High-Fidelity Integrated Aerial Platform Simulation for Control, Perception, and Learning

Jianrui Du, Kaidi Wang, Yingjun Fan, Ganghua Lai, *Graduate Student Member, IEEE*, and Yushu Yu

***Abstract*—This paper presents a simulator framework tailored Integrated Aerial Platforms (IAPs) using multiple quadrotors. Our framework prioritizes photo and contact fidelity, achieved through a modular design that balances rendering and dynamics computation. Key features include: i) support for diverse IAP configurations; ii) a customizable physics engine for realistic motion and contact simulation for aerial manipulation; and iii) Unreal Engine 5 for lifelike rendering, with sensor designs for visual-inertial SLAM positioning simulation. We showcase our framework's versatility through a range of scenarios, including trajectory tracking for both fully and under-actuated IAPs, peg-in-hole and direct wrench control tasks under external wrench influence, tightly-coupled SLAM positioning with physical constraints, and air docking task training and testing using offline-to-online reinforcement learning. Furthermore, we validate our simulator framework's fidelity by comparing results with real flight data for trajectory tracking and direct wrench control tasks. Our simulator framework promises to be valuable for developing and testing integrated aerial platform systems for aerial manipulation.**

***Note to Practitioners*—Motivated by the demand for effective simulation tools for Integrated Aerial Platforms (IAPs), this research addresses a significant gap in the availability of comprehensive simulation platforms designed to meet their unique challenges. This paper presents a high-fidelity simulation platform tailored specifically for IAPs, supporting a variety of configurations and capabilities. The platform not only generates high-fidelity image data and facilitates contact simulation but also serves as a vital resource for advancing perception, control, and learning for IAPs. By offering a robust simulation environment, this work aims to bridge the divide between theoretical research and practical applications, ultimately driving advancements in the field of aerial robotics.**

***Index Terms*—Simulation and animation, aerial systems: perception and autonomy, SLAM, deep learning in robotics and automation.**

I. Introduction

In RECENT years, the increasing utilization of drones has garnered considerable attention from researchers, particularly in areas such as aerial manipulation [1], [2], [3], aerial interaction with the environment [4], [5], [6], [7], aerial slung load [8], and active vision [9], [10], [11]. However, multi-rotor drones inherently present challenges as under-actuated systems, where attitude and position control are not decoupled [12], [13]. This complexity is further compounded when equipping drones with robotic arms, leading to insufficient flexibility in motion control. Consequently, such under-actuated systems are not ideal for aerial applications requiring active manipulation. For instance, the under-actuated configuration hinders the drone's ability to hover stably while performing simultaneous actions like grasping or placing objects, maintaining fine positioning during payload transportation, or conducting tasks that demand independent control of both attitude and position. To address this limitation, a novel concept of Integrated Aerial Platform (IAP) has been proposed, consisting of multiple small quadrotors as modules, connected by kinematic pairs [14], [15], [16]. By combining the forces and torques provided by multiple drones, this composite aerial platform can achieve up to six degrees of freedom (DOF) in motion and generate up to six-DOF independent forces and torques. Therefore, this composite flying robot can compensate for the shortcomings of a single drone as an operation platform, offering increased operational flexibility and greater force capacity. Compared with the single drone mounting a robotic arm, this composite robot can better realize the six-DOF compliant interaction with the environment.

Existing mainstream drone simulators, such as AirSim [17], typically model their simulation objects as single rigid bodies, without relative constraints and contact simulation with environmental objects. However, IAPs, different from regular UAVs, which have relative physical constraints among their modules, are designed for aerial operation tasks. Therefore, mainstream drone simulation platforms are not suitable for accurately simulating IAPs. In addition, robot simulators such as Gazebo [18] can offer motion simulation capabilities for IAPs, but their visual fidelity falls short for algorithm learning and verification purposes.

Therefore, we propose a new simulator framework, specially designed for high-fidelity data-driven simulation of IAPs in aerial operation. Unlike conventional drone simulation software, our framework is designed to handle heterogeneous aircraft composed of multiple rigid bodies with structural connections. Our simulation accurately captures the kinematic and dynamic constraints among the IAP components, allowing for a realistic representation of their interactions. Our modular software architecture separates rendering and physics modules, facilitating accurate collision and contact simulations essential for aerial manipulation tasks. The physics module takes into

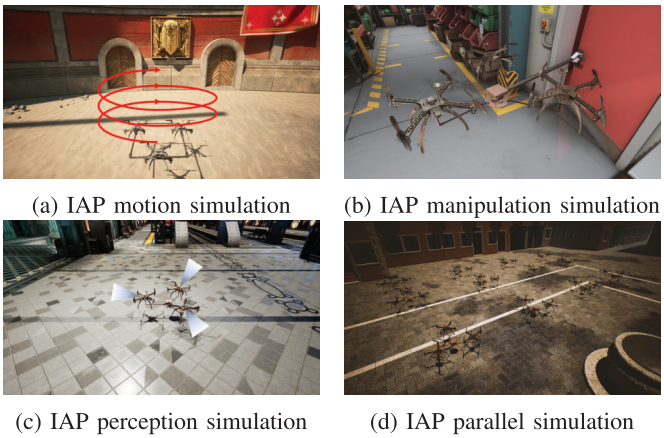
Received 22 October 2024; revised 14 February 2025; accepted 22 March 2025. Date of publication 26 March 2025; date of current version 18 April 2025. This article was recommended for publication by Associate Editor Z. Liu and Editor J. Yi upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62173037, in part by the National Key Research and Development Program of China, in part by the State Key Laboratory of Robotics and Systems Harbin Institute of Technology (HIT), and in part by Beijing Institute of Technology Research Fund Program for Young Scholars. *(Corresponding author: Yushu Yu.)*

The authors are with the School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: yushu.yu@bit.edu.cn).

Data is available on-line at https://github.com/BIT-aerial-robotics/IapSimulatorROS

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2025.3555014, provided by the authors.

Digital Object Identifier 10.1109/TASE.2025.3555014



(a) IAP motion simulation (b) IAP manipulation simulation

(c) IAP perception simulation (d) IAP parallel simulation

Fig. 1. Typical scenarios for IAP simulation.

account collisions and contacts not only between aircraft and the environment but also among aircraft themselves, making it suitable for simulating aerial manipulations. Through robust data communication and synchronization mechanisms, each module can run at different frequencies while ensuring temporal consistency and low latency, thereby meeting the real-time requirements of simultaneous localization and mapping (SLAM) simulations. The IAP model is developed using C++ and UE Blueprints (Blueprints for rendering only), allowing for parallel simulation of multiple IAPs within a single environment and parallel simulation across multiple environments, which can be used for data collection and training machine learning algorithms. The typical scenarios for IAP simulation are illustrated in Fig. 1.

We conduct four distinct sets of experiments to validate the efficacy of our simulator framework. In the first set of experiments, to validate the feasibility of our simulator for simulating the baseline motion of IAPs, we verify the trajectory tracking performance of various IAP configurations connected by spherical joints, including the fully-actuated three-star IAP and three-square IAP with six-dimensional motion, the under-actuated two-line IAP with three-dimensional spatial motion, and the three-line IAP with five-dimensional independent motion. Moreover, we compare the results of the fully-actuated configurations with real flight data obtained from our prototype. In the second set of experiments, to confirm the simulator's effectiveness in simulating contact tasks for IAPs, we conduct a peg-in-hole experiment under impedance control and a simulation experiment under direct wrench control for the three-star IAP. Furthermore, we compare the results of the direct wrench control simulation with those from real flight experiments. In the third set of experiments, to validate the synchronization design of multiple sensors and real-time data processing in the simulator, we perform tightly coupled SLAM positioning simulations for the three-star IAP with multiple trajectories. Lastly, in the fourth set of experiments, to assess the data collection and machine learning algorithm training capabilities of our framework, we employ an offline-to-online reinforcement learning method to conduct training and verification for the air docking task of two quadrotors.

Our paper presents the following main contributions:

- We have designed a simulator framework supporting customizable configurations of Integrated Aerial Platforms (IAPs) with physical constraints, accommodating various configurations: IAPs with varying numbers of sub-aircraft, IAPs with different configurations of spherical joint connections, and fixed-link IAPs.
- The simulator framework we design supports IAP control simulations involving interactions with the environment and among different aircraft, including external wrench contact, thus demonstrating a realistic contact effect.
- The tightly coupled localization algorithm that fuses odometry data with physical constraints for the IAP is validated. Leveraging the capabilities of the latest game engine, Unreal Engine 5 [19], our simulator can integrate highly realistic three-dimensional environments with the design of sensors such as IMU and UWB. This integration facilitates the generation of highly realistic images and sensor data for purposes such as visual perception verification for tightly coupled SLAM localization under physical constraints.
- We have employed our simulator framework to conduct a series of simulation tests covering various aspects of IAPs, including motion control, interaction control, tightly coupled SLAM positioning, and offline-to-online reinforcement learning training and validation. Additionally, we have compared the fidelity of our simulator framework in motion control and direct wrench control, against real flight experiments involving two three-sub-aircraft IAP configurations. To the best of the authors' knowledge, this is the first simulator that supports motion control, interaction control, visual-based odometry, and reinforcement learning for heterogeneous aerial systems.

The paper is structured as follows: Section II provides a detailed review of recent interaction control methods applied to fully-actuated aerial manipulators, as well as an overview of popular physics engines and simulators of multi-jointed robots and data-driven simulators of drones. Section III presents an overview of our proposed IAP system and its dynamics modeling. In Section IV, we elaborate on the theoretical background of various interaction control, perception, and learning algorithms used in our IAP systems. Following this, Section V introduces the architecture of our simulator framework and offers insights into the design of its key modules. In Section VI, we demonstrate the testing results of our proposed IAP simulator framework through the design of various experiments to establish its feasibility and fidelity. Finally, Section VII concludes the paper.

II. Related Work

A. Interaction Control of Fully Actuated Aerial Manipulators

In the standard multirotor configuration, multirotor UAVs are typically limited to four-DOF actuation due to the parallel orientation of all propeller force vectors. Consequently, to better realize omnidirectional force/torque operation in the air, researchers have designed various 6-DOF omnidirectional

fully-actuated aerial vehicles. Franchi et al. designed a hexarotor aerial vehicle with fixed tilted propellers [20], enabling independent control of its position and orientation in free space without the need for additional hardware. This design allows the aircraft to apply force to the environment to resist force and torque in any direction, performing aerial manipulation tasks. They demonstrated the maneuverability of the aerial vehicle in tasks such as peg-in-hole and sliding along a wall [21]. Siegwart et al. studied the contact-based inspection tasks with a similar hexarotor with tilted propellers [22], [23]. Voyles et al. explored an inspection-on-the-fly method for aerial manipulators using hybrid physical interaction control [24]. Moreover, Lee et al. proposed the OmniDirectional Aerial Robot (ODAR), which boasts a stronger payload capacity and realizes hybrid pose/wrench control with a downward force of 60 N, significantly exceeding its weight of 2.6 kg [25].

Fully-actuated aerial vehicles with fixed rotor angles can achieve basic omnidirectional force/torque operation in the air. However, due to the inherent cancellation of part of the propeller thrust, they are limited in their actuation force/torque and endurance, resulting in low energy efficiency. As a solution, researchers have devised fully actuated aerial vehicles with dynamically variable actuator angles. Kovac et al. introduced a morphing TiltDrone composed of several actuated joints used to tilt the rotors [26]. Franchi et al. designed FAST-Hex [27], a novel UAV concept capable of smoothly transitioning its configuration from under-actuated to fully actuated using only one additional motor that tilts all propellers simultaneously. This design enables the aircraft to switch between the more energy-efficient under-actuated flight and the less efficient but fully-actuated flight mode based on task requirements, thereby enhancing the efficiency of omnidirectional force/torque production.

However, the inclusion of additional motors that drive propeller angles increases the mechanical complexity of the aerial vehicle, without enhancing its operational force/torque output, thereby limiting its practical application. Therefore, some researchers have explored the utilization of off-the-shelf ready-made small multi-rotor UAVs as actuation modules, connecting multiple UAVs with passive joints to form fully actuated aerial vehicles, suitable for various aerial manipulation applications. Lee et al. designed a robotic platform for aerial manipulation using quadrotors as rotating thrust generators and studied control and allocation problems under both fully-actuated and under-actuated configurations [14]. Su et al. investigated fault-tolerant control of a tiltable-rotor aerial platform comprising quadrotors and passive hinges [28], [29], and downwash-aware control allocation for over-actuated UAV platforms [30]. Nevertheless, these studies did not demonstrate omnidirectional aerial manipulation. In terms of aerial manipulation capabilities, Zhao et al. designed versatile articulated aerial robot DRAGON [31], [32], and Su et al. studied sequential manipulation planning for over-actuated unmanned aerial manipulators [33]. However, in terms of positioning, these aerial vehicles heavily rely on external positioning methods, such as motion capture systems, rendering them unsuitable for operating in complex indoor scenarios and severely limiting their practical applications. Additionally, before real-world testing, they could only perform simple numerical simulations of dynamics, resulting in limited fidelity in simulating contact dynamics. Moreover, they could not generate virtual environments for visual information, thus hindering the simulation validation of algorithms reliant on visual input, further constraining the applicability of their aerial vehicles and failing to uncover the full potential applications of such fully actuated flight platforms.

B. Physics Engines for Robot Motion and Contact Simulation

IAP is a multi-jointed robot for aerial manipulation, underscoring the significance of dynamics computation in simulation. A physics engine is a vital software tool that simulates the dynamics of robot motion and interaction with the environment, playing a crucial role in robot learning and control. Gazebo [18] and MuJoCo [34] are two popular lightweight open-source robot physics engines, that excel in modeling complex joints and constraints, and provide C/C++ interfaces. Also, Gazebo seamlessly integrates with ROS (Robot Operating System). On the other hand, PyBullet [35] is a robot physics engine based on the Bullet engine, offering a user-friendly Python interface, but its simulation accuracy and stability are inferior to Gazebo and MuJoCo [36]. RaiSim [37] is an emerging commercial physics engine, that outperforms the previous physics engines in terms of efficiency and accuracy of contact simulation, but it is not open-source and lacks a ROS interface, which hinders its integration with ROS-based algorithms for development and testing. Isaac Sim [38], an open-source robot simulator based on NVIDIA Omniverse, leverages NVIDIA's most recent technology like RTX and PhysX to achieve high-quality rendering and physics simulation. Isaac Sim's advantages include impressive visual effects and visual sensor quality, support for various robots and scenarios, and provision of robot learning, navigation, and manipulation functions, as well as its compatibility with ROS and Isaac SDK. However, Isaac Sim imposes high computational resource requirements, requiring the use of NVIDIA's GPU, which makes it unsuitable for deployment on computers without NVIDIA hardware.

Existing physics engines are not inherently tailored for aerial robots. Efforts have been made to apply physics engines to multirotors, such as RotorS [39] based on Gazebo, quadrotor simulation [40] based on RaiSim, and Pegasus Simulator [41] based on Isaac. However, these engines do not address the challenges of simulating multi-jointed aerial robots for aerial manipulation. Such simulations require not only accurate modeling of motion and contact dynamics but also additional development and adaptation of robot configuration and program interface to meet the specific needs of aerial manipulation tasks.

C. Data-Driven Rotorcraft Flight Simulators

Data-driven flight simulators play a crucial role in the development and validation of flight algorithms, and can also accelerate the training of machine learning algorithms with a large amount of realistic simulation data [42], [43].

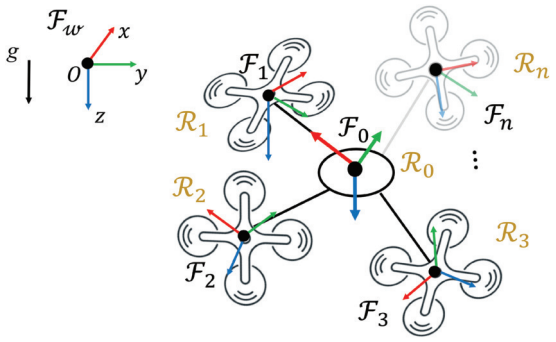


Fig. 2. Coordinate system description of the IAP.

AirSim [17] is an open-source multi-rotor simulation software developed by Microsoft based on Unreal Engine 4 (UE4), which provides a rich multi-modal sensor suite, offering high-quality visual effects and visual sensor data, while also supporting ROS. However, the physical accuracy and stability of AirSim simulations need improving, particularly in complex environments and dynamic conditions, where phenomena that do not conform to physical laws may occur. Sim4CV [44] is a simulator built on top of the Unreal Engine, integrating full-featured physics-based cars, UAVs, and animated human actors in diverse urban and suburban 3D environments. Flightmare [45] is an open-source multi-rotor simulation software developed by Robotics and Perception Group, based on Unity 3D [46], which decouples the physics engine and the rendering engine, and designs an API for reinforcement learning capable of simulating hundreds of quadrotors in parallel. FlightGoggles [47] is an open-source multi-rotor simulation software developed by LIDS, MIT, based on Unity 3D, focusing on visual effects and considering the aerodynamics of the aircraft. RFlySim [48] is a multi-rotor simulation software based on UE4, which adopts a model-based design approach, and can be used for multi-rotor control and safety testing, but it is a commercial software.

Existing data-driven flight simulation software does not typically handle complex multi-joint aircraft as simulation objects, and does not consider contact simulation relevant to aerial manipulation applications. Furthermore, these simulators lack essential sensor design and synchronization of multiple drones' images and sensor timestamps, rendering them unsuitable for testing and validating coupled SLAM localization algorithms for numerous UAVs.

III. Integrated Aerial Platform Overview

A. Platform Overview

We consider an IAP with N ($N = 2, 3, \ldots, n$) quadrotors attached to the central platform via spherical joints, as illustrated in Fig. 2. Each sub-aircraft, along with the central platform, is treated as a rigid body, with the central platform denoted as $\mathcal{R}_0$ and the sub-aircraft as $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$, respectively. The inertial frame is denoted by $\mathcal{F}_w$, a right-handed NED coordinate system where the x-axis points North, the y-axis points East and the z-axis points downward. The frame attached to $\mathcal{R}_n$ is denoted by $\mathcal{F}_n$, a body-fixed right-handed FRD coordinate system, where the x-axis, y-axis, and z-axis point forward, rightward, and downward of the rigid body, respectively.

Assuming that the center of mass (CoM) of each sub-aircraft coincides with its spherical joint position, we use $\boldsymbol{d}_i \in \mathbb{R}^3$ ($i = 1, 2, \ldots, n$) to denote the position of the CoM of the i-th sub-aircraft $\mathcal{R}_i$ in $\mathcal{F}_0$, and $\boldsymbol{p}_i^w \in \mathbb{R}^3$ ($j = 0, 1, \ldots, n$) to denote the position of the CoM of $\mathcal{R}_i$ in $\mathcal{F}_w$. We then have the following relation between the CoM of the i-th sub-aircraft and the CoM of the central platform, all expressed in $\mathcal{F}_w$, as

$$\boldsymbol{p}_i^w = \boldsymbol{p}_0^w + \boldsymbol{R}_0 \boldsymbol{d}_i, \ i = 1, 2, \ldots, n \tag{1}$$

where $\boldsymbol{R}_i \in \mathrm{SO}(3)$ ($i = 0, 1, \ldots, n$) is the attitude matrix of $\mathcal{R}_i$ expressed in $\mathcal{F}_w$.

Remark 1: Theoretically, our simulation supports IAPs with different types of joints and combinations of multiple types of joints. However, the modeling, control, and verification of IAPs that introduce other types of joints in addition to spherical joints are completely new issues that go beyond the scope of this paper. Therefore, only IAPs with spherical joints are modeled and tested.

B. Dynamics Modeling

We conduct a force analysis on the entire IAP, where each sub-aircraft can be regarded as providing a rotating thrust $\boldsymbol{\Lambda}_i \in \mathbb{R}^3$ to the central platform through the spherical joint. The resulting force in the center comprises three parts: thrusts, gravity, and external forces. The translational dynamics equation of the IAP expressed in $\mathcal{F}_0$ can then be derived as

$$m\dot{\boldsymbol{v}}_0 + (m\boldsymbol{\omega}_0)^\wedge \boldsymbol{v}_0 = \sum_{i=1}^{n} \boldsymbol{\Lambda}_i + mg\boldsymbol{R}_0^{\mathrm{T}} \boldsymbol{e}_3 + \boldsymbol{f}_e \tag{2}$$

where $\boldsymbol{v}_0, \boldsymbol{\omega}_0 \in \mathbb{R}^3$ represent the linear velocity and the angular velocity of the central platform expressed in $\mathcal{F}_0$, $\boldsymbol{\Lambda}_i$ denotes the thrust of the i-th sub-aircraft, $\boldsymbol{f}_e \in \mathbb{R}^3$ indicates the external force acting on the CoM of the central platform, m stands for the total mass of the central platform and sub-aircraft, $g = 9.81$ m/s represents the gravitational acceleration, the mapping $(\times)^\wedge : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ satisfies $(\boldsymbol{\omega})^\wedge \boldsymbol{v} := \boldsymbol{\omega} \times \boldsymbol{v}, \forall \boldsymbol{\omega}, \boldsymbol{v} \in \mathbb{R}^3$, and $\boldsymbol{e}_3 := [0, 0, 1]^{\mathrm{T}}$ denotes the unit vector of the z-axis.

Assuming that the rotation of the sub-aircraft has a negligible influence on the overall inertia tensor of the IAP $\boldsymbol{J} \in \mathbb{R}^{3\times 3}$ and that $\boldsymbol{J}$ remains constant during the motion of the IAP, the resultant torque on the center is also composed of three parts: the thrust torque, gravity torque, and external torque. Therefore, the rotational dynamics equation of the IAP expressed in $\mathcal{F}_0$ can be derived as

$$\boldsymbol{J}\dot{\boldsymbol{\omega}}_0 - (\boldsymbol{J}\boldsymbol{\omega}_0)^\wedge \boldsymbol{\omega}_0 = \sum_{i=1}^{n} (\boldsymbol{d}_i)^\wedge \boldsymbol{\Lambda}_i + (\boldsymbol{d}_0)^\wedge mg\boldsymbol{R}_0^{\mathrm{T}} \boldsymbol{e}_3 + \boldsymbol{\tau}_e \tag{3}$$

where $\boldsymbol{\tau}_e \in \mathbb{R}^3$ represents the external torque acting on $\mathcal{R}_0$, and $\boldsymbol{d}_0 \in \mathbb{R}^3$ denotes the position of the CoM of the IAP expressed in $\mathcal{F}_0$, as the weighted average of the CoM positions of the sub-aircraft and the central platform:

$$\boldsymbol{d}_0 = \frac{m_0 \boldsymbol{d}_c + \sum_{i=1}^{n} m_i \boldsymbol{d}_i}{\sum_{i=0}^{n} m_i}$$

where m_i and $\boldsymbol{d}_i$ (i =1,2,...,n) represent the mass and CoM position of the i-th sub-aircraft, respectively, and m_0, $\boldsymbol{d}_c$ correspond to the mass and CoM position of the central platform, and $\boldsymbol{J}$ as

$$\boldsymbol{J} = \boldsymbol{J}_0 - \sum_{i=1}^{n} m_i(\boldsymbol{d}_i^{\wedge})^2$$

where $\boldsymbol{J}_0$ is the inertia tensor of the central platform.

Then, the more concise 6-DOF dynamics of the IAP can be expressed as [14]

$$\boldsymbol{M}\dot{\boldsymbol{\xi}} + \boldsymbol{C}\boldsymbol{\xi} = \boldsymbol{U} + \boldsymbol{G} + \boldsymbol{F}_e \tag{4}$$

where $\boldsymbol{\xi} := [\boldsymbol{v}_0; \boldsymbol{\omega}_0]$, and

$$\boldsymbol{M} := \begin{bmatrix} m\boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{J} \end{bmatrix}$$
$$\boldsymbol{C} := \begin{bmatrix} (m\boldsymbol{\omega}_0)^{\wedge} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -(\boldsymbol{J}\boldsymbol{\omega}_0)^{\wedge} \end{bmatrix}$$
$$\boldsymbol{G} := \begin{bmatrix} mg\boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{e}_3 \\ (\boldsymbol{d}_0)^{\wedge}mg\boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{e}_3 \end{bmatrix}$$
$$\boldsymbol{F}_e := \begin{bmatrix} \boldsymbol{f}_e \\ \boldsymbol{\tau}_e \end{bmatrix}$$

are the lumped (symmetric/positive-definite) inertia matrix, the (skew-symmetric) Coriolis matrix, the gravity effect, and the external forcing, respectively. Here, $\boldsymbol{I}_3$ is the 3-dimensional identity matrix, $\boldsymbol{0}_{3\times3}$ is the 3-dimensional zero matrix, and

$$\boldsymbol{U} := \boldsymbol{B}\boldsymbol{\Lambda} \tag{5}$$

is the resultant active wrench generated by the sub-aircraft and acting on the center, where $\boldsymbol{\Lambda} := [\boldsymbol{\Lambda}_1; \boldsymbol{\Lambda}_2; \cdots; \boldsymbol{\Lambda}_n] \in \mathbb{R}^{3n}$ is the collective thrust vector, and

$$\boldsymbol{B} := \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{I}_3 & \dots & \boldsymbol{I}_3 \\ (\boldsymbol{d}_1)^{\wedge} & (\boldsymbol{d}_2)^{\wedge} & \dots & (\boldsymbol{d}_n)^{\wedge} \end{bmatrix} \tag{6}$$

is the mapping matrix reflecting how the thrust actuation of each sub-aircraft affects the IAP dynamics based on its mechanical structure design.

In addition, assuming smooth spherical joints and neglecting joint rotational resistance torque, the rotational dynamics equation of each sub-aircraft is given by

$$\boldsymbol{J}_i\dot{\boldsymbol{\omega}}_i - (\boldsymbol{J}_i\boldsymbol{\omega}_i)^{\wedge}\boldsymbol{\omega}_i = \boldsymbol{\tau}_i \tag{7}$$

Eqn. (4) and (7) together constitute the complete (6+3n)-DOF dynamics of the IAP.

IV. Interaction Control, Perception, and Learning Algorithms of Aerial Vehicles Under Physical Interaction

A. Trajectory Tracking Controller

The IAP system can be separated into a slow-varying subsystem and the fast-varying subsystems. The slow-varying subsystem represents the entire IAP system, while the fast-varying subsystems are the individual flight systems of each sub-aircraft. Hence, prioritization is given to designing the controller for the slow-varying subsystem, serving as the outer loop of the IAP controller, to generate overall expected wrench commands for the system, and subsequently, decompose the overall commands into the commands for each sub-aircraft. The quadrotor controllers then serve as the inner loop of the IAP controller, executing the decomposed thrust and attitude commands.

In Eqn. (4), $\boldsymbol{U}$ represents the expected wrench output of the central platform and serves as input for the allocation stage. Eqn. (5) reveals that when the matrix $\boldsymbol{B}$ achieves full rank, that is, rank($\boldsymbol{B}$) = 6, $\boldsymbol{U} \in \mathbb{R}^6$, indicating the IAP as a fully-actuated platform with 6 degrees of freedom. Conversely, if $\boldsymbol{B}$ fails to achieve full rank, the IAP becomes an under-actuated platform. Eqn. (6) reveals that if $\boldsymbol{d}_i$ satisfies

$$\boldsymbol{d}_i - \boldsymbol{d}_1 = t_i(\boldsymbol{d}_2 - \boldsymbol{d}_1),\ t_i \in \mathbb{R},\ i = 1, 2, \dots, n \tag{8}$$

where all spherical joints are collinear, then rank($\boldsymbol{B}$) = 5, indicating that the IAP is under-actuated. Conversely, if Eqn. (8) is not satisfied, the IAP is fully-actuated.

The objective of the motion controller for the fully actuated IAP is to achieve the goal expressed as:

$$(\boldsymbol{p}_0^w(t), \boldsymbol{R}_0(t)) \to (\bar{\boldsymbol{p}}_0^w(t), \bar{\boldsymbol{R}}_0(t)) \tag{9}$$

This objective entails designing $\boldsymbol{U} \in \mathbb{R}^6$ to guide the actual state trajectory of the IAP, $(\boldsymbol{p}_0^w(t), \boldsymbol{R}_0(t)) \in \mathrm{SE}(3)$, to converge toward the desired pose trajectory, $(\bar{\boldsymbol{p}}_0^w(t), \bar{\boldsymbol{R}}_0(t)) \in \mathrm{SE}(3)$. Here, $\bar{\boldsymbol{p}}_0^w \in \mathbb{R}^3$ and $\bar{\boldsymbol{R}}_0 \in \mathrm{SO}(3)$ represent the nominal pose trajectory expressed in $\mathcal{F}_w$.

According to Lee et al. [14], a controller can be designed using the Lyapunov method, expressed as:

$$\boldsymbol{U} := \boldsymbol{M}\dot{\bar{\boldsymbol{\xi}}} + \boldsymbol{C}\bar{\boldsymbol{\xi}} - \boldsymbol{k}^{\mathrm{P}}\boldsymbol{e}_T - \boldsymbol{k}^{\mathrm{I}}\int \boldsymbol{e}_T \mathrm{d}t - \boldsymbol{k}^{\mathrm{D}}\boldsymbol{e}_{\xi} - \boldsymbol{G} - \boldsymbol{F}_e \tag{10}$$

Here, $\boldsymbol{k}^{\mathrm{P}}, \boldsymbol{k}^{\mathrm{I}}, \boldsymbol{k}^{\mathrm{D}}$ are 6-dimensional non-negative definite diagonal matrices. $\bar{\boldsymbol{\xi}} \in \mathbb{R}^6$ represents the nominal speed of $\boldsymbol{\xi}$ in Eqn. (4) expressed in $\mathcal{F}_0$, $\boldsymbol{e}_{\xi} := \boldsymbol{\xi} - \bar{\boldsymbol{\xi}} \in \mathbb{R}^6$ denotes the linear and angular velocity error of the central platform, and $\boldsymbol{e}_T \in \mathbb{R}^6$ signifies the pose error of the platform. Specifically,

$$\bar{\boldsymbol{\xi}} := \begin{bmatrix} \boldsymbol{R}_0^{\mathrm{T}}\bar{\boldsymbol{R}}_0\bar{\boldsymbol{v}}_0 \\ \boldsymbol{R}_0^{\mathrm{T}}\bar{\boldsymbol{R}}_0\bar{\boldsymbol{\omega}}_0 \end{bmatrix} \in \mathbb{R}^6$$
$$\boldsymbol{e}_T := \begin{bmatrix} \boldsymbol{R}_0^{\mathrm{T}}(\boldsymbol{p}_0^w - \bar{\boldsymbol{p}}_0^w) \\ \frac{1}{2}(\bar{\boldsymbol{R}}_0^{\mathrm{T}}\boldsymbol{R}_0 - \boldsymbol{R}_0^{\mathrm{T}}\bar{\boldsymbol{R}}_0)^{\vee} \end{bmatrix} \in \mathbb{R}^6$$

where $\bar{\boldsymbol{v}}_0 = \bar{\boldsymbol{R}}_0^{\mathrm{T}}\dot{\bar{\boldsymbol{p}}}_0^w, \bar{\boldsymbol{\omega}}_0 = (\bar{\boldsymbol{R}}_0^{\mathrm{T}}\dot{\bar{\boldsymbol{R}}}_0)^{\vee} \in \mathbb{R}^3$ represent the nominal linear and angular velocity expressed in $\mathcal{F}_0$, and $(\times)^{\vee} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is the inverse mapping of $(\times)^{\wedge}$.

According to the stability analysis using the Lyapunov method in [14], it has been established that when $\boldsymbol{k}^{\mathrm{I}} = \boldsymbol{0}_{6\times6}$, the controller achieves almost globally asymptotically stability. However, the inclusion of the integral term affects global stability. Nevertheless, the integral term is deemed necessary for enhanced anti-interference performance but should be tuned to a smaller value to minimize its potential destabilizing effect on the system.

When the spherical joints of the sub-aircraft are all collinear, satisfying Eqn. (8), the IAP becomes an under-actuated system. Consequently, the control objective shifts from Eqn. (9) to

$$(\boldsymbol{p}_0^w(t), \boldsymbol{r}_0^w(t)) \to (\bar{\boldsymbol{p}}_0^w(t), \bar{\boldsymbol{r}}_0^w(t)) \tag{11}$$

Here, $(\boldsymbol{p}_0^w(t), \boldsymbol{r}_0^w(t)) \in \mathbb{R}^3 \times \mathrm{S}^2$, where $\boldsymbol{r}_0^w = \boldsymbol{R}_0 \boldsymbol{r}_0^0 \in \mathrm{S}^2$ represents the unit vector along the under-actuated IAP's x-axis in $\mathcal{F}_w$. Here, $\boldsymbol{r}_0^0 = \boldsymbol{e}_1 := [1,0,0]^{\mathrm{T}}$ signifies the unit vector along the x-axis expressed in $\mathcal{F}_0$.

Following [14], [49], the configuration $\boldsymbol{q} := (\boldsymbol{p}_0^w, \boldsymbol{R}_0)$ of the IAP evolves on a 6-dimensional manifold $\mathcal{M}$, with the velocity $\boldsymbol{\xi} := \dot{\boldsymbol{q}} \in T_{\boldsymbol{q}}\mathcal{M}$ and forces $\boldsymbol{U}, \boldsymbol{G}, \boldsymbol{F}_e \in T_{\boldsymbol{q}}^*\mathcal{M}$. Here, $T_{\boldsymbol{q}}\mathcal{M}$ and $T_{\boldsymbol{q}}^*\mathcal{M}$ represent the tangent and cotangent spaces at $\boldsymbol{q} \in \mathcal{M}$, respectively, defined as

$$T_{\boldsymbol{q}}\mathcal{M} = \boldsymbol{\Delta}_a \oplus \boldsymbol{\Delta}_u, \quad T_{\boldsymbol{q}}^*\mathcal{M} = \boldsymbol{\Omega}_a \oplus \boldsymbol{\Omega}_u$$

where the dynamic model is divided into two components: fully actuated and under-actuated, where the subscript a denotes the fully actuated part and the subscript u represents the under-actuated part. Alternatively, in coordinates, this can be expressed as

$$\boldsymbol{\xi} = \boldsymbol{\Delta} \begin{bmatrix} \boldsymbol{\nu}_a \\ \nu_u \end{bmatrix} \tag{12}$$

$$\boldsymbol{U} = \boldsymbol{\Omega}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{u}_a \\ u_u \end{bmatrix}, \ \boldsymbol{G} = \boldsymbol{\Omega}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{g}_a \\ g_u \end{bmatrix}, \ \boldsymbol{F}_e = \boldsymbol{\Omega}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{f}_a \\ f_u \end{bmatrix} \tag{13}$$

Then, we can rewrite Eqn. (4) as

$$\boldsymbol{M}_a \dot{\boldsymbol{\nu}}_a + \boldsymbol{C}_a \boldsymbol{\nu}_a + \boldsymbol{C}_{au} \nu_u = \boldsymbol{u}_a + \boldsymbol{g}_a + \boldsymbol{f}_a \tag{14}$$

$$\boldsymbol{M}_u \dot{\nu}_u + C_u \nu_u + \boldsymbol{C}_{ua} \boldsymbol{\nu}_a = u_u + g_u + f_u \tag{15}$$

where $\boldsymbol{M}_a := \boldsymbol{\Delta}_a^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\Delta}_a$, $\boldsymbol{M}_u := \boldsymbol{\Delta}_u^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\Delta}_u$, and

$$\begin{bmatrix} \boldsymbol{C}_a & \boldsymbol{C}_{au} \\ \boldsymbol{C}_{ua} & C_u \end{bmatrix} := \begin{bmatrix} \boldsymbol{\Delta}_a^{\mathrm{T}} \boldsymbol{C} \boldsymbol{\Delta}_a & \boldsymbol{\Delta}_a^{\mathrm{T}} \boldsymbol{C} \boldsymbol{\Delta}_u \\ \boldsymbol{\Delta}_u^{\mathrm{T}} \boldsymbol{C} \boldsymbol{\Delta}_a & \boldsymbol{\Delta}_u^{\mathrm{T}} \boldsymbol{C} \boldsymbol{\Delta}_u \end{bmatrix}$$

Similarly, for the fully-actuated part in Eqn. (14), we can design the control law as follows:

$$\begin{aligned} \boldsymbol{u}_a := \ & \boldsymbol{M}_a \dot{\bar{\boldsymbol{\nu}}}_a + \boldsymbol{C}_a \bar{\boldsymbol{\nu}}_a + \boldsymbol{C}_{au} \nu_u \\ & - \boldsymbol{k}_u^{\mathrm{P}} \boldsymbol{e}_T - \boldsymbol{k}_u^{\mathrm{I}} \int \boldsymbol{e}_T \mathrm{d}t - \boldsymbol{k}_u^{\mathrm{D}} \boldsymbol{e}_{\boldsymbol{\xi}} - \boldsymbol{g}_a - \boldsymbol{f}_a \end{aligned} \tag{16}$$

Here, $\boldsymbol{k}_u^{\mathrm{P}}, \boldsymbol{k}_u^{\mathrm{I}}, \boldsymbol{k}_u^{\mathrm{D}}$ are 5-dimensional non-negative definite diagonal matrices. $\bar{\boldsymbol{\nu}}_a \in \mathbb{R}^5$ satisfies $\bar{\boldsymbol{\nu}} = [\bar{\boldsymbol{\nu}}_a; \bar{\nu}_u] := \boldsymbol{\Delta}^{-1} \bar{\boldsymbol{\xi}}$, $\boldsymbol{e}_{\boldsymbol{\xi}} := \boldsymbol{\nu}_a - \bar{\boldsymbol{\nu}}_a \in \mathbb{R}^5$, and $\boldsymbol{e}_T \in \mathbb{R}^5$ is the 5-dimensional fully-actuated space error of the platform, s.t.

$$\boldsymbol{e}_T = \boldsymbol{\Delta}_a^{\mathrm{T}} \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 3} \\ (\boldsymbol{d}_0)^{\wedge} & -(\boldsymbol{r}_0^0)^{\wedge} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_0^{\mathrm{T}} (\boldsymbol{p}_0^w - \bar{\boldsymbol{p}}_0^w) \\ \boldsymbol{R}_0^{\mathrm{T}} \bar{\boldsymbol{r}}_0^w(t) \end{bmatrix}$$

For the under-actuated part, setting $u_u = 0$, we obtain the controller output $\boldsymbol{U}$ in Eqn. (13).

Then, we can determine the thrust vector of each sub-aircraft $\boldsymbol{\Lambda}$ according to Eqn. (6). Let $\boldsymbol{B} := \boldsymbol{X}\boldsymbol{Y}$ be a full rank decomposition of $\boldsymbol{B}$, where $\boldsymbol{X} \in \mathbb{R}^{6\times r}$, $\boldsymbol{Y} \in \mathbb{R}^{r\times 3n}$, $r = \mathrm{rank}(\boldsymbol{B})$. For the fully-actuated case, $r = 6$, we can take $\boldsymbol{X} := \boldsymbol{I}_6$, $\boldsymbol{Y} := \boldsymbol{B}$. For the under-actuated case, $r = 5$, we can take $\boldsymbol{Y} := (\mathrm{span}(\boldsymbol{B}^{\mathrm{T}}))^{\mathrm{T}}$. Then, the Moore-Penrose pseudo-inverse matrix of $\boldsymbol{B}$ is

$$\boldsymbol{B}^{\dagger} = \boldsymbol{Y}^{\mathrm{T}} (\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}})^{-1} (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} \boldsymbol{X}^{\mathrm{T}}$$

Therefore, we can obtain the minimum thrust solution, which is the most energy-saving thrust of the sub-aircraft, as

$$\boldsymbol{\Lambda} = \boldsymbol{B}^{\dagger} \boldsymbol{U} \tag{17}$$

The sub-aircraft thrusts $\boldsymbol{\Lambda}_i$ in the collective thrust vector $\boldsymbol{\Lambda}$ from Eqn. (17) are spatial vectors, which can be further decomposed into roll and pitch $\bar{\phi}_i, \bar{\theta}_i \in (-0.5\pi, 0.5\pi)$ commands by specifying arbitrary yaw angles $\bar{\psi}_i \in (-\pi, \pi]$ of each sub-aircraft based on the task requirements, given by

$$\begin{cases} \bar{\phi}_i = \arcsin \dfrac{\Lambda_y \cos\bar{\psi}_i - \Lambda_x \sin\bar{\psi}_i}{|\boldsymbol{\Lambda}_i|} \\ \bar{\theta}_i = \arcsin \left(- \dfrac{\Lambda_x \cos\bar{\psi}_i + \Lambda_y \sin\bar{\psi}_i}{\sqrt{(\Lambda_x \cos\bar{\psi}_i + \Lambda_y \sin\bar{\psi}_i)^2 + \Lambda_z^2}} \right) \end{cases} \tag{18}$$

This decomposition provides the desired roll $\bar{\phi}_i$ and pitch $\bar{\theta}_i$ angles for each sub-aircraft, based on the collective thrust vector and the specified yaw angle $\bar{\psi}_i$.

B. Admittance Controller

In aerial manipulation tasks, traditional position control methods can face challenges when dealing with the geometric constraints of the target operation points. Therefore, we design an admittance controller for the IAP. The admittance controller, acting on the outer loop of the baseline motion controller, responds to external force and torque inputs, $\boldsymbol{f}_e^w, \boldsymbol{\tau}_e^w \in \mathbb{R}^3$, adjusting the original target trajectory based on these external influences.

We conceptualize the aircraft's contact as a mass-damping-spring system. The admittance control law for the fully actuated IAP is formulated as follows [50]

$$\boldsymbol{F}_e^w := \boldsymbol{M}_e \ddot{\boldsymbol{e}}_e + \boldsymbol{D}_e \dot{\boldsymbol{e}}_e + \boldsymbol{K}_e \boldsymbol{e}_e \tag{19}$$

Here, $\boldsymbol{F}_e^w := [\boldsymbol{f}_e^w; \boldsymbol{\tau}_e^w]$ represents the collective external wrench expressed in $\mathcal{F}_w$. $\boldsymbol{M}_e, \boldsymbol{D}_e, \boldsymbol{K}_e$ are 6-dimensional non-negative definite diagonal matrices representing mass, damping, and spring coefficients, respectively. The pose command error $\boldsymbol{e}_e$ expressed in $\mathcal{F}_w$ is defined as

$$\boldsymbol{e}_e := \begin{bmatrix} \bar{\boldsymbol{p}}_r^w - \bar{\boldsymbol{p}}_0^w \\ \bar{\boldsymbol{\gamma}}_r - \bar{\boldsymbol{\gamma}} \end{bmatrix} \tag{20}$$

Here, $\bar{\boldsymbol{p}}_0^w, \bar{\boldsymbol{p}}_r^w \in \mathbb{R}^3$ represent the position commands before and after revision, while $\bar{\boldsymbol{\gamma}}, \bar{\boldsymbol{\gamma}}_r \in \mathbb{R}^3$ represent the Euler angle commands before and after revision.

C. Direct 6-DOF Wrench Control

In certain aerial manipulation tasks, the IAP must maintain a specific pose while actively applying a specific wrench to the manipulated object. For this purpose, we implement a direct wrench PID controller. By substituting the external wrench in Eqn. (10) acquired through measurement or estimation with the PID wrench control law, we can achieve direct 6-DOF wrench control based on the baseline trajectory controller. The direct wrench PID controller is expressed as

$$\begin{aligned} \boldsymbol{U} := \ & \boldsymbol{M} \dot{\bar{\boldsymbol{\xi}}} + \boldsymbol{C} \bar{\boldsymbol{\xi}} - \boldsymbol{k}^{\mathrm{P}} \boldsymbol{e}_T - \boldsymbol{k}^{\mathrm{I}} \int \boldsymbol{e}_T \mathrm{d}t - \boldsymbol{k}^{\mathrm{D}} \boldsymbol{e}_{\boldsymbol{\xi}} - \boldsymbol{G} \\ & - \boldsymbol{k}_w^{\mathrm{P}} \boldsymbol{e}_w - \boldsymbol{k}_w^{\mathrm{I}} \int \boldsymbol{e}_w \mathrm{d}t - \boldsymbol{k}_w^{\mathrm{D}} \dot{\boldsymbol{e}}_w \end{aligned} \tag{21}$$

Here,

$$\boldsymbol{e}_w := \begin{bmatrix} \boldsymbol{f}_e - \bar{\boldsymbol{f}}_e \\ \boldsymbol{\tau}_e - \bar{\boldsymbol{\tau}}_e \end{bmatrix}$$

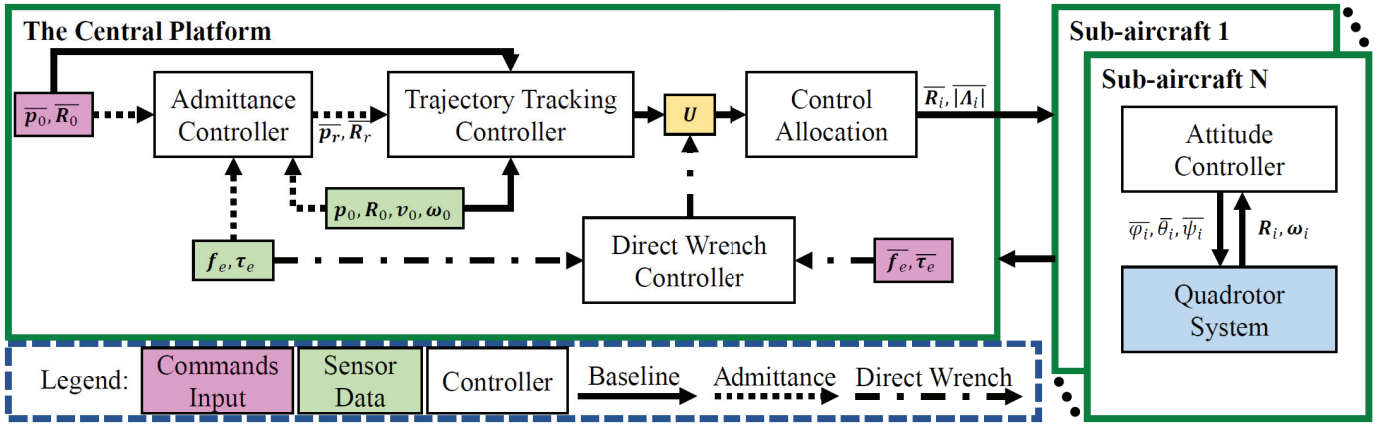


Fig. 3. Overall control architecture of the IAP system.

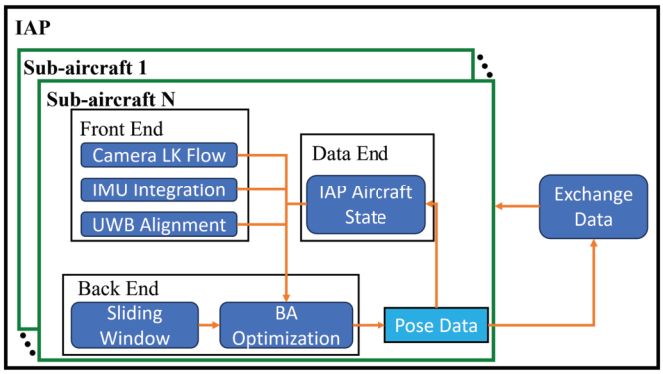


Fig. 4. The architecture of VIRPO in the IAP.

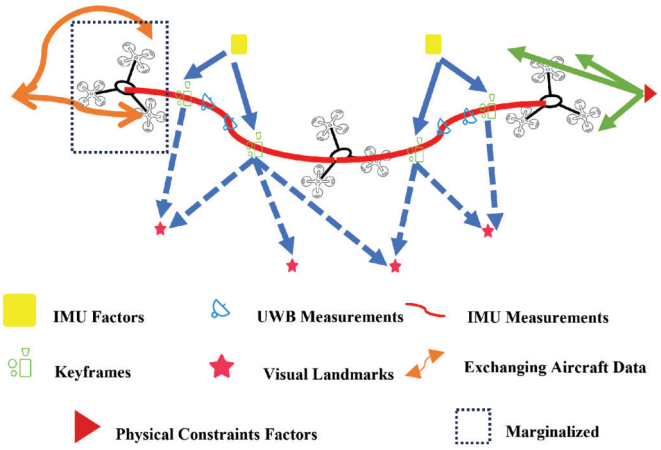


Fig. 5. Tightly coupled formulation of VIRPO.

represents the external wrench error, where $\bar{f}_e, \bar{\tau}_e \in \mathbb{R}^3$ are the external force and torque commands expressed in $\mathcal{F}_0$. Additionally, $k_w^{\mathrm{P}}, k_w^{\mathrm{I}}, k_w^{\mathrm{D}}$ are 6-dimensional non-negative definite diagonal gain matrices.

Fig. 3 depicts the full control system framework.

D. Visual-Inertial Localization Fusion

In our previous work, we introduced a theoretical framework for IAPs that leverages the physical constraints among IAP sub-aircraft [51]. We developed a tightly-coupled algorithm, termed Visual-Inertial-Range-Physical Odometry (VIRPO), to enhance localization accuracy, validated using datasets collected from our real IAP physical prototype. In this paper, we aim to apply VIRPO to a dataset obtained from our simulator to validate its efficacy. Thus, this section provides an overview of the optimization-solving problems and the main optimization process of VIRPO.

The VIRPO algorithm operates on each sub-aircraft, utilizing camera image data, IMU readings, UWB sensor data, and IAP data, as illustrated in Fig. 4. It integrates UWB-ranging data and incorporates the physical structural constraints among IAP sub-aircraft to improve localization accuracy. In the IAP system, each sub-aircraft utilizes data from cameras, IMUs, UWB sensors, and the IAP's odometry data in a Bundle Adjustment (BA) optimization to execute the VIRPO algorithm. Additionally, each sub-aircraft shares its odometry data with other sub-aircraft.

The optimization process entails managing UWB residuals and exchanging data, including pose, velocity, and angular velocity, within the VIRPO framework, as illustrated in Fig. 5. In the tightly-coupled formulation of VIRPO, the sliding window integrates information from IMUs, visual inputs, physical constraints, and UWB measurements.

The state vector of the i-th sub-aircraft is defined as

$$\mathcal{X}^i = [\mathbf{x}_0^i, \mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_K^i, \mathbf{x}_c^b, \lambda_0^i, \lambda_1^i, \ldots, \lambda_M^i]$$

Here,

$$\mathbf{x}_k^i = [\mathbf{p}_{b_k}, \mathbf{v}_{b_k}, \mathbf{q}_{b_k}, \mathbf{b_a}, \mathbf{b_g}]$$

$\mathbf{x}_k^i$ represents the IMU state at the time when the k-th image of the i-th sub-aircraft is captured. It includes the position, velocity, and orientation of the IMU in the i-th aircraft's local frame $\mathcal{F}_i^L$, along with the acceleration bias $\mathbf{b_a}$ and gyroscope bias $\mathbf{b_g}$ in the IMU body frame. K is the total number of key frames, and λ_m^i is the inverse distance of the m-th feature of the i-th sub-aircraft from its first observation. A visual-inertial-UWB-physical constraint BA is performed to estimate the

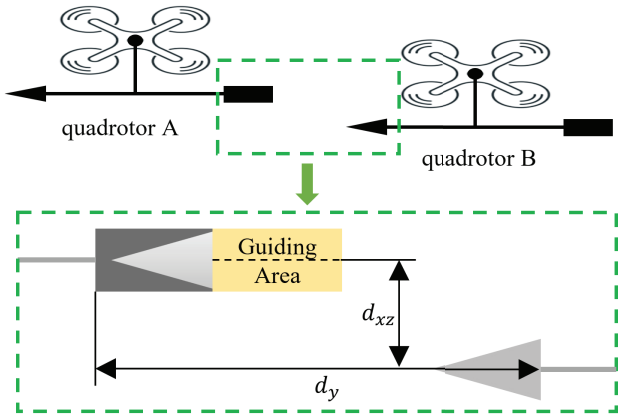


Fig. 6. Schematic diagram of the docking mechanism.

states. The formulation for the i-th sub-aircraft is as follows:

$$\begin{aligned}\min_{\mathcal{X}^i}\Big\{&\|\mathbf{r}_p-\mathbf{H}_p\mathcal{X}^i\|^2+\sum_{k\in\mathfrak{B}}\|\mathbf{r}_{inertial}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}},\mathcal{X}^i)\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}}\\&+\sum_{(m,k)\in\mathfrak{C}}\|\mathbf{r}_{visual}(\hat{\mathbf{z}}^{c_k}_{m},\mathcal{X}^i)\|^2_{\mathbf{P}^{c_k}_{m}}\\&+\|\mathbf{r}_{kp}(\mathcal{X}^i,\mathcal{F})\|^2_{\sigma_l^{-2}}+\|\mathbf{r}_{kv}(\mathcal{X}^i,\mathcal{F})\|^2_{\sigma_\omega^{-2}}\\&+\|\mathbf{r}_{kq}(\mathcal{X}^i,\mathcal{F})\|^2_{\sigma_{att}^{-2}}+\|\mathbf{r}_{uwb}(\mathcal{X}^i,\mathcal{J})\|^2_{\sigma_d^{-2}}\\&+\sum_{o\neq i}\|\mathbf{r}_{drift}(\mathcal{X}^o)\|^2_{\sigma^{-2}_{drift}}\Big\}\end{aligned}\tag{22}$$

Here, $\mathbf{r}_{inertial}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}},\mathcal{X}^i)$ and $\mathbf{r}_{visual}(\hat{\mathbf{z}}^{c_j}_{m},\mathcal{X}^i)$ are residuals for IMU and visual measurements, respectively. $\mathfrak{B}$ is the set of all IMU measurements, and $\mathfrak{C}$ is the set of features observed at least twice in the current sliding window. $\mathbf{r}_p$ and $\mathbf{H}_p$ represent prior information obtained from marginalization [52]. $\mathbf{r}_{kp}(\mathcal{X}^i)$, $\mathbf{r}_{kv}(\mathcal{X}^i)$, $\mathbf{r}_{kq}(\mathcal{X}^i)$ are residuals for position, angular velocity, and attitude constraints among sub-aircraft. $\mathbf{r}_{uwb}(\mathcal{X}^i)$ is the residual for UWB measurements. $\mathcal{F},\mathcal{J}$ represent the set of pose data obtained for the other sub-aircraft in the IAP (including the central platform) and the corresponding set of UWB range data. σ_l, σ_v, σ_{att}, and σ_d represent the covariance of the position, velocity, attitude, and range constraints, respectively. To account for accumulated drift errors observed in other aircraft within the IAP, residual drift error terms $\|\mathbf{r}_{drift}(\mathcal{X}^o)\|$ from these aircraft are incorporated into our tightly coupled optimization. This inclusion aims to enhance the robustness of the system and mitigate the influence of outliers. The Ceres solver is utilized for solving this nonlinear problem in our implementation.

E. Air Docking via Reinforcement Learning

In this section, we explore the scenario of air docking involving an IAP system comprising two quadrotors. Each quadrotor is equipped with conical and cylindrical docking mechanisms, as illustrated in Fig. 6, with d_{xz} and d_y as the radial and axial errors, respectively. Upon successful docking, the mechanism locks, enabling combined flight.

Due to three distinct processes involved in air docking (single-aircraft flight, docking, and combined flight), achieving stable flight of each vehicle, stability during docking, and stable hover after docking pose a multi-objective problem [53]. This complexity may lead to catastrophic forgetting and is not well-suited for conventional single-stage reinforcement learning methods [54]. Additionally, the contact process is highly complex and challenging to model accurately. Traditional docking methods, such as hybrid force/position control [55], are sensitive to sensor noise, which can significantly impact the success rate of air docking, making high success rates difficult to achieve. Therefore, for the process of two quadrotors docking in mid-air to form a composite body, we propose an offline-to-online reinforcement learning approach [56], [57], as depicted in Fig. 7.

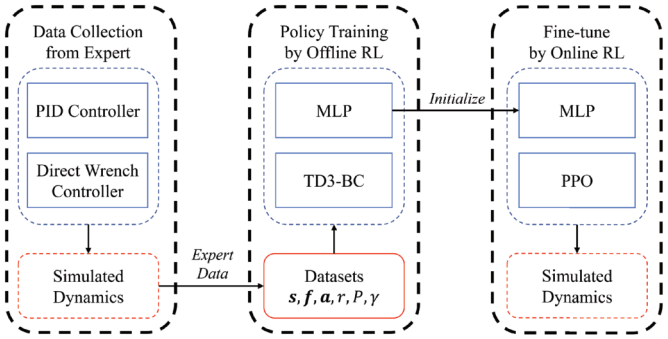


Fig. 7. Flowchart of the proposed offline-to-online RL method.

This method consists of three stages: expert data collection, offline reinforcement learning training, and online reinforcement learning training. In the expert data collection stage, docking tasks are executed in the simulator using traditional PID control and direct wrench control methods. Data including the states of the two drones $\boldsymbol{s}^A=[\boldsymbol{p}^w_A-\bar{\boldsymbol{p}}^w_A,\boldsymbol{v}_A,\boldsymbol{R}_A,\boldsymbol{\omega}_A]$, $\boldsymbol{s}^B=[\boldsymbol{p}^w_B-\bar{\boldsymbol{p}}^w_B,\boldsymbol{v}_B,\boldsymbol{R}_B,\boldsymbol{\omega}_B]$, contact wrench $\boldsymbol{f}_c=[f_x,f_y,f_z,\tau_x,\tau_y,\tau_z]$, actions taken by the two drones $\boldsymbol{a}^A=[\dot{\bar{\boldsymbol{v}}}_A,\bar{\boldsymbol{\omega}}_A]$, $\boldsymbol{a}^B=[\dot{\bar{\boldsymbol{v}}}_B,\bar{\boldsymbol{\omega}}_B]$, and the success status of docking γ are collected to establish an expert database. During the training phase, a reward function r is designed considering the tracking errors of the two quadrotors' poses, whether the two quadrotors are in the guiding area, their relative position during docking, and the magnitude of the contact force f_y along the y-axis. In the offline reinforcement learning stage, leveraging the expert database, a Multi-Layer Perceptron (MLP) policy P is pre-trained using the offline reinforcement learning method TD3-BC to initialize the MLP [58]. Once pre-training is complete, in the online reinforcement learning stage, the MLP is fine-tuned using the online reinforcement learning method PPO in simulation to obtain the final policy [59].

V. Simulator Framework Design

A. Overall Architecture

Our simulator framework adopts a modular design, enabling functional expansion and development. The rendering engine, the physics engine, and the ROS (Robot Operating System) interfaces constitute its three main modules. The rendering engine module generates realistic visual scenes and produces high-fidelity camera data. The physics engine module simulates the dynamic behavior of the IAP system, such as

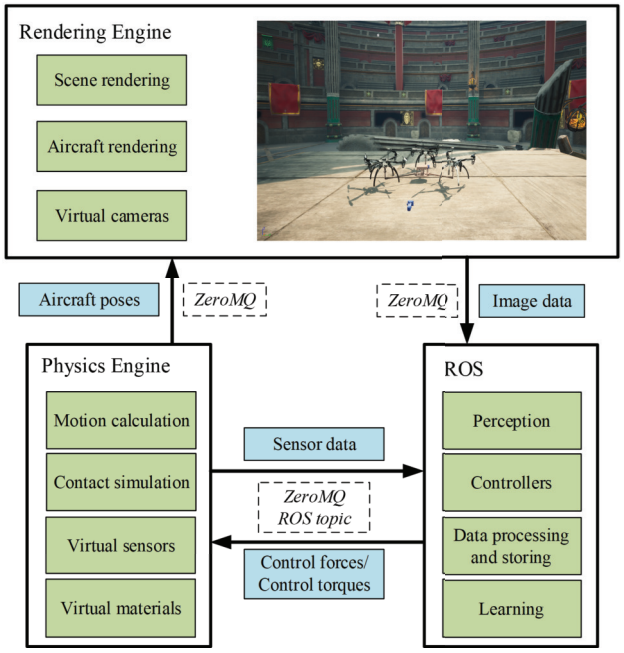


Fig. 8. Overall design of IAP simulator framework.

the motion, collision, and mechanical reactions of the IAPs. The ROS interface module facilitates communication between different modules and the ROS network. It processes data exchanged among various modules, such as the rendering engine and physics engine, and interfaces with ROS program nodes. Additionally, it handles data storage and temporal information processing to synchronize concurrent module operations. This module serves as the backbone for simulating and validating perception, control, and learning algorithms of IAPs and generating training datasets.

To facilitate inter-module data exchange, our simulator framework adopts the open-source communication framework ZeroMQ, which is built upon socket architecture. Following the design principles of ROS, we implement a pub-sub pattern to enable multi-to-multi parallel communication. This design not only provides a robust framework foundation for efficient data transmission between modules but also enables distributed computing, thereby enhancing the computational efficiency and overall performance of the simulator framework.

The design approach depicted in Fig. 8 is employed. In this scheme, the physics engine module computes the state information of the aircraft and simultaneously transmits the computed state data to both the rendering engine and the ROS network. The physics engine module may run independently or within the ROS network. If it runs within ROS, it communicates directly with other nodes in the ROS network via ROS topics. The rendering engine module then renders the aircraft accordingly within the scene and forwards the rendered virtual image data to the ROS network. Upon receiving the state and image data of the aircraft, ROS executes nodes like perception and control algorithms, generates control wrench command information based on the controller outputs, and then sends the command information back to the physics engine module. Upon receiving the control command information, the physics engine updates the model states, thus forming a closed-loop data exchange. This design enables efficient and synchronized data exchange and collaboration among the physics engine, rendering engine, and ROS, facilitating comprehensive data simulation of the IAP system. Moreover, this approach exhibits high scalability and customizability, as each module can be replaced with engines with similar functions, and whether or not to use the rendering engine can be decided based on task requirements, potentially conserving computational resources if the rendering engine is not utilized. Such a design approach meets the requirements of various application scenarios for IAPs while providing a flexible foundation for functional expansion and modular development.

B. Rendering

The simulator framework is developed using C++ and Blueprints within Unreal Engine 5 (UE5) as the rendering engine. Quadrotor and IAP mesh models from CAD software can be imported into Unreal Editor and subsequently configured for appearance and properties. Engine-side handling of dynamic lighting is integrated, while various configurations of IAP rendering models such as star-shaped, square-shaped, and line-shaped are obtained by setting parameters, particularly in terms of joint type and positioning.

Moreover, the simulator framework provides an image capture interface that furnishes RGB images, allowing users to produce real-time high-fidelity images during simulation. Virtual cameras are integrated into simulation scenes via the USceneCaptureComponent2D component, which can be set to a fixed position and viewpoint in the scene or tethered to an object to maintain relative pose. Users are afforded flexibility in adjusting camera settings such as exposure and field of view. Captured data from these cameras are stored within the UTextureRenderTarget2D component, with customizable parameters including encoding format and pixel configuration. Images, captured per frame during simulation, are encoded in PNG format and published via the ZeroMQ data interface, thereby offering potential utility in the realm of vision algorithms.

C. Physics Engine

A data interface has been designed for RaiSim [37], enhancing the simulator's capacity for dynamic simulation of IAP platforms, thereby facilitating more precise dynamics and contact simulation. RaiSim stands as an advanced physics engine used for simulating multi-joint robots, demonstrating exceptional performance in the simulation field, particularly in contact simulation [60].

In the simulation initialization phase, the various joints and rigid body components of the simulated IAP are defined using Universal Robot Description Format (URDF) files. Additionally, for contact simulation, the mobility, material characteristics, and collision groups of objects within the operating scene are configured either programmatically or through URDF file specifications.

In our setup, to distinguish external wrenches and active wrenches, during the model update phase, active forces or

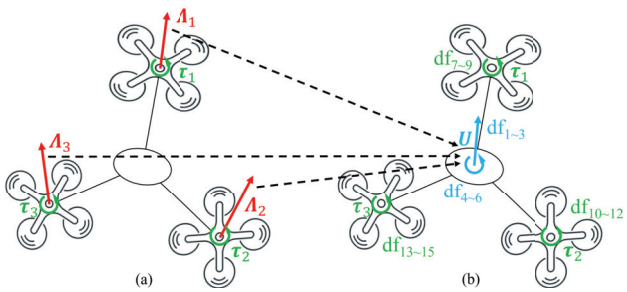


Fig. 9. Equivalence force transformation of the three-star IAP.

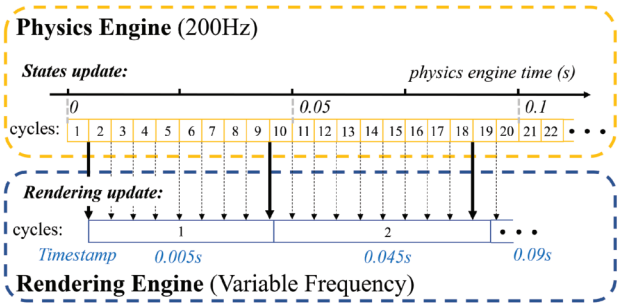


Fig. 10. Image timestamp synchronization illustration.

torques are assigned to each joint based on its generalized degrees of freedom. In other words, forces acting on rigid bodies are equivalently converted into joint active forces, torques, and resultant wrenches on the central platform, ensuring that the equivalent mechanical model matches the original model. Consider a three-star IAP as an example, featuring 15 degrees of freedom comprising three translational (df_{1-3}) and three rotational (df_{4-6}) degrees of freedom for the central platform, along with three rotational degrees of freedom for each of the three sub-aircraft with spherical joints (df_{7-9}, df_{10-12}, df_{13-15}). Spherical joints ideally transmit forces but not torques. Therefore, the equivalent approach is to transfer all active thrust forces from the sub-aircraft $\mathbf{\Lambda}_{1-3}$ to the central platform using mechanics principles, thereby generating a wrench $\boldsymbol{U}$ on the central platform $\mathcal{R}_0$. Additionally, the active torques generated by sub-aircraft $\boldsymbol{\tau}_{1-3}$ are loaded in the form of joint active torques, with counter-torques applied to the central platform connected by the spherical joint to obtain the resultant torques applied to the central platform. This process is visually illustrated in Fig. 9. The dynamics simulation update process for other configurations of IAP adheres to the same underlying principles.

D. Sensors

Our simulator provides sensor models for IMU and UWB. In the case of IMU sensors, the gyroscope and accelerometer serve as the primary components. We adopt the modeling approach introduced in AirSim [17], incorporating Gaussian noise and bias drift over time to the ground truth, as

$$\hat{\omega} = \omega + \eta_a^{\omega} + b_t^{\omega},\ \eta_a^{\omega} \sim N(0, r_a^{\omega})$$

where

$$b_t^{\omega} = b_{t-1}^{\omega} + \eta_b^{\omega},\ \eta_b^{\omega} \sim N(0, r_b^{\omega})$$

and

$$\hat{a} = a + \eta_a^{a} + b_t^{a},\ \eta_a^{a} \sim N(0, r_a^{a})$$

where

$$b_t^{a} = b_{t-1}^{a} + \eta_b^{a},\ \eta_b^{a} \sim N(0, r_b^{a})$$

For UWB sensors, we adopt the simulation sensor modeling approach proposed by Xie et al. [61], adding Gaussian noise to the computed ground truth, as

$$\hat{\gamma} = \gamma + \eta_a^{\gamma},\ \eta_a^{\gamma} \sim N(0, r_a^{\gamma})$$

The frequency of the sensors aligns with the ground truth frequency within the simulation.

The pose, velocity, IMU, and UWB sensor data are all synchronized and updated based on the results from the physics engine, with timestamps attached. Considering that the rendering frequency of images is not fixed, image timestamps cannot be calculated based on fixed intervals, and due to the large size of image data, which may lead to image read/write times, a significant delay of over 10ms may occur, that could impact algorithms with high time-synchronization requirements such as VINS. Therefore, we have implemented a method within the rendering engine. This method computes and attaches the timestamp of the latest frame's pose used for rendering to the image timestamp, ensuring complete synchronization between image and aircraft state timestamps. An illustration of the image timestamp synchronization method is shown in Fig 10.

Assuming that the physics engine updates the aircraft state at a rate of 200Hz, each cycle of the physics engine spans 0.005s. The timestamp for the state of each cycle can be initiated from 0 and incremented by 0.005s for subsequent cycles. Operating asynchronously to the physics engine in terms of time, the rendering engine's cycles do not align with those of the physics engine. The rendering engine visually represents the aircraft's pose in the scene based on the latest received state and assigns the timestamp of this frame to the image data retrieved for the ongoing cycle. In Fig. 10, in approximately 0.1s, the rendering engine visualizes the state data from the 1st, 9th, and 18th frames of the physics engine within cycles 1, 2, and 3, respectively, and timestamps them according to the corresponding physics engine times: 0.005s, 0.045s, and 0.09s.

VI. Experiments and Results

This section discusses the experiments performed to demonstrate the usability and fidelity of our simulator. We explored various IAP configurations, including the three-star IAP, three-square IAP, two-line IAP, and three-line IAP, and validated the simulator's accuracy in replicating IAP dynamics. We also investigated flight contact control for the three-star IAP, focusing on two experiments: admittance control and direct wrench control. Additionally, we examined the performance of tightly-coupled SLAM localization within the simulator, highlighting its real-time performance and accuracy in vision and sensor designs. Finally, we discussed the utility of the simulator in training and validating air-docking procedures and its potential applications in machine learning tasks with contact control.

TABLE I
CONFIGURATION PARAMETERS OF THE THREE-STAR IAP

m_{center}/kg	m_{sub}/kg	d_{h}/m	d_{v}/m
1.1	1.6	0.48	0.07

TABLE II
PARAMETERS OF THE THREE-STAR IAP MOTION CONTROLLER

	x	y	z	ϕ	θ	ψ
$\boldsymbol{k}^{\text{P}}$	8	8	8	8	5	6
$\boldsymbol{k}^{\text{I}}$	1	1	1	1	1	1
$\boldsymbol{k}^{\text{D}}$	10	10	5	10	10	15

Both our real flight experiments and simulation experiments utilize the DJI F450 drones as sub-aircraft. The drone and the IAP central platform models employed in the simulation are exported from CAD software, including SolidWorks or Autodesk Inventor, ensuring alignment with real-world counterparts. Parameters such as mass, and spherical joint installation positions are configured to approximate real measurement values. Both CAD models and parameters in the simulator can be customized based on real-flight models. In addition, the F450 model used in the docking task is a custom model equipped with a docking mechanism and has a different CAD model and mass parameters compared to other experiments in this paper. Real flight experiments are carried out in an open outdoor field, employing a PX4 open-source flight controller integrated with a built-in 9-DOF IMU (200Hz) and providing positioning information (200Hz) through multi-agent visual-inertial localization with loose fusion of odometry and kinematics [62], [63]. The central platform is equipped with NVIDIA TX2, running the IAP controller (200Hz). Simulation executes on a laptop equipped with an Intel i7-11800H @ 2.30GHz CPU, NVIDIA GeForce RTX 3070 Laptop GPU, and 64GB RAM. Rendering-intensive simulation tasks are handled using Unreal Engine 5.1.1 on Windows 11. ROS and RaiSim operate within WSL2, utilizing the 20.04 version. Both the IAP real flight experiments and simulation experiments use the same controller, deployed as ROS nodes written in C++, running on Ubuntu. The only distinction lies in the interface topic name for differentiation purposes.

A. Motion Control Comparison of the Three-Star IAP

This section presents an assessment of the motion control performance of the three-star IAP by comparing actual flight testing and simulation outcomes. Since the three sub-aircraft are not collinear, i.e., they do not satisfy (8), the three-star IAP is a fully actuated IAP. The primary objective is to validate the efficacy and fidelity of our simulator in accurately replicating motion control. The simulation entails an IAP configuration mirroring that of the real flight experiments, encompassing identical mass parameters and structural configurations. The IAP is composed of three sub-aircraft connected by spherical joints, forming an equilateral triangle. Each joint is positioned equidistantly from the center of the IAP in both horizontal d_{h} and vertical d_{v} directions. Detailed parameters are outlined in TABLE I.

TABLE III
CONFIGURATION PARAMETERS OF THE THREE-SQUARE IAP

$\boldsymbol{d}_1$/m	$\boldsymbol{d}_2$/m	$\boldsymbol{d}_3$/m	$\psi_{1,2,3}^{\text{initial}}$/°
(0.62;0;0.05)	(-0.26;-0.52;0.05)	(-0.26;0.52;0.05)	0

Furthermore, uniform controller parameters in (10) are employed in both simulation and real-world scenarios, as outlined in TABLE II. Implemented as ROS nodes, the controller maintains uniformity across both platforms, with variations only in the interface.

The experimental procedure commences with outdoor flight testing to collect target pose commands and actual flight data. These commands, generated programmatically, direct the aircraft along a spiraling trajectory toward a predetermined fixed point. Subsequently, the actual flight target poses serve as inputs for the simulation, where the same control strategy is executed. The resulting simulated poses are then compared with the actual flight data. The comparative results of pose estimation are illustrated in Fig. 11.

The result reveals slightly larger tracking errors during actual flight than those observed in simulation. This discrepancy may be attributed to uncontrollable stochastic errors arising from factors such as wind, sensor noise, and configuration parameters, which cannot be replicated in the simulation environment. Nevertheless, both simulation and actual flight exhibit comparable tracking performance. The absolute errors in position comparison remain within approximately 0.4 meters, while the roll and pitch angle errors are maintained below 4°. Although the yaw angle is more susceptible to wind disturbances, it remains below 12°.

This comparative analysis validates the capability of our simulator to realistically simulate the 6-DOF motion control of the three-star IAP, thereby facilitating convenient and realistic experimentation.

B. Motion Control Comparison of the Three-Square IAP

This section conducts a comparative experiment to assess the motion control performance of the three-square IAP. The experiment employs the same controller and control parameters as those utilized for the three-star IAP, but with necessary adjustments to parameters pertaining to the distribution of sub-aircraft positions and initial yaw angles, to evaluate the robustness of the controller across different IAP configurations. The overarching objective is to validate the feasibility and fidelity of our simulator in simulating fully-actuated IAPs with different configurations. Detailed initial configuration parameters are provided in TABLE III.

In the real flight experiment, position commands are given via a remote controller, while pre-defined attitude commands are triggered through a designated switch channel, enabling 6-DOF pose control. Both the commands and actual flight poses are recorded in rosbag format. Subsequently, the recorded rosbag commands are replayed within the simulator environment, where the identical control strategy is executed to obtain simulation results. The results are depicted in Fig. 12.

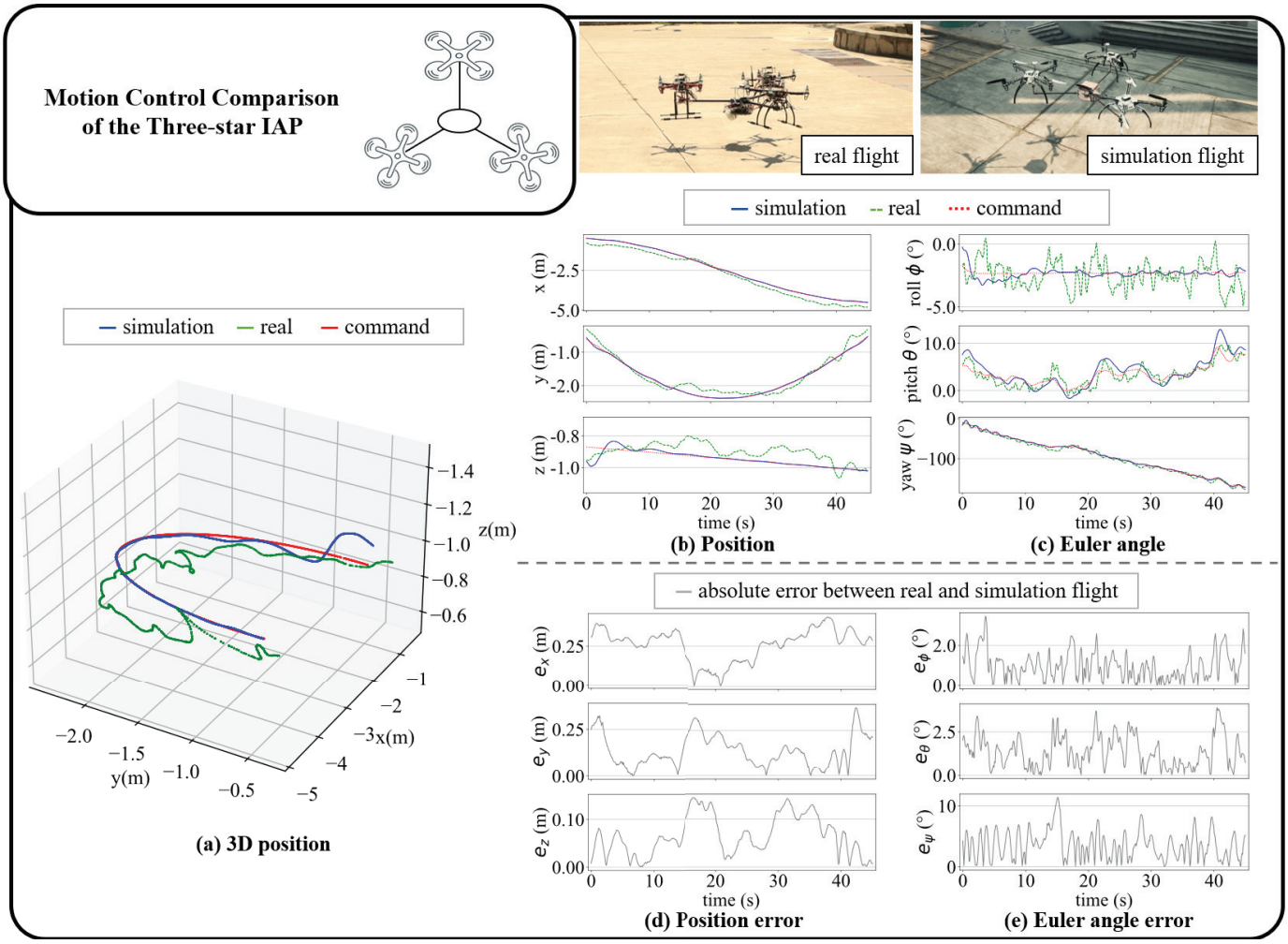


Fig. 11. Motion control comparison of the three-star IAP.

The error plots reveal slightly amplified position errors compared with those observed in the three-star IAP experiment. This amplification of errors can be attributed to the utilization of control parameters tailored for star-shaped configurations in the experiment. Additionally, it is noted that both actual flight and simulation exhibit a degree of lag or overshoot in tracking angle commands, particularly when the derivative of angle commands changes abruptly. This behavior is conspicuously absent in the smoother trajectory commands employed in the three-star IAP experiment.

Despite the inability of simulation flight to precisely replicate the motion trajectories observed in actual flight due to unreplicable external disturbances, our simulator still demonstrates a similar trend. Through the comparative experiment of three-square IAP motion control, the robustness of the employed controller for fully actuated IAPs with different configurations is validated, and the broad applicability of our simulator to various configurations of fully actuated IAPs is demonstrated.

C. Motion Control Simulation of the Two-Line IAP

This section presents a simulation experiment to verify the feasibility and fidelity of our simulator in replicating the three-dimensional motion of under-actuated IAPs. Specifically, the experiment focuses on replicating the real flight motion trajectories of the two-line IAP reported in prior literature [14] by Lee et al. using our simulation platform. Since the two-line IAP has only two sub-aircraft, it meets the conditions of (8) for the under-actuation of IAP.

In our simulation for the two-line IAP, we adopt the control strategy described in (16). The motion control simulation results are shown in Fig. 13. Specifically, the simulated IAP is configured to track a circular trajectory on the XY plane at a constant velocity of 0.5 m/s post takeoff, while simultaneously maintaining a fixed orientation. Our evaluation focuses on assessing the simulated controller's capacity to adeptly, swiftly, and accurately track and replicate the real flight position variations associated with the relatively high velocity while ensuring that the absolute error in orientation remains negligible.

Despite lacking original real flight data for comparison, the results of the simulated motion trajectories offer insights into the efficacy of our simulator in replicating the motion control behavior of the two-line IAP. The consistency between the simulated trajectories and the expected motion characteristics, as described in the literature, demonstrates the validity and

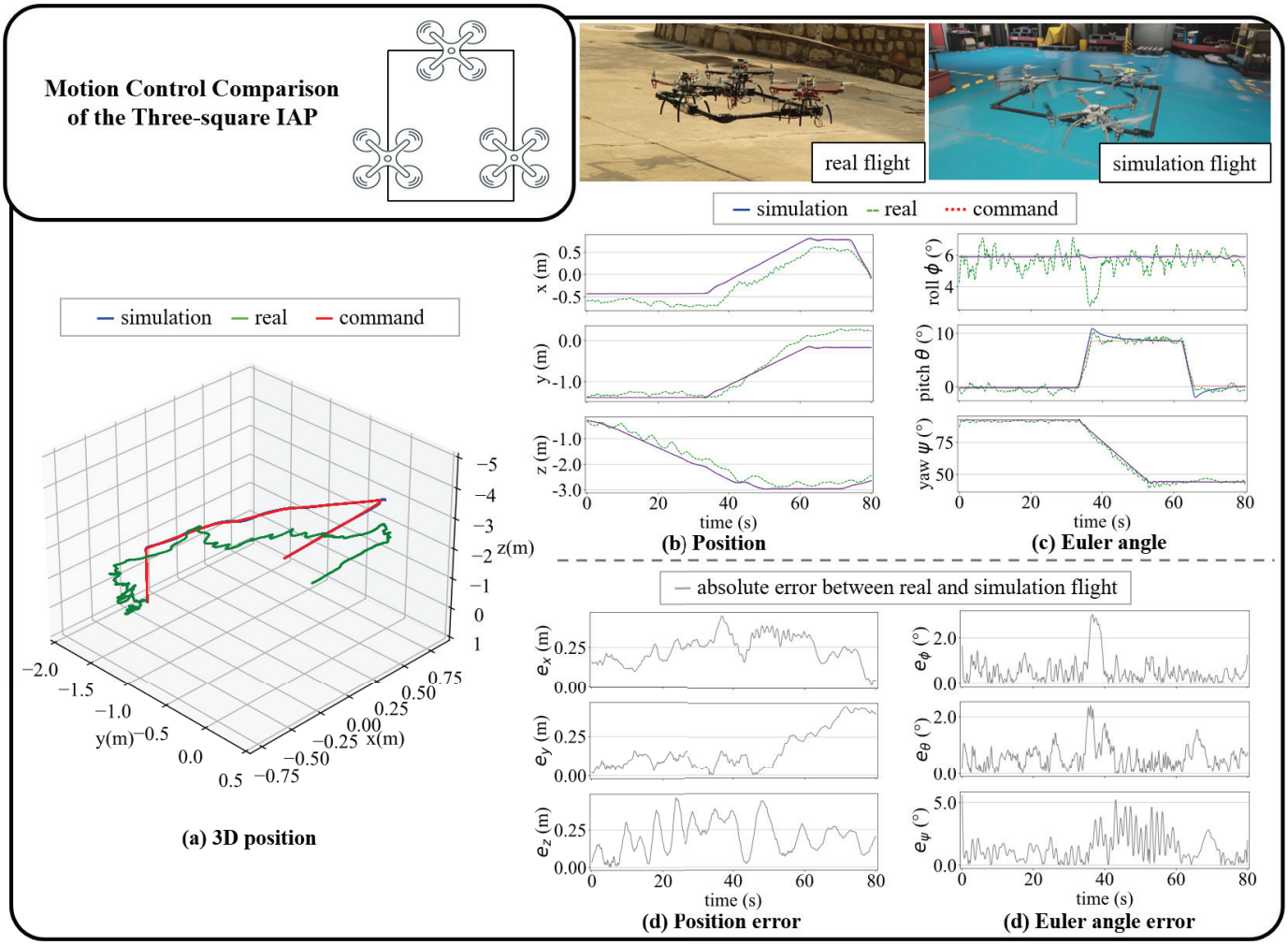


Fig. 12. Motion control comparison of the three-square IAP.

fidelity of our simulation platform for under-actuated IAP motion control strategies.

D. Motion Control Simulation of the Three-Line IAP

This section focuses on simulating the motion control of a three-line IAP using our simulation platform. In this simulation scenario, the three sub-aircraft of the IAP are aligned in a straight line parallel to the axis of the pole. The yaw commands of the sub-aircraft are constantly synchronized with the yaw commands of the IAP, directing them along the axis of the pole. The spacing between adjacent sub-vehicles is set to 1 meter.

Dynamic 5-DOF pose commands are inputted during the experiment. The experiment results, as depicted in Fig. 14, demonstrate the simulated three-line IAP's adeptness in accurately tracking the dynamic three-dimensional position, pitch angle, and yaw angle commands, as well as dynamic three-dimensional linear and two-dimensional angular velocity commands.

The experiment validates the capability of our simulator to simulate under-actuated IAPs with different numbers of sub-aircraft, thus highlighting its versatility. Furthermore, for IAP configurations with more sub-aircraft, their dynamics, controllers, and simulations are all similar to those configurations verified previously. Therefore, these four experiments of fully-actuated and under-actuated IAPs corroborate the effectiveness of the control allocation methods proposed in (17) and (18) for both fully-actuated and under-actuated IAPs with varying numbers of sub-aircraft.

E. Admittance Control Simulation of the Three-Star IAP

This section presents a simulation experiment focusing on the admittance control of a three-star IAP. Inspired by previous experiments conducted by Antonio et al. [21] on a fully-actuated hexacopter peg-in-hole task, our experiment involves simulating the insertion of a pole mounted on the central platform of the IAP into a fixed funnel while maintaining a horizontal attitude command. The pole is connected to the central platform along the negative x-axis. During the peg-in-hole process, the contact between surfaces results in the continuous variation of the number and position of contact points. Directly obtaining the true values of external forces and torques from the simulator may lead to spikes when abrupt changes occur in the contact conditions, which is detrimen-

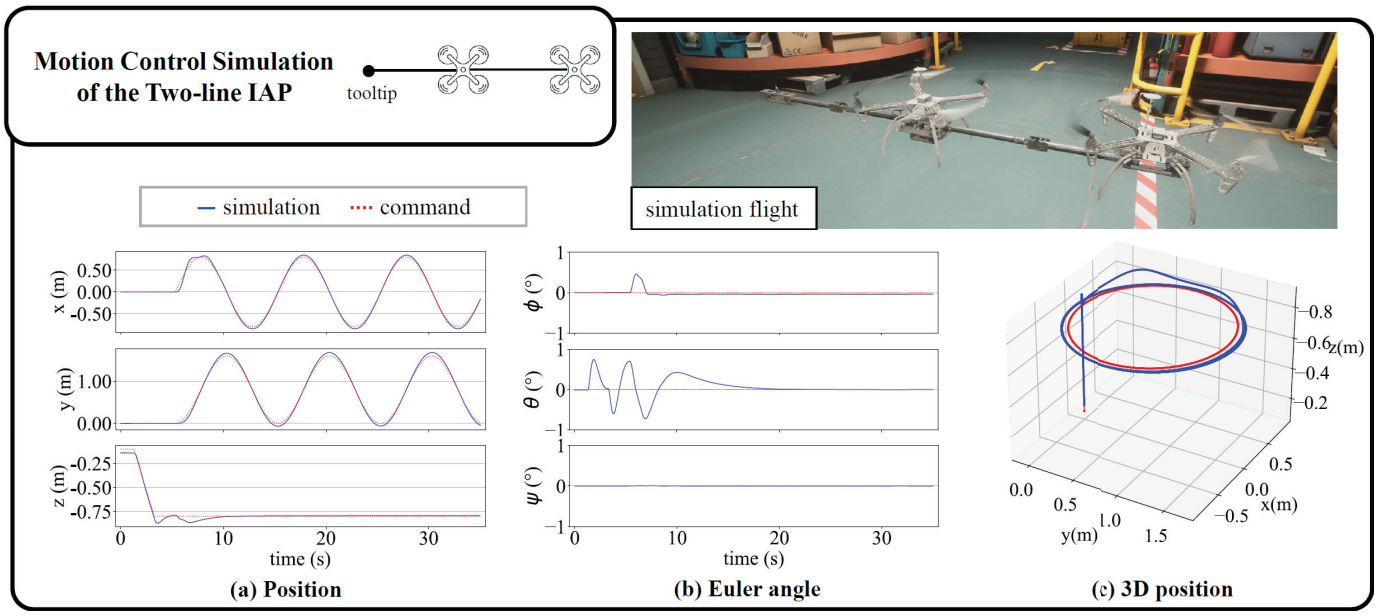


Fig. 13. Motion control simulation of the two-line IAP.

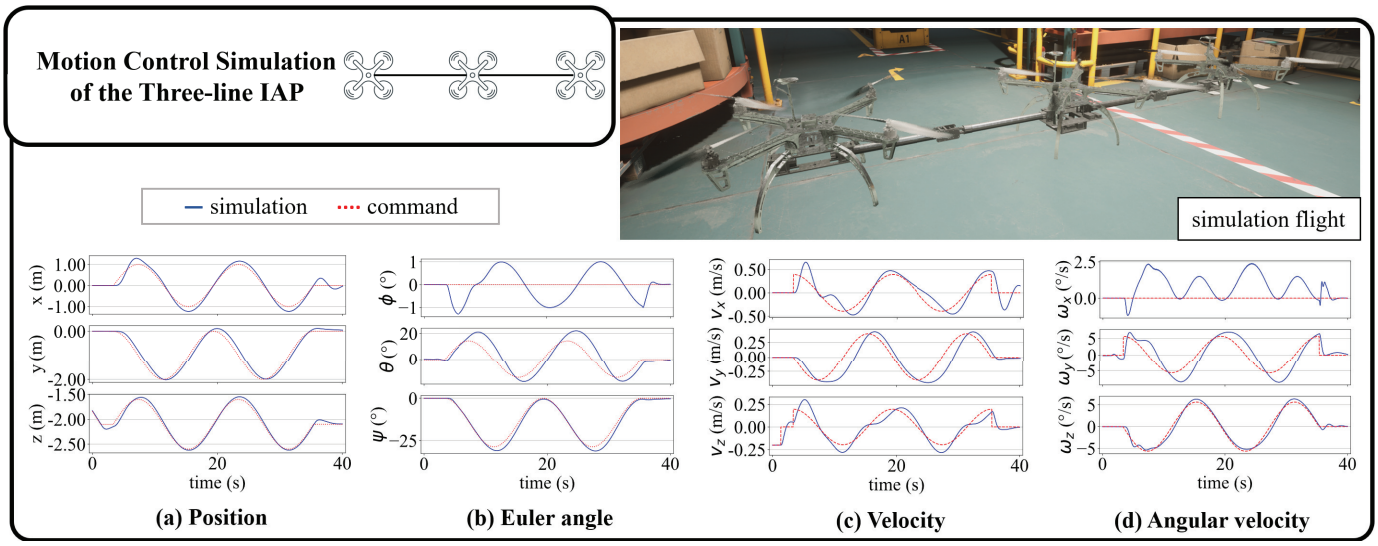


Fig. 14. Motion control simulation of the three-line IAP.

tal to control. Therefore, second-order low-pass filtering is employed to process contact forces and torques, which is

$$G_{2\text{order}}(s) = \frac{\omega_c^2}{s^2 + 2\xi\omega_c s + \omega_c^2}$$

where the cut-off frequency $\omega_c = 10$ and damping ratio $\xi = 1.414$ for each axis of the external wrench. Using simulated filtered external forces and moments applied to the center of the IAP, described in the inertial frame $\mathcal{F}_w$, the admittance controller autonomously adjusts the pose commands to facilitate smooth insertion of the pole into the funnel and maintain contact.

The schematic of the task, along with the dimensions and installation parameters of the pole and funnel, is depicted in Fig. 15. The parameters of the admittance controller utilized in the experiment are detailed in TABLE IV.

TABLE IV
PARAMETERS OF THE THREE-STAR IAP ADMITTANCE CONTROLLER

	x	y	z	ϕ	θ	ψ
$\boldsymbol{M}_e$	1	1	1	1	1	1
$\boldsymbol{D}_e$	28.28	28.28	28.28	28.28	14.14	28.28
$\boldsymbol{K}_e$	100	100	100	100	25	100

During the experiment, given the necessity for flexible adjustments primarily in the pitch angle direction to ensure stable contact, the pitch angle admittance parameters are set relatively small. This facilitates greater adaptability of the pitch angle command to changes in external moments. Conversely, the admittance parameters for the other five dimensions are set larger to mitigate the influence of external forces and moments.

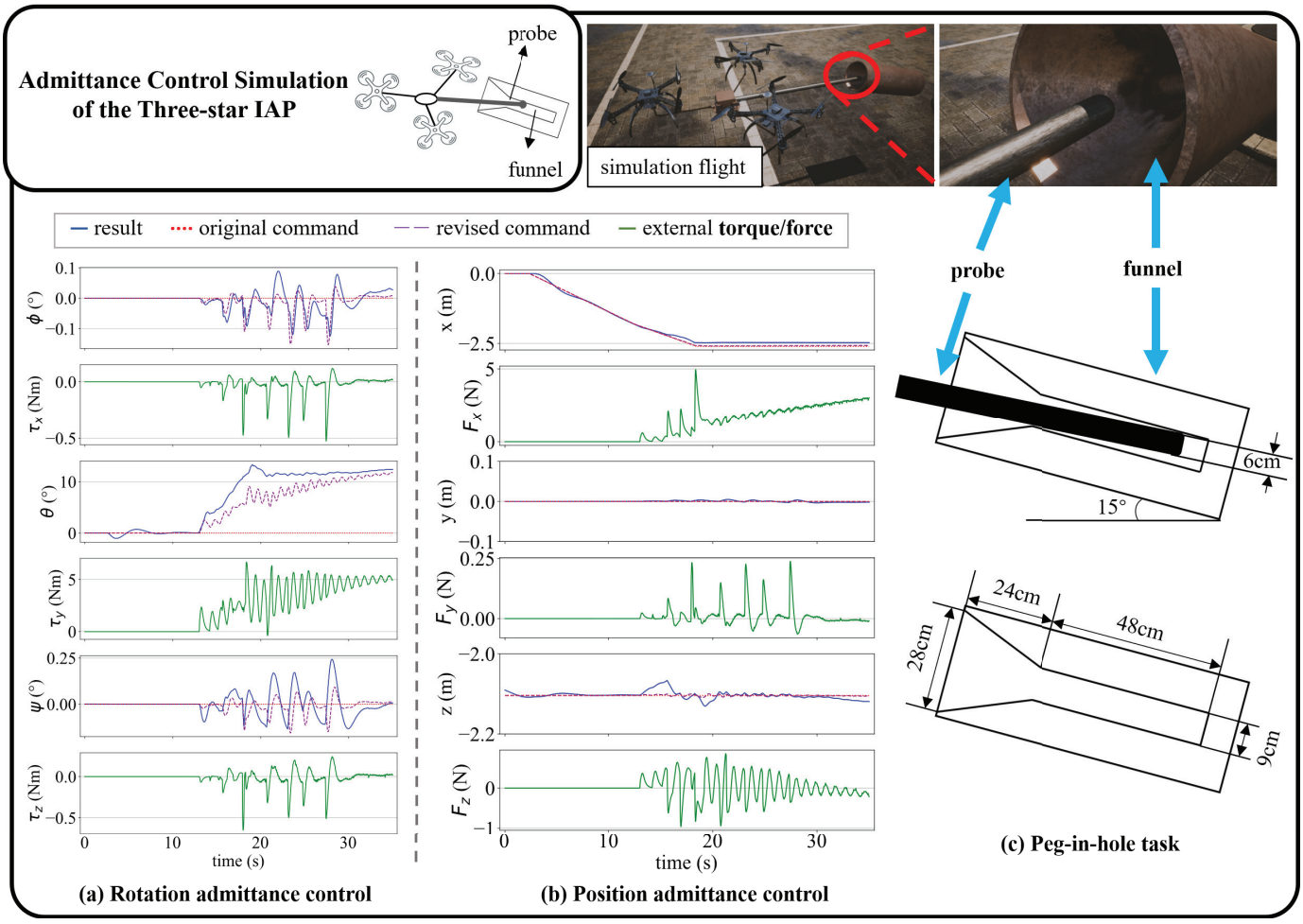


Fig. 15. Admittance control simulation of the three-star IAP.

The results illustrate that, under the influence of the admittance controller, the pitch angle command undergoes gradual adjustments and eventually stabilizes at 12°. Additionally, owing to the existence of multiple contact points between the pole and the funnel, abrupt changes in contact force and torque result in oscillatory effects. These simulation outcomes align with theoretical expectations, validating the capability of our simulator to simulate the external forces and moments generated during contact, thus facilitating the simulation of aerial manipulation tasks for IAPs.

F. Direct Wrench Control Comparison of the Three-Star IAP

This section presents a comparative analysis of direct wrench-pose control for a three-star IAP conducted through both real outdoor flight testing and simulation. The primary objective is to manipulate the IAP, utilizing the direct wrench controller, to maintain flight pose while establishing contact with a wall and generating a desired six-dimensional wrench on the IAP.

In the real outdoor flight testing, the same three-star IAP utilized in VI-A is employed, with a centrally mounted pole along the negative x-axis. To enhance the contact area with the surface and prevent intermittent contact, a rough plate is installed at the end of the pole. In contrast, the simulated IAP solely mounts a pole of equal length without the additional plate, as simulation tends to produce sharp force spikes attributed to variations in contact point numbers, a phenomenon not observed in real flight experiments. For real flight experiments, the wrench estimation method proposed by Zhao et al. is employed to estimate the external wrench acting on the center of the IAP for control feedback [64]. In simulation, adjustments are made to the properties of the contact material to approximate the contact forces generated in actual experiments. The six-dimensional wrenches directly generated from the simulation are then utilized for control feedback.

During the experiment, the IAP is maneuvered along the pole using the baseline motion controller until contact is established with the surface. Subsequently, the direct wrench controller is activated to generate the desired external force ([1.5; 4; 0]N) and torque ([0; 0; -4]Nm) on the IAP.

Both the real outdoor flight and simulation employ identical direct wrench control parameters, as listed in TABLE V. The experimental results, depicted in Fig. 16, showcase the forces and torques generated by the direct wrench controller (a, b) and the deviations between the achieved and desired pose (c, d) for both the real-flight and simulated experiments.

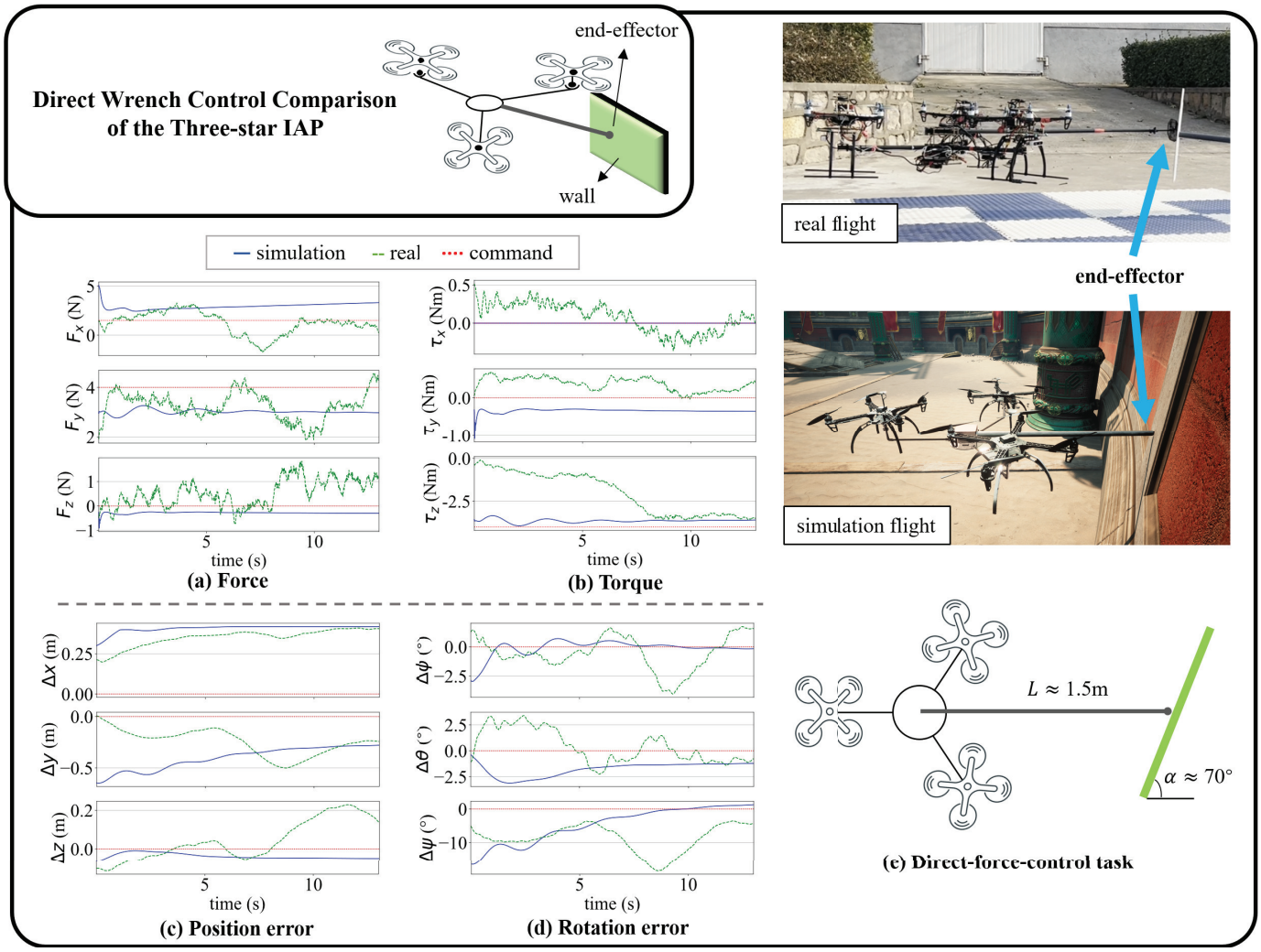


Fig. 16. Direct wrench control comparison of the three-star IAP.

TABLE V
PARAMETERS OF THE THREE-STAR IAP DIRECT-WRENCH CONTROLLER

	x	y	z	ϕ	θ	ψ
$\boldsymbol{k}^{\mathrm{P}}$	7	6	6.5	4.5	4.5	3
$\boldsymbol{k}^{\mathrm{I}}$	0.02	0.02	0.8	1.1	1.1	1.2
$\boldsymbol{k}^{\mathrm{D}}$	20	15	20	6.2	6.0	6.5
$\boldsymbol{k}_w^{\mathrm{P}}$	1	1	1	0.5	0.5	0.5
$\boldsymbol{k}_w^{\mathrm{I}}$	0.1	0.1	0.1	0.05	0.05	0.05
$\boldsymbol{k}_w^{\mathrm{D}}$	2	2	2	1	1	1

Due to constraints imposed by the contact surface, the IAP is unable to reach the desired position in the x direction, leading to steady-state position and force errors. However, a notable similarity is observed in the trends of force or torque errors with respect to position or angle errors along each axis between the real-flight and simulated experiments. Additionally, when force or torque errors align along a particular axis, corresponding position or angle errors also exhibit similar alignment, and vice versa.

Although the complexity of contact simulation mechanisms presents challenges in fully replicating real-world scenarios in simulation, the analysis of similar trend variations validates the capability of our simulator to effectively simulate the aerial operation process of IAPs in the presence of contact forces.

G. Visual-Inertial Localization Fusion Simulation of the Three-Star IAP

This section presents the simulation results of tightly coupled SLAM localization for the three-star IAP. The experiment aims to apply our proposed tightly-coupled SLAM theory, analyze the simulation results, and validate the feasibility of our simulator for IAP tightly-coupled SLAM.

In this experiment, we utilize the same parameters for the three-star IAP in the simulator as those used in the control experiments outlined in Section VI-A. The same trajectory tracking controller serves as the baseline controller, with identical controller parameters. Due to computational constraints, data collection for each experimental set is divided into four sessions to ensure a sufficiently high frame rate of image input for the SLAM algorithm. Initially, motion time series and sensor data (200Hz) are recorded, followed by three sessions of trajectory replay to capture stereo camera images of each sub-aircraft (approximately 20Hz, with a

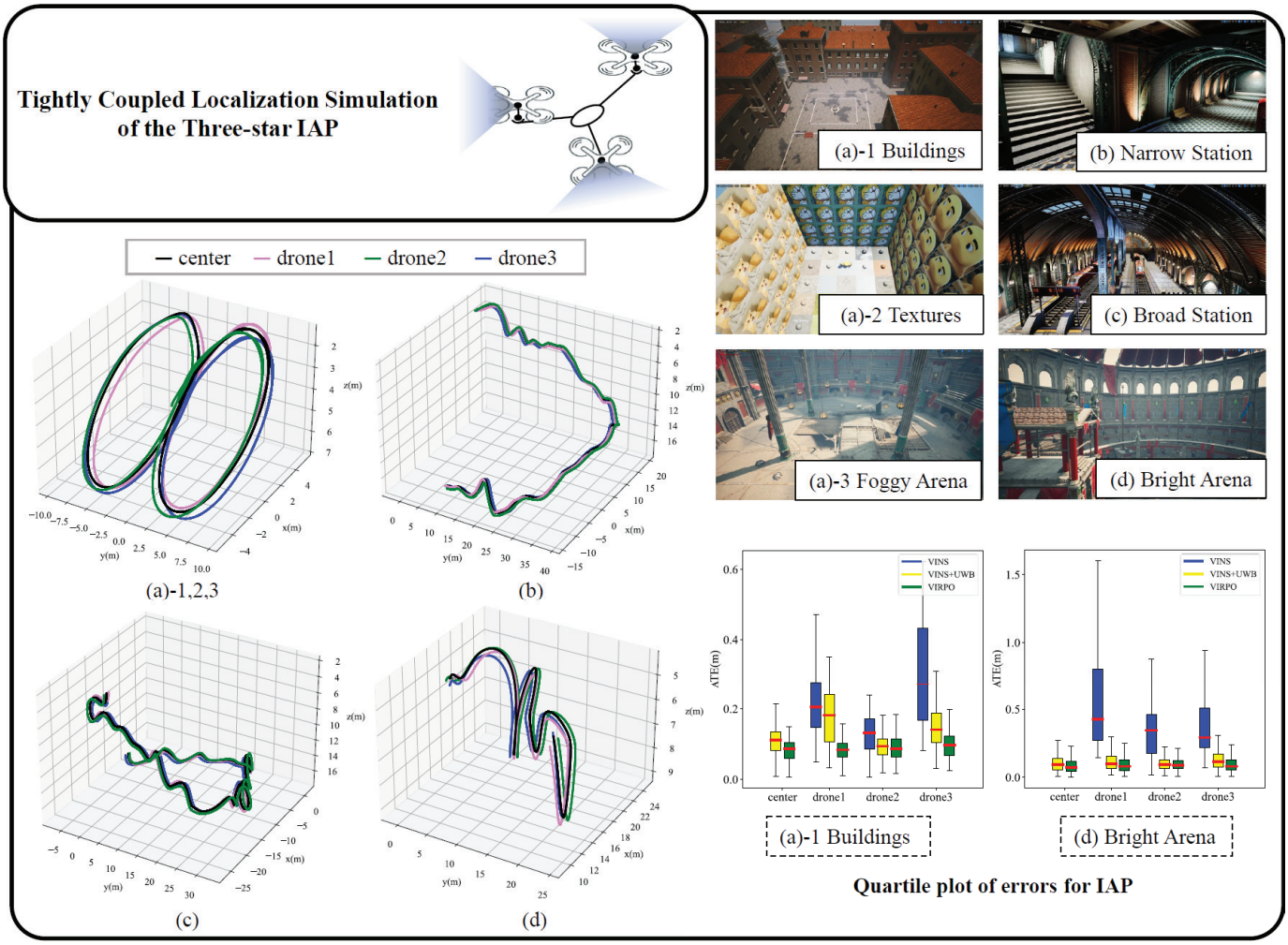


Fig. 17. Tightly coupled SLAM localization of the three-star IAP.

monocular resolution of 640*480). Each frame of the images is timestamped to synchronize with IMU messages based on the trajectory timestamps. Subsequently, all data are integrated based on the timestamps, and VINS, VINS+UWB, and VIRPO localization experiments are conducted for each sub-aircraft to obtain localization results. The center position of the IAP is calculated based on known configuration information. The IMUs of each sub-aircraft are positioned at the center of mass of each aircraft. The centers of the two image planes of the stereo cameras of each sub-aircraft are spaced 12cm apart and positioned 19.8cm in front of the corresponding IMU, with no image distortion.

Remark 2: In this simulation, the real-time acquisition of images by the Unreal Engine incurs a relatively high cost in computing resources, while the dynamics simulation and VIRPO programs consume fewer resources. In the same configuration, the more images that are acquired simultaneously, the lower the image frame rate. If six images from three sub-aircraft are acquired simultaneously, the frequency will drop below 10Hz. Therefore, after careful consideration, we choose to collect two image data from one sub-aircraft at a time. The image data is collected in three separate sessions and then synchronized.

TABLE VI
PARAMETERS OF IMU AND UWB NOISES

r_a^ω	r_b^ω	r_a^a	r_b^a	r_a^p
0.0212	1.4×10^{-5}	0.2828	1.4×10^{-3}	0.7071

TABLE VII
COMPARISON OF RMSE VALUES OF ATE (M) ON DIFFERENT ANCHOR NUMBERS, USING SINGLE-AIRCRAFT LOCALIZATION

number of anchors	0	1	2	3	4
RMSE	0.66	0.53	0.50	0.19	0.13

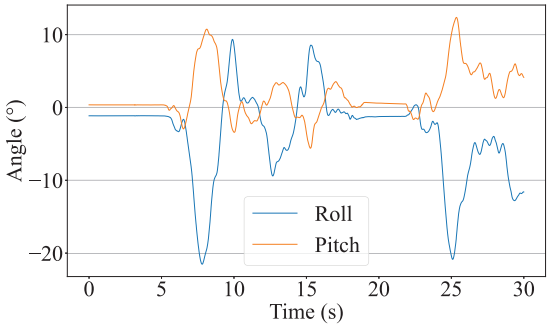
The noise parameters for each IMU and UWB used in each experiment are consistent and modeled in V-D, as shown in TABLE VI.

The attitude inputs of the algorithms in the simulation are estimated by the EKF and closely resemble the noise from IMU in the Pixhawk hardware, as depicted in Fig. 18. Absolute trajectory error (ATE) is employed to evaluate the localization estimation results. Initially, single-aircraft data are used to assess the impact of the number of UWB anchors on localization results. VINS is run to obtain baseline localization

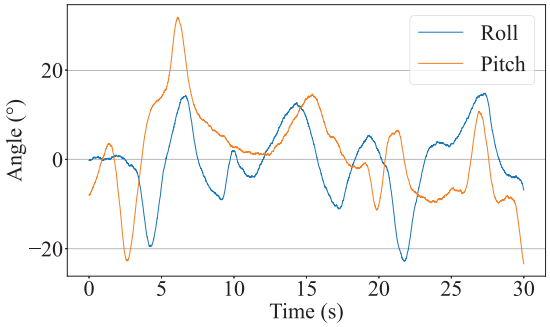
TABLE VIII

COMPARISON OF RMSE VALUES OF ATE (M) ON VARIOUS SIMULATION SCENARIOS

	drone1			drone2			drone3			center	
	VINS	VINS+UWB	VIRPO	VINS	VINS+UWB	VIRPO	VINS	VINS+UWB	VIRPO	VINS+UWB	VIRPO
(a)-1	0.276	0.193	**0.093**	0.139	0.103	**0.091**	0.329	0.163	**0.104**	0.118	**0.087**
(a)-2	0.313	0.112	**0.109**	0.33	0.175	**0.107**	0.418	0.158	**0.113**	0.093	**0.089**
(a)-3	0.187	0.098	**0.089**	0.197	0.143	**0.097**	0.203	0.163	**0.132**	0.119	**0.099**
(b)	0.512	0.482	**0.287**	0.631	0.465	**0.34**	0.465	0.402	**0.297**	0.374	**0.245**
(c)	0.219	0.176	**0.146**	0.229	0.174	**0.142**	0.247	0.17	**0.129**	0.137	**0.117**
(d)	0.66	0.133	**0.107**	0.43	0.123	**0.102**	0.45	0.174	**0.114**	0.134	**0.098**



(a) Real roll and pitch angles by a Pixhawk IMU



(b) Simulated roll and pitch angles by EKF

Fig. 18. Comparison between real and simulated attitude sensor data.

TABLE IX

PARAMETERS FOR VIRPO

σ_l	σ_ω	σ_{att}	σ_d
0.04	0.1	0.01	0.05
$\sigma^{\text{continuous}}_{\text{drift,yaw}}$	$\sigma^{\text{continuous}}_{\text{drift,position}}$	$\sigma^{\text{prior}}_{\text{drift,yaw}}$	$\sigma^{\text{prior}}_{\text{drift,position}}$
0.04	0.01	0.1	0.05

results, using 0 anchors, followed by testing the localization results with the addition of 1 to 4 UWB anchors separately. A summary of the experimental results is presented in TABLE VII. Based on the single-aircraft experiment results, we determine that using 4 UWB anchors provides sufficient accuracy in trajectory estimation to serve as a baseline for comparison with VIRPO in subsequent IAP experiments.

Six sets of experiments are conducted, with trajectories, scenes, and quartile graphs of ATE depicted in Fig. 17. The first three sets ((a)-1,2,3) demonstrate trajectories obtained under the influence of the baseline controller for the same helical figure-eight trajectory, captured in different scenes (clusters of buildings, walls with textures, and a foggy arena). In the subsequent three sets, experiments are conducted in three different scenes (a narrow station, a broad station, and a bright arena) using a PlayStation DualSense controller for manual

TABLE X

POSITIONS OF UWB ANCHORS (M) FOR HELICAL FIGURE-EIGHT TRAJECTORY

	anchor 1	anchor 2	anchor 3	anchor 4
x	-8.17	2.93	8.76	-4.48
y	-4.35	-6.65	6.12	1.17
z	1.38	3.3	1.59	1.14

TABLE XI

POSITIONS OF UWB ANCHORS (M) FOR MANUAL FLIGHT TRAJECTORIES

	anchor 1	anchor 2	anchor 3	anchor 4
x	-38.17	32.93	38.76	-34.48
y	-34.35	-36.65	46.12	31.17
z	1.38	3.3	1.59	1.14

IAP flight control, and trajectories are recorded. Experimental results are summarized in TABLE VIII, with the parameters used in VIRPO presented in TABLE IX. In TABLE IX, the selection of σ_l in VIRPO is based on manual measurement errors in the IAP hardware joints, with no errors introduced in the simulation. σ_ω is chosen based on IMU angular velocity errors, σ_{att} is determined based on attitude standard deviation, σ_d is selected based on UWB standard deviation, and drifts are chosen based on VIO drift results. For VINS+UWB and VIRPO, to compute the center position, the VINS coordinate systems of each sub-aircraft are unified using the ground truth of the first frame data in the simulation. The positions of UWB anchors in the inertial frame $\mathcal{F}_w$ during the experiment are presented in TABLE X and TABLE XI. Anchor positioning is based on including the motion trajectory within the lines connecting the anchors. The same set of anchor positions is used for the helical figure-eight trajectory, while a different set is used for the three manual flight tests.

The experiment trajectories exhibit diversity, with results indicating that under UWB absolute positioning, all sub-aircraft exhibit improved localization accuracy compared with using VINS alone. Moreover, with the introduction of physical constraints in the VIRPO algorithm, the precision of both sub-aircraft and center localization is enhanced compared with the baseline VINS+UWB localization experiment, with an average improvement of 24.07% and a maximum of 32%. Notably, when a certain sub-aircraft exhibits lower accuracy before tight coupling, tightly-coupled SLAM localization could constrain the improvement in the accuracy of the sub-aircraft with higher accuracy, as observed in experiment (a)-1, where sub-aircraft 1's accuracy increases by 51.8%, while sub-aircraft 2's only increases by 11.7% after tight coupling. Similar results are observed in other group experiments, consistent with the expectations of tight coupling theory. Additionally, due to generally more aggressive maneuvers during manual

TABLE XII
SUMMARY OF SIMULATION REQUIREMENTS AND SETUPS FOR EACH RL STAGE

Feature	Offline RL (Expert Data Collection)	Online RL (Fine-Tuning with PPO)
State Updates	Precomputed from expert policy	Real-time state transitions based on RL control inputs
Control Type	Deterministic (rule-based expert controller)	Stochastic (PPO with Gaussian exploration)
Policy Initialization	MLP, random parameters	Pretrained network by TD3BC
Environment Randomization	Fixed initial conditions	Randomized starting positions and orientations
Exploration	Not needed (follows expert policy)	logσ adaptation for PPO
Parallel Execution	Not needed (offline dataset)	three-threaded parallel simulation

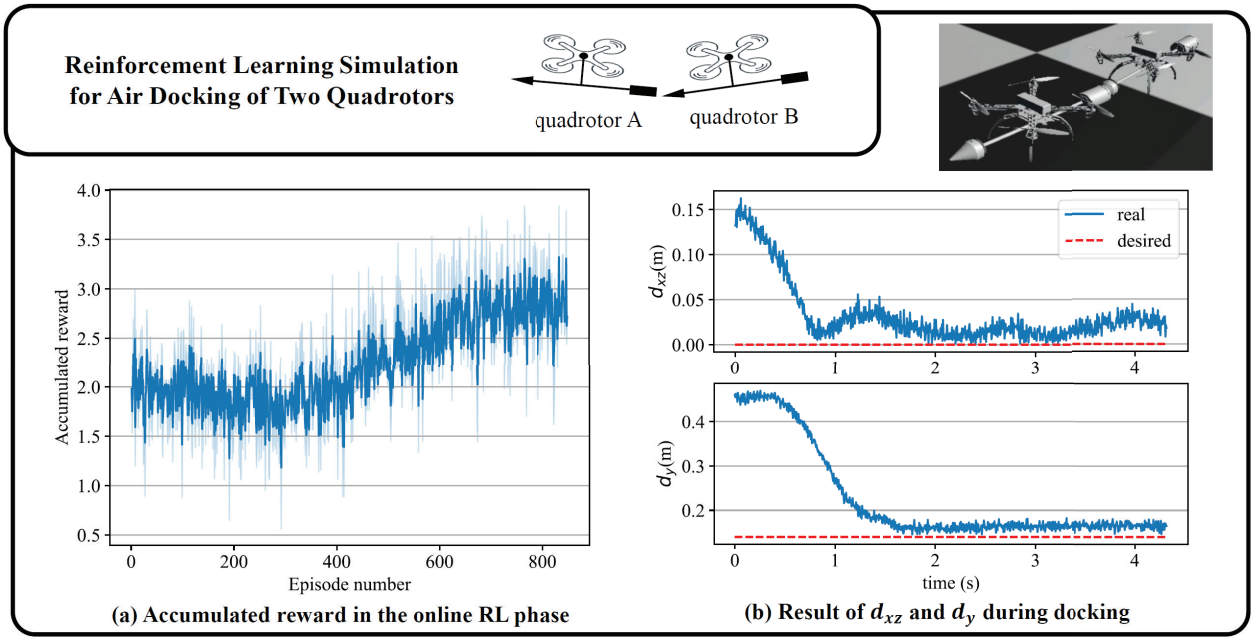


Fig. 19. Reinforcement learning simulation for air docking of two quadrotors.

flight, resulting in higher speeds, the VINS accuracy in manual flight tests generally lags behind that in the helical figure-eight trajectory.

These simulation results of tightly coupled SLAM localization for the three-star IAP align with theoretical expectations and previous experimental findings. Building upon the validation of the feasibility of control simulation, this section verifies the suitability of our simulator for SLAM and tightly-coupled SLAM localization algorithm validation, demonstrating good real-time data performance, synchronization, and sufficiently high-quality image data generation, alongside feasible sensor data.

H. Reinforcement Learning Air Docking Simulation

In this section, we demonstrate the training and validation of air docking for two quadrotors using the offline-to-online RL strategy on our simulation platform. The training is conducted on a computer equipped with a 32-core 64-thread CPU and 4 NVIDIA GeForce RTX 2080 Ti GPUs. Both in the expert data collection phase and online reinforcement learning phase, RaiSim is utilized as the physics engine, and the AquaML framework is employed for parallel training of the task.[1] Each simulation process runs at 200Hz, corresponding to 200 steps per second. The quadrotor model used is the same DJI F450 model as in the previous experimental sections. During the initialization phase of the training scene, quadrotor A serves as the origin, and the relative initial position of quadrotor B is within the following random range (23):

$$\begin{cases} x_B \sim [-0.15, 0.15]\text{m} \\ y_B \sim [-1.5, -1.3]\text{m} \\ z_B \sim [4.85, 5.15]\text{m} \end{cases} \tag{23}$$

Gaussian noises are added to the pose, velocity, angular velocity, and acceleration data in both the training and testing environments. However, due to spikes in force data resulting from surface-to-surface contact in RaiSim, no extra noises are introduced to force and torque data. In Table XII, there is a summary of simulation requirements and setups for offline and online RL stages.

After initialization, the two drones approach each other and perform docking. After docking completion, the combined IAP hovers for 1 second before the task ends. Task illustration, reward variation, and docking test results are depicted in Fig. 19.

During training, the success rate of docking expert data collected using direct force control is approximately 40%,

[1]https://github.com/BIT-aerial-robotics/AquaML/tree/2.1.11

with a relatively long process time. After offline reinforcement learning with TD3-BC, the docking success rate increases to around 65%, with the processing time shortened to 980 steps (4.9s). Finally, after online training with PPO, the reward further increases, and the success rate reaches 95%, with the docking time reduced to 890 steps (4.45s). Due to the favorable results achieved during the offline training phase, the increase in reward after online training is not significant, as shown in Fig. 19 (a). Additionally, during the validation phase, variations in axial and radial errors between the two structures during the docking process are depicted in Fig. 19 (b). According to Fig. 6, upon completion of docking, the radial error d_{xz} should ideally be 0, while the axial error d_y corresponds to the inherent thickness of the cylindrical structure at the tip of the conical structure.

These simulation results demonstrate that our simulator can be utilized for training and validating reinforcement learning strategies for air docking with contact force interaction.

VII. Conclusion

This paper presents a highly realistic simulator framework designed to support aerial operations tasks for Integrated Aerial Platforms (IAPs). Building upon the baseline motion controllers for IAPs, methods such as impedance control and direct wrench control are proposed to enhance compliance during interaction with the environment and precise control capability over operational objects. Through simulation verification of trajectory tracking, interaction control, tightly coupled SLAM positioning under physical constraints, and offline-to-online reinforcement learning for air docking, we demonstrate the feasibility and fidelity of our simulator framework in terms of aerial operation control, perception, and learning for IAPs. Notably, the consistency between our simulation results and actual flight test results further confirms the accuracy and reliability of the simulation platform, underscoring its utility as a powerful tool for addressing practical flight operation challenges. Our work provides significant support for advancing research and applications in the field of IAPs.

Despite promising results, our study has several limitations that could be addressed in future research. Challenges remain in simulating a larger number of IAPs in parallel, particularly the potential decrease in rendering frequency caused by the increased number of virtual camera images. In the future, these issues could be addressed through techniques such as UE5 optimization and GPU acceleration. Additionally, on the basis of this framework, future work could extend to testing and validating IAPs with various joint combinations, such as hinge joints and spherical joints. Finally, efforts could be made to make the framework compatible with ROS2 to facilitate algorithm testing based on ROS2.

References

[1] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1957–1964, Jul. 2018.

[2] V. Lippiello, G. A. Fontanelli, and F. Ruggiero, "Image-based visual-impedance control of a dual-arm aerial manipulator," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1856–1863, Jul. 2018.

[3] S. A. Emami and A. Banazadeh, "Simultaneous trajectory tracking and aerial manipulation using a multi-stage model predictive control," *Aerosp. Sci. Technol.*, vol. 112, May 2021, Art. no. 106573. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963821000845

[4] K. Zhang et al., "Aerial additive manufacturing with multiple autonomous robots," *Nature*, vol. 609, no. 7928, pp. 709–717, 2022.

[5] X. Meng et al., "Contact force control of an aerial manipulator in pressing an emergency switch process," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2107–2113.

[6] M. Xu, A. Hu, and H. Wang, "Image-based visual impedance force control for contact aerial manipulation," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 1, pp. 518–527, Jan. 2023.

[7] D. Tzoumanikas, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, "Aerial manipulation using hybrid force and position NMPC applied to aerial writing," in *Proc. Robot., Sci. Syst. XVI*, Corvalis, OR, USA, Jul. 2020, pp. 1–10, doi: .

[8] Y. Yu, K. Wang, R. Guo, V. Lippiello, and X. Yi, "A framework to design interaction control of aerial slung load systems: Transfer from existing flight control of under-actuated aerial vehicles," *Int. J. Syst. Sci.*, vol. 52, no. 13, pp. 2845–2857, Oct. 2021, doi: 10.1080/00207721.2021.1909777.

[9] C. Shi, G. Lai, Y. Yu, M. Bellone, and V. Lippiello, "Real-time multimodal active vision for object detection on UAVs equipped with limited field of view LiDAR and camera," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6571–6578, Oct. 2023.

[10] J. P. Queralta et al., "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191617–191643, 2020.

[11] J. Sandino, F. Vanegas, F. Gonzalez, and F. Maire, "Autonomous UAV navigation for active perception of targets in uncertain and cluttered environments," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–12.

[12] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annu. Rev. Control*, vol. 46, pp. 165–180, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578818300932

[13] Y. Yu, C. Shi, D. Shan, V. Lippiello, and Y. Yang, "A hierarchical control scheme for multiple aerial vehicle transportation systems with uncertainties and state/input constraints," *Appl. Math. Model.*, vol. 109, pp. 651–678, Sep. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X22002268

[14] H. Nguyen, S. Park, J. Park, and D. Lee, "A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 353–369, Apr. 2018.

[15] C. Shi and Y. Yu, "Design and implementation of a fully-actuated integrated aerial platform based on geometric model predictive control," *Micromachines*, vol. 13, no. 11, p. 1822, Oct. 2022. [Online]. Available: https://www.mdpi.com/2072-666X/13/11/1822

[16] J. Sun, Y. Yu, and B. Xu, "Towards flying carpet: Dynamics modeling, and differential-flatness-based control and planning," in *Cognitive Systems and Information Processing*, F. Sun, A. Cangelosi, J. Zhang, Y. Yu, H. Liu, and B. Fang, Eds., Singapore: Springer, 2023, pp. 351–370.

[17] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Springer Proceedings in Advanced Robotics*. Cham, Switzerland: Springer, 2018, pp. 621–635.

[18] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Oct. 2004, pp. 2149–2154.

[19] G. Marques, D. Sherry, D. Pereira, G. Marques, D. Sherry, and D. Pereira, *Elevating Game Experiences With Unreal Engine 5: Bring Your Game Ideas to Life Using the New Unreal Engine 5 and C++*. Birmingham, U.K.: Packt Publishing, 2022.

[20] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 4006–4013.

[21] M. Ryll et al., "6D interaction control with aerial robots: The flying end-effector paradigm," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1045–1062, Aug. 2019, doi: 10.1177/0278364919856694.

[22] K. Bodie et al., "Active interaction force control for contact-based inspection with a fully actuated aerial vehicle," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 709–722, Jun. 2021.

[23] M. Brunner, G. Rizzi, M. Studiger, R. Siegwart, and M. Tognon, "A planning-and-control framework for aerial manipulation of articulated objects," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10689–10696, Oct. 2022.

[24] A. Praveen, X. Ma, H. Manoj, V. LN. Venkatesh, M. Rastgaar, and R. M. Voyles, "Inspection-on-the-fly using hybrid physical interaction control for aerial manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 1583–1588.
[25] S. Park et al., "ODAR: Aerial manipulation platform enabling omnidirectional wrench generation," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 4, pp. 1907–1918, Aug. 2018.
[26] P. Zheng, X. Tan, B. B. Kocer, E. Yang, and M. Kovac, "TiltDrone: A fully-actuated tilting quadrotor platform," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6845–6852, Oct. 2020.
[27] M. Ryll, D. Bicego, and A. Franchi, "Modeling and control of FAST-hex: A fully-actuated by synchronized-tilting hexarotor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1689–1694.
[28] L. Ruan, C.-H. Pi, Y. Su, P. Yu, S. Cheng, and T.-C. Tsao, "Control and experiments of a novel tiltable-rotor aerial platform comprising quadcopters and passive hinges," *Mechatronics*, vol. 89, Feb. 2023, Art. no. 102927. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415822001453
[29] Y. Su, P. Yu, M. J. Gerber, L. Ruan, and T.-C. Tsao, "Fault-tolerant control of an overactuated UAV platform built on quadcopters and passive hinges," *IEEE/ASME Trans. Mechatronics*, vol. 29, no. 1, pp. 1–12, Feb. 2024.
[30] Y. Su et al., "Downwash-aware control allocation for over-actuated UAV platforms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 10478–10485.
[31] T. Nishio, M. Zhao, K. Okada, and M. Inaba, "Design, control, and motion planning for a root-perching rotor-distributed manipulator," *IEEE Trans. Robot.*, vol. 40, pp. 660–676, 2024.
[32] M. Zhao, K. Okada, and M. Inaba, "Versatile articulated aerial robot DRAGON: Aerial manipulation and grasping by vectorable thrust control," *Int. J. Robot. Res.*, vol. 42, nos. 4–5, pp. 214–248, Apr. 2023, doi: 10.1177/02783649221112446.
[33] Y. Su et al., "Sequential manipulation planning for over-actuated unmanned aerial manipulators," in *Proc. IROS*, 2023, pp. 6905–6911.
[34] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[35] E. Coumans and Y. Bai. (2016). *Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*. [Online]. Available: http://pybullet.org
[36] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51416–51431, 2021.
[37] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 895–902, Apr. 2018. [Online]. Available: http://www.raisim.com
[38] NVIDIA.(2021). *ISAAC Sim*. [Online]. Available: https://developer.nvidia.com/isaac-sim
[39] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS—A Modular Gazebo MAV Simulator Framework*. Cham, Switzerland: Springer, 2016, pp. 595–625, doi: 10.1007/978-3-319-26054-9_23.
[40] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.
[41] M. Jacinto et al., "Pegasus simulator: An ISAAC sim framework for multiple aerial vehicles simulation," in *Proc. Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, Jun. 2023, pp. 917–922.
[42] M. Zhang, M. Li, K. Wang, T. Yang, Y. Feng, and Y. Yu, "Zero-shot sim-to-real transfer of robust and generic quadrotor controller by deep reinforcement learning," in *Cognitive Systems and Information Processing*, F. Sun, Q. Meng, Z. Fu, and B. Fang, Eds., Singapore: Springer, 2024, pp. 27–43.
[43] C. A. Dimmig et al., "Survey of simulators for aerial robots: An overview and in-depth systematic comparisons," *IEEE Robot. Autom. Mag.*, early access, 2024.
[44] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4CV: A photo-realistic simulator for computer vision applications," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 902–919, Sep. 2018.
[45] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proc. Conf. Robot Learn.*, Jan. 2020, pp. 1147–1157.
[46] A. Juliani et al., "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.
[47] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 6941–6948.
[48] X. Dai, C. Ke, Q. Quan, and K.-Y. Cai, "RFlySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations," *Aerosp. Sci. Technol.*, vol. 114, Jul. 2021, Art. no. 106727. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963821002376
[49] D. Lee and P. Y. Li, "Passive decomposition of mechanical systems with coordination requirement," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 230–235, Jan. 2013.
[50] N. Hogan, "Impedance control: An approach to manipulation," in *Proc. Amer. Control Conf.*, San Diego, CA, USA, Jun. 1984, pp. 304–313.
[51] Y. Fan et al., "Tight fusion of odometry and kinematic constraints for multiple aerial vehicles in physical interconnection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 3891–3897.
[52] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[53] N. V. Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, p. 1363, Aug. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/9/1363
[54] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013, *arXiv:1312.6211*.
[55] L. Villani, *Hybrid Force and Position Control*. Berlin, Germany: Springer, 2020, pp. 1–6, doi: 10.1007/978-3-642-41610-1_110-1.
[56] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID
[57] Y. Feng, T. Yang, and Y. Yu, "Enhancing UAV aerial docking: A hybrid approach combining offline and online reinforcement learning," *Drones*, vol. 8, no. 5, p. 168, Apr. 2024. [Online]. Available: https://www.mdpi.com/2504-446X/8/5/168
[58] A. Beeson and G. Montana, "Improving TD3-BC: Relaxed policy constraint for offline learning and stable online fine-tuning," 2022, *arXiv:2211.11802*.
[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[60] Y. Feng, C. Shi, J. Du, Y. Yu, F. Sun, and Y. Song, "Variable admittance interaction control of UAVs via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 1291–1297.
[61] T. H. Nguyen, T.-M. Nguyen, and L. Xie, "Tightly-coupled single-anchor ultra-wideband-aided monocular visual odometry system," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 665–671.
[62] G. Lai, C. Shi, K. Wang, Y. Yu, Y. Dong, and A. Franchi, "Multi-agent visual-inertial localization for integrated aerial systems with loose fusion of odometry and kinematics," *IEEE Robot. Autom. Lett.*, vol. 9, no. 7, pp. 6504–6511, Jul. 2024.
[63] G. Lai, Y. Yu, F. Sun, J. Qi, and V. Lippiello, "Efficiently kinematic-constraint-coupled state estimation for integrated aerial platforms in GPS-denied environments," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2838–2845, Mar. 2025.
[64] F. Shi, M. Zhao, T. Anzai, X. Chen, K. Okada, and M. Inaba, "External wrench estimation for multilink aerial robot by center of mass estimator based on distributed IMU system," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1891–1897.

Jianrui Du received the B.S. degree in detection, guidance, and control technology and the M.S. degree in mechanical engineering from Beijing Institute of Technology, Beijing, China, in 2020 and 2023, respectively, where he is currently pursuing the Ph.D. degree with the School of Mechatronical Engineering. His research interests include the planning and control of aerial robotic systems.

Kaidi Wang received the B.S. and M.S. degrees in mechanical engineering from Beijing Technology and Business University, Beijing, China, in 2016 and 2020, respectively. He is currently pursuing the Ph.D. degree in mechanical engineering with Beijing Institute of Technology, Beijing. His research interests include the design, modeling, and control of new aerial robotic systems.

Yingjun Fan received the bachelor's degree in detection, guidance, and control technology from Beijing Institute of Technology, Beijing, China, in 2022, where he is currently pursuing the master's degree. His current research interests include localization, and simultaneous localization and mapping.

Ganghua Lai (Graduate Student Member, IEEE) received the B.S. degree in mechanical engineering from Beijing Jiaotong University, Beijing, China, in 2021. He is currently pursuing the Ph.D. degree in mechanical engineering with Beijing Institute of Technology, Beijing. His research interests include the visual SLAM, multi-robot systems, and multi-modal sensor fusion.

Yushu Yu received the B.S., M.S., and Ph.D. degrees in mechanical engineering from Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, in 2007, 2010, and 2013, respectively. From 2013 to 2014, he was an Engineer at China Aerospace Science and Industry Corporation. From 2014 to 2019, he conducted research at BUAA; Nanyang Technological University, Singapore; and the Chalmers University of Technology, Sweden. He is currently an Associate Professor at Beijing Institute of Technology, Beijing. His research interests include the control and perception of aerial systems. He has received multiple best paper awards at international conferences and the Excellent Doctoral Dissertation Award from BUAA. He serves as an associate editor for several journals and conferences.