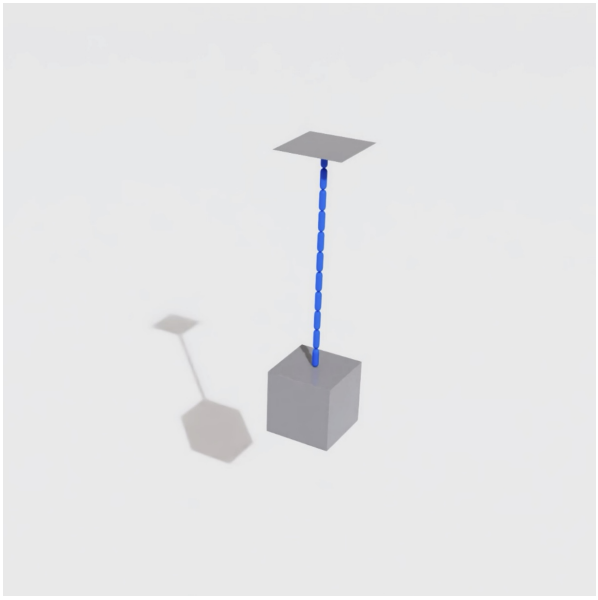

Figure 6: Single cable test scene

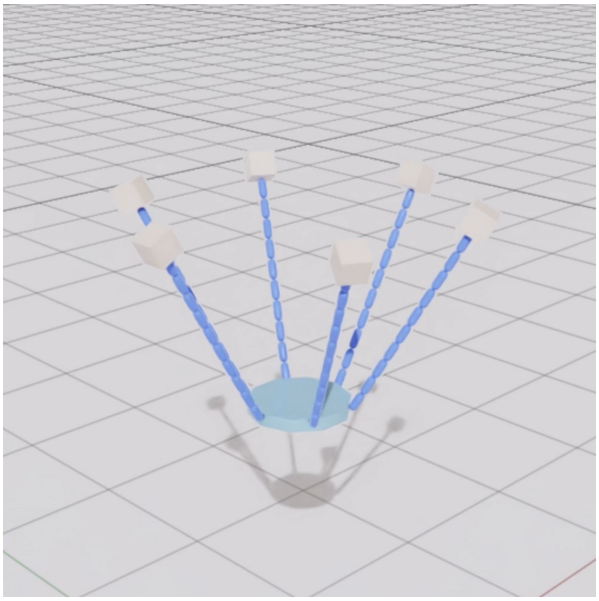

Figure 7: Six cable test scene

Table 1 summarizes the representative single-cable results. The tested stiffness values remain close to 10^5 N/m, with a maximum relative error of 2.2%. These data confirm that the simulation captures the cable's elasticity well, especially given the wide range of payloads tested. This also indicates that during the process of numerical iteration of the simulation, the damping force determined by the chosen damping coefficient, 10^3 Ns/m,

provides a sufficiently rapid settling time without causing a numerical instability.

Table 1: Results of the single-cable test for varying payload masses.

Payload (kg)	Tested Stiffness (N/m)	Relative Error (%)
0.1	1.000×10^5	0.0
0.5	1.000×10^5	0.0
1.0	9.995×10^4	0.5
5.0	9.978×10^4	2.2

Figures 8–11 illustrate how the elongations of single cables evolve under various mass of payloads. In each case, the rope elongation exhibits small oscillations around the expected theoretical second-order response, settling rapidly toward the predicted equilibrium. These minor deviations stem from numerical rounding in the real-time simulation but remain within acceptable margins. Overall, the tests validate that our model's stiffness and damping parameters yield physically consistent dynamics and maintain stability across different load conditions.

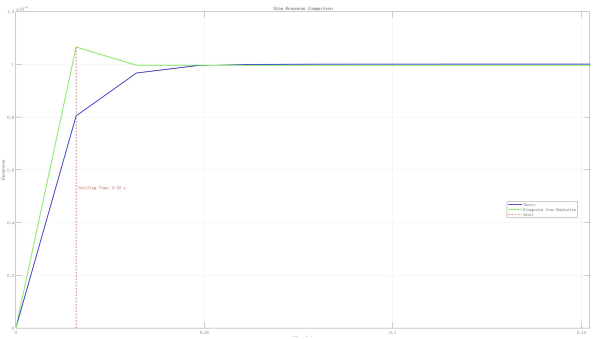

Figure 8: Single cable with 0.1 Kg payload

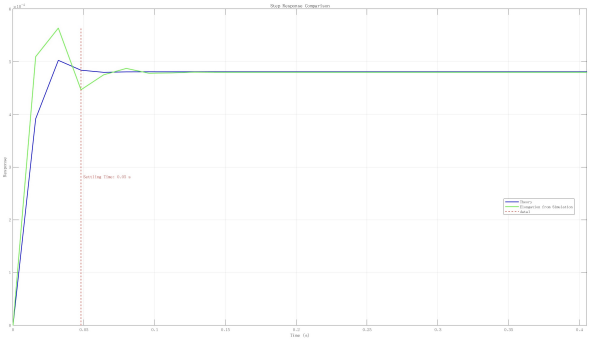

Figure 9: Single cable with 0.5 Kg payload

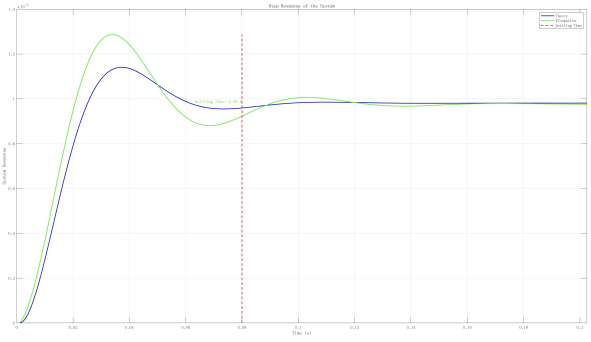

Figure 10: Single cable with 1 Kg payload

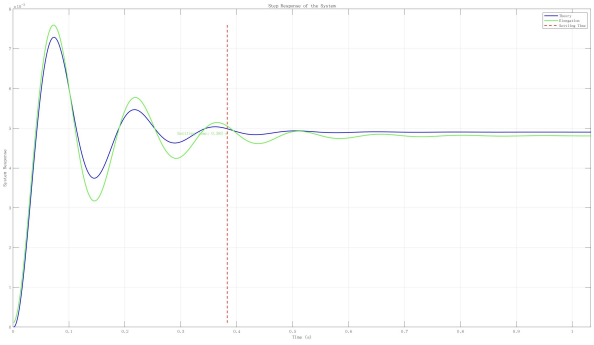

Figure 11: Single cable with 5 Kg payload

Figure 12-14 present the results of the six-cable experiments conducted with varying

payload masses.

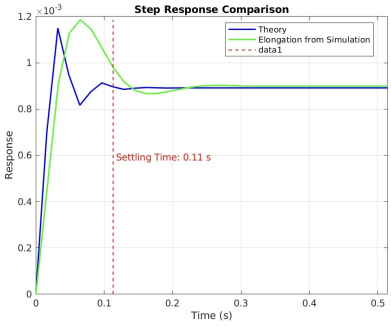

Figure 12: 6 cables with 6 Kg payload

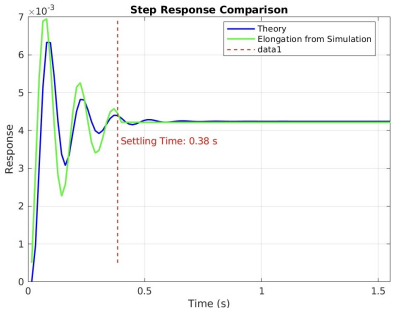

Figure 13: 6 cables with 30 Kg payload

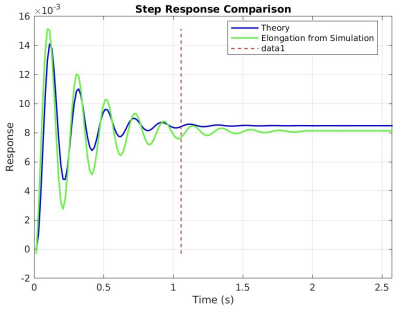

Figure 14: 6 cables with 60 Kg payload

As illustrated in Figures 12–14, the multi-cable system can be simulated in real time while maintaining excellent agreement with theoretical predictions across a wide range of payload masses. For lighter payloads (e.g., 6 kg), the system displays mild overshoot and rapid settling, consistent with classic second-order system behavior. Increasing the payload to 30 kg and 60 kg introduces more noticeable oscillations and longer settling times, which match the anticipated decrease in natural frequency for higher mass-to-stiffness ratios.

Despite these larger oscillations at higher loads, the simulated elongation remains centered around the theoretical curve, indicating both numerical stability and high physical accuracy. These findings confirm that the multi-cable system—designed with the specified stiffness and damping parameters—delivers real-time performance without compromising on fidelity, accurately capturing force distribution and dynamic effects even in challenging multi-rope scenarios.

4.2 Multilift Trajectory Following

To comprehensively assess the design of the SITL multi-lift system, we conducted a detailed trajectory-following experiment in which each quadrotor was commanded to travel along a horizontal circular path with a 5,m radius, at a constant angular velocity of 0.2,rad/s. The linear velocity for each quadrotor is therefore 2,m/s. This circular motion ensures that the system experiences continuous, varying orientations of tension and load distribution among the drones. Additionally, each drone's yaw angle was continuously adjusted so that its forward axis aligned tangentially to the circle, keeping the drone oriented in the direction of travel. By enforcing these conditions, the experiment provides a rigorous test of the geometric controller's ability to maintain both formation and accurate tracking. The experiment offers valuable insight into the overall system stability under dynamic load-sharing and highlights how effectively the geometric controller can execute coordinated maneuvers in real time.

The different states of the quadrotors during this experiment—takeoff, hover, and task execution—are shown in Figure 15-17. As illustrated in these figures, the system transitions smoothly from a static ground state into a fully airborne formation without significant overshoot or drift. During the *takeoff* phase (Figure 15), the quadrotors ascend while maintaining relative positions; cables remain slack and show its flexibility. In the *hover* phase (Figure 16), each vehicle maintains a stable altitude above the ground, allowing us to verify that the geometric controller can hold the formation in place with minimal oscillation. Finally, in the *task* phase (Figure 17), the quadrotors proceed along the circular path with precise velocity and orientation control, thus achieving the trajectory-following objective.

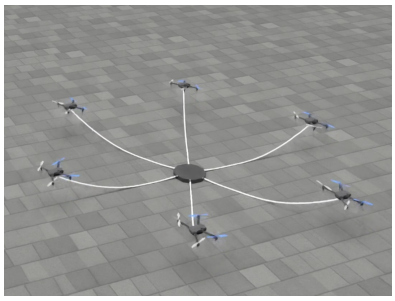

Figure 15: Takeoff

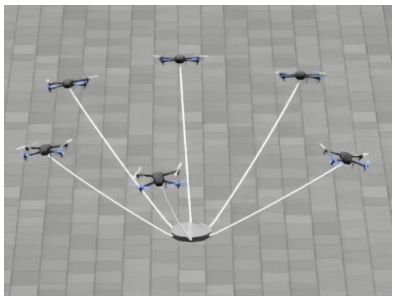

Figure 16: Hover

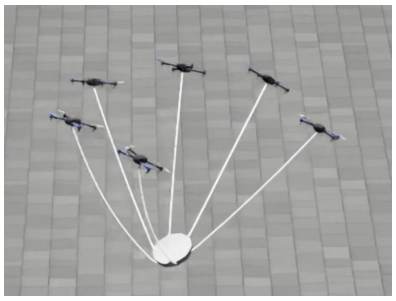

Figure 17: Task

The position and orientation tracking errors for individual quadrotors were also monitored throughout the experiment. The trajectory for one of the drones is visualized in Figure 18. Average position-tracking errors remained under 3 cm from the reference trajectory, underscoring the effectiveness of the control algorithm in regulating the system's translational dynamics. Meanwhile, the yaw angles consistently followed the intended tangential orientation, with deviations not exceeding 1°. These results demonstrate the capacity of the multi-lift system to maintain coherent, formation-level control, even under relatively high-speed maneuvers.

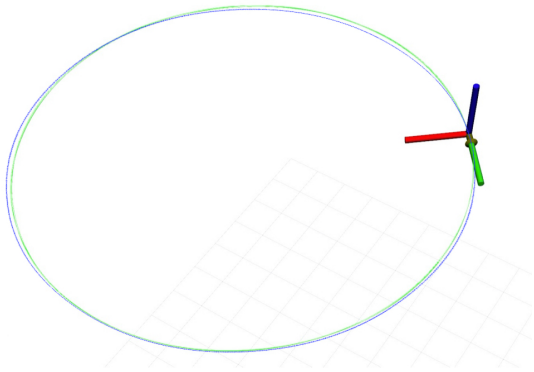

Figure 18: Trajectory tracking visualization in Rviz

Overall, the multilift trajectory-following experiment validates the system's ability to coordinate multiple quadrotors sharing a payload, while maintaining precise tracking of a dynamically path. The low position-tracking errors and stable orientation control collectively confirm that the designed system for multilift simulation is effective and robust for complex SITL cooperative transport tasks.

5 Conclusion

In this work, a high-fidelity simulation and control framework for multi-lift aerial transportation is developed and validated. The proposed system integrates Isaac Sim's TGS solver with a mass–spring–damper (MSD) cable model, ensuring stable real-time dynamics for highly stiff tethering. A robust SITL pipeline, combining PX4, ROS 2, and a geometric controller with decoupled yaw regulation, was demonstrated to achieve accurate multi-quadrotor trajectory tracking for cooperative load transport. Experimental results show that the simulator faithfully captures cable elasticity and damping behavior under a range of payloads and cable configurations, while sustaining minimal tracking error in dynamic maneuvers.

However, there remain several areas where further refinements are warranted. While wind-induced disturbances were modeled, aerodynamic interactions (e.g., rotor down wash, turbulent wakes) have not been comprehensively addressed. Future work will investigate these higher-order effects to reduce the sim-to-real gap and improve robustness in real-world deployments. Despite these limitations, the presented framework offers a solid foundation for distributed learning and control algorithms in multi-lift applications, with the flexibility to incorporate more advanced dynamics and hardware-in-the-loop testing moving forward.

References

[1] T. Lee, K. Sreenath, and V. Kumar, "Geometric control of cooperating multiple quadrotor uavs with a suspended payload," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 5510–5515. pages 7

[2] K. Klausen, C. Meissen, T. I. Fossen, M. Arcak, and T. A. Johansen, "Cooperative control for multirotors transporting an unknown suspended load under environmental disturbances," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 2, pp. 653–660, 2020. pages 7

[3] T. Lee, "Geometric control of multiple quadrotor uavs transporting a cable-suspended rigid body," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6155–6160. pages 7

[4] 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. pages 7

[5] 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. pages 7

[6] S. Sun, X. Wang, D. Sanalitro, A. Franchi, M. Tognon, and J. Alonso-Mora, "Agile and cooperative aerial manipulation of a cable-suspended load," 2025. [Online]. Available: https://arxiv.org/abs/2501.18802 pages 8

[7] Y. Wang, J. Wang, X. Zhou, T. Yang, C. Xu, and F. Gao, "Safe and agile transportation of cable-suspended payload via multiple aerial robots," 2025. [Online]. Available: https://arxiv.org/abs/2501.15272 pages 8

[8] B. Wang, R. Huang, and L. Zhao, "Auto-multilift: Distributed learning and control for cooperative load transportation with quadrotors," 2024. [Online]. Available: https://arxiv.org/abs/2406.04858 pages 8

[9] M. Jacinto, J. Pinto, J. Patrikar, J. Keller, R. Cunha, S. Scherer, and A. Pascoal, "Pegasus simulator: An isaac sim framework for multiple aerial vehicles simulation," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 917–922. pages 8

[10] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 1623–1629. pages 11

[11] 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. pages 17

[12] K. Gamagedara, M. Bisheban, E. Kaufman, and T. Lee, "Geometric controls of a quadrotor uav with decoupled yaw control," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 3285–3290. pages 17