IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.
Cite as T-ASE paper.

SIGMA: An Agent-Based Modeling UAV Swarm Simulator for Swarm Intelligence Algorithms

Sheng Zhang, Juan Li, Chang Liu, Lei Fu, Zehao Bai, and Jie Li

***Abstract*—Swarm intelligence for uncrewed aerial vehicles (UAVs) significantly improves the success rate of executing intricate tasks using "distributed platforms and aggregated effects". However, the high experimental costs and safety risks constrain its development. This paper introduces SIGMA (Swarm Intelligence Generic simulator for Multi-UAVs), a high-fidelity distributed UAV swarm simulator for swarm intelligence algorithms. As an agent-based modeling simulator (ABMS), SIGMA has three key innovations: First, an automatic model tuning method improves aircraft dynamics fidelity. Second, a bidirectional discrete-event simulation (BiDES) architecture resolves the time alignment challenges in distributed systems. Third, a multi-agent learning toolbox ensures algorithm compatibility via an episodic training structure and a memory replay mechanism. In the verification part, the fidelity and scalability of the simulator are verified by quantitative simulations and experiments, and several successful applications demonstrate the practicality of the proposed simulator.**

***Notice to Practitioners*—The motivation for this paper stems from the need to develop a scalable and high-fidelity simulator for practical applications of swarm intelligence algorithms. Simulators developed based on game engines are widely used in swarm robotics due to their realistic 3D environments. However, as the number of nodes increases, the real-time performance and scalability of the simulators will decrease significantly. To achieve real-time simulation of large-scale swarms and improve swarm fidelity, SIGMA is proposed. It uses ABMS technology to achieve better performance and can effectively fit the learning tasks of UAV swarms.**

***Index Terms*—Agent-based modeling and simulation, UAV Swarm, swarm intelligence, discrete-event simulation**

I. Introduction

SWARM intelligence in uncrewed aerial vehicles (UAVs) is inspired by biological collective [1], where individual behaviors are governed by simple rules and local information, giving rise to complex and efficient collective behavior [2]. Swarm intelligence algorithms, such as evolutionary algorithms [3], particle swarm optimization [4], and multi-agent reinforcement learning [5]–[8], use multiple rounds to improve UAV coordination and decision-making. Through competition and selection, the swarm is able to exhibit high-quality collective performance. However, the high costs and risks of experiments limit the development efficiency of swarm intelligence algorithms. To address this challenge, the development of high-fidelity and scalable swarm simulation platforms has become a necessary trend [9].

UAV swarm simulators are critical tools in modern UAV research, enabling the modeling of complex scenarios involving dynamics, communication, navigation, and task execution in coordinated formation flight. Widely adopted simulators include Gazebo [10], AirSim [11], Webots [12], ARGoS [13], PyFlyt [14], FlightGoggles [15], FastSim [16], RFlySim [17]. The characteristics of these simulators are presented in TABLE I. Gazebo, Webots, and ARGoS are well-integrated with ROS and offer modular, open-source architectures. However, they require significant programming effort and offer limited 3D rendering performance. AirSim, RFlySim, and FastSim use game engines to provide photo-realistic 3D environments, making them effective for vision-based research, but they require high hardware and necessitate customized interface development. To enhance the scalability of high-fidelity simulation, distributed architectures are being increasingly adopted by modern simulation systems.

Distributed architectures balance computational loads and support scalable node deployment, making them ideal for swarm simulations. Platforms like RFlySim, Isaac Sim [18], Gazebo, and ARGoS use these architectures to extend the node count limit. However, these platforms also have limitations. First, they rely on high-overhead protocols such as RPC or ROS, which result in heavy communication loads. Second, they employ Lock-Step alignment, meaning that any node failure or delay can lead to global crashes or frame drops.

Agent-Based Modeling and Simulation(ABMS) [19] is well-suited for swarm intelligence algorithms because of its bottom-up modeling and adaptive interactions. The distributed ABMS architecture, where each node operates independently and communicates through a hybrid peer-to-peer (P2P) framework [20], is particularly advantageous for enabling large-scale, high-fidelity simulations.

Due to computation frequency discrepancies, distributed architectures can lead to global information in-

This work was supported by the National Natural Science Foundation of China under Grant 62373053 (Corresponding author: Juan Li).

Sheng Zhang, Juan Li, Lei Fu, Zehao Bai, and Jie Li are with the School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China (email: 3120225103@bit.edu.cn; juanli@bit.edu.cn; 3120225143@bit.edu.cn; 3220220088@bit.edu.cn; lijie@bit.edu.cn).

Chang Liu is with the Delta Region Academy of Beijing Institute of Technology, Jiaxing 310025, China (email: 1210494158@qq.com)

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

TABLE I
OVERVIEW OF UAV SWARM SIMULATORS

Name	Engines		Vehicles		Sensors				OS			Reality		Arch.	Nodes
	Physics	Render	Rotor	FW	RGB	Depth	Seg.	LiDar	Win.	Linux	Mac	Model	Photo.		
FlightGear	JSBSim	OSG	✗	✓	✓	✗	✗	✗	✓	✓	✓	High	Low	Cent.	2∼5
XPlane	BET	Vulkan	✗	✓	✓	✗	✗	✗	✓	✗	✗	High	Low	Dist.	20
Gazebo	DART	OGRE2	✓	✓	✓	✓	✗	✓	✗	✓	✗	Low	Low	Dist.	50∼100
Webots	ODE	OpenGL	✓	✗	✓	✗	✗	✓	✓	✓	✓	Middle	Low	Cent.	20∼40
ARGoS	ODE	OGRE1	✓	✗	✓	✓	✓	✓	✗	✓	✓	Low	Low	Dist.	50+
PyFlyt	PyBullet	OpenGL	✓	✓	✓	✓	✗	✗	✓	✓	✗	Middle	Low	Cent.	30∼50
AirSim	PhyX	UE4	✓	✗	✓	✓	✗	✗	✓	✓	✓	Low	Middle	Cent.	30∼50
RFlySim	Matlab	UE4/5	✓	✓	✓	✓	✗	✓	✓	✗	✗	Middle	High	Dist.	50+
FastSim	Flexible	Unity3D	✓	✗	✓	✓	✓	✓	✓	✓	✗	Middle	High	Cent.	50∼100
Issac Sim	PhysX	Omniverse	✓	✓	✓	✓	✗	✓	✓	✓	✗	Low	High	Dist.	256
SIGMA(ours)	JSBSim+	UE4/5	✓	✓	✓	✓	✓	✓	✓	✓	✓	High	High	Dist.	**200∼500**

* *Cent.* represents centralized architecture, *Dist.* represents distributed architecturde.
The node scalability test was conducted on multiple PCs equipped with Intel i7-12700F CPUs, RTX 4070 Ti GPUs, and 64 GB of RAM.

consistencies due to, which may disrupting group interactions. Distributed systems like cloud computing and joint simulations have introduced time alignment techniques such as timestamp synchronization, vector clocks, and discrete-event simulation(DES) [21]. Among them, the optimistic event alignment is one of the DES methods. It allows brief asynchronous operation, preventing a single failure from affecting the entire system [22].

In summary, a simulator for swarm intelligence algorithms should be scalable, high-fidelity, and compatible. This paper proposes a **S**warm **I**ntelligence **G**eneric Simulator for **M**ulti-U**A**V (SIGMA) to provide an efficient training and deduction platform for swarm intelligence algorithms. The main contributions of the paper are as follows:

- *Distributed Simulation Architecture based on ABMS*: The simulator adopts a hybrid P2P computing architecture to ensure stability and real-time performance by sharing computing power. This architecture supports ABMS technology. By independently modeling the behavior of each aircraft, the simulator can capture complex dynamic changes and emergent behaviors to provide accurate simulation results.
- *High-Fidelity Simulation Capability*: To ensure model fidelity, an automatic tuning method for aircraft aerodynamic parameters is proposed, enabling data-driven optimization based on flight log data. For platform fidelity, a hardware-in-the-loop (HIL) simulation strategy is employed to provide a realistic computing environment for embedded algorithm execution, effectively bridging the gap between simulation and real-world deployment.
- *Time Alignment Method*: A BiDES method is designed to solve the time alignment problem of distributed systems. This method employs event-driven and sequence alignment algorithms. The time alignment method deploys the algorithm at the edge instead of the server, ensuring the simulator's scalability.
- *Multi-Agent Learning Toolbox*: To support the training of swarm intelligence algorithms, the system integrates data interfaces such as ROS, ZMQ, and GStreamer within the Unreal Engine, enabling unified scheduling and independent control in distributed systems. This supports the construction of a parallelizable episodic training framework. Additionally, to meet data collection requirements of intelligent algorithms, an experience replay module is implemented using incremental storage and log interpolation strategies.

The remainder of this paper is organized as follows: Section II presents a comparative analysis of the strengths and limitations of existing simulators. Section III elaborates on the proposed distributed simulation architecture based on ABMS techniques. Section IV details the implementation of the SIGMA simulator, including model fine-tuning strategies, the BiDES time alignment method, and a multi-agent learning toolbox. Section V evaluates the scalability and fidelity of the SIGMA through simulation and experimental validation, and provides several successful applications to demonstrate the practicality. Finally, Section VI concludes the paper and outlines potential directions for future research.

II. RELATED WORKS

UAV swarm simulation, as a key technology for low-altitude economies and combat systems, has gained growing attention, prompting the development of various simulators. This section compares SIGMA with other UAV swarm simulators (see TABLE I) and analyzes core components of them.

A. General Robotics Swarm Simulators

Compared to autonomous driving and robotics, UAV swarm simulation is still nascent [23]. Existing work largely relies on general-purpose platforms (e.g., Gazebo, AirSim, Webots, ARGoS, Isaac Sim, Genesis), which, despite offering basic infrastructure, lack UAV-specific aerodynamic fidelity.

Specifically, Gazebo facilitates quick integration of UAV models and sensors, making it suitable for early-stage algorithm testing but suffering from limited fidelity due to engine constraints. Webots performs reliably at small scales but faces latency beyond 30 agents. ARGoS supports large-scale, parallel simulations with low computational overhead, yet its simplified physics reduce realism. NVIDIA Isaac Sim delivers high-fidelity rendering and real-time performance but requires substantial GPU resources and user expertise. Overall, while general-purpose

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

platforms provide flexibility and extensibility, they often fall short in simulating UAV swarms with high fidelity.

B. Photo-Realistic Simulators

To meet Sim2Real requirements, several photorealistic simulators—such as XPlane, FlightGoggles, RFlySim, AirSim, FastSim, MARSim [24] and Sim4CV [25]—have been developed using engines like Unreal and Unity to enhance visual realism for perception and control testing. However, limitations persist: XPlane supports only fixed-wing UAVs with restricted scalability; FlightGoggles and Sim4CV offer low-latency, vision-in-the-loop simulation but lack scalability and physical realism; AirSim and FastSim provide rich visual output and multi-UAV interfaces but degrade beyond 50 agents; RFlySim improves scalability via hardware-in-the-loop design but sacrifices flexibility and affordability. These simulators provide higher 3D realism and sensor accuracy, but usually overlook the software compatibility.

C. Distributed Swarm Simulators

Distributed architectures are an inevitable trend for swarm simulators, with the key challenge lying in time alignment across multiple nodes during computation. Simulators such as Gazebo, Isaac Sim, and ARGoS employ client-server architectures with Lock-Step alignment strategies. Gazebo Ignition leverages ROS2/FastDDS for global time coordination but experiences performance degradation beyond 80 nodes due to communication overhead. ARGoS achieves efficient alignment via simplified models and discrete event simulation, at the expense of fidelity. Isaac Sim enables alignment of up to 256 nodes through GPU-based timestamps and deterministic physics, though its scalability is constrained by hardware demands and deployment complexity. These trade-offs highlight the need for lightweight, low-latency alignment frameworks that balance scalability and accuracy.

In summary, to meet the simulation requirements of UAV swarm algorithms, an ideal simulation platform should possess the following core characteristics: First, it should adopt a **high-fidelity** simulation engine to ensure the accuracy of model construction and scene rendering. Second, it should be designed with a distributed architecture to enable **scalability** for large-scale node simulations. Finally, it should support multiple universal interface protocols to ensure **compatibility** with swarm intelligence algorithms.

III. Distributed Architecture Design

The SIGMA simulator's distributed architecture is shown in Fig.1. The architecture includes four types of modules: interface, physical, rendering, and behavior module. It begins with the *interface module* for global configuration and initialization. After the initial setups and task specifications, other modules within the local network are remotely activated. Each node utilizes independent *physical modules* for high-fidelity simulations. The time alignment server within the *rendering module* ensures inter-node synchronization and delivers environmental data to the *behavior modules*, which utilize internal behavior libraries for decision-making, enabling emergent collective behaviors from individual agent actions. Modules communicate independently over a hybrid P2P network without a central server. The following sections detail the implementation of the four modules.

A. Agent-based Modeling and Simulation Method

ABMS adopts a bottom-up approach to represent complex systems by simulating the behaviors and interactions of individual agents, thereby capturing emergent group dynamics. SIGMA adopts a P2P architecture, where all modules operate independently with predefined data interfaces. To ensure global coordination and computational consistency, a centralized interface module is retained. Each agent comprises standalone physics, rendering, and behavior modules. SIGMA features a modular architecture. To elaborate on the structure shown in Fig.1, the following sections detail the composition and internal operational logic of each module.

1) Interface module: The interface module, depicted in Fig.1, comprises a graphical user interface (GUI) for pre-simulation configuration and an application programming interface (API) for runtime control. The GUI manages agent initialization, environment setup, and platform selection. The API, based on publish-subscribe protocols (e.g., ROS, ZMQ, MQTT), enables real-time access and modification of simulation parameters. Both interfaces interact with submodules via the Setting API Server to ensure parameter synchronization across the system.

The interface module is critical for ensuring global consistency in SIGMA. It utilizes a time step trigger to unify physical module computations at a fixed frequency, while receiving UAV state information from all nodes. The DES alignment server embeds a built-in alignment algorithm. Upon data alignment completion, it triggers an aligned-event and broadcasts the aligned trigger to all modules. Since the simulator is event-driven, SIGMA can be considered a DES system. It may skip certain time steps when there are no events to process. The data flows illustrated in Fig.1 depict the DES methods implementation process, which will be discussed in detail in Section IV.B.

To prevent its overload, the interface module avoids complex computations or rendering tasks, ensuring software stability. It monitors the operation frequency of each node's physical and rendering modules and can adjust the event trigger frequency or restart modules in case of overload or disruptions, maintaining stable simulation performance.

2) Physics module: ABMS emphasizes the self-organization and distributed nature of the swarm. Each aircraft contains a physical module to ensure individual autonomy. The components of the physics module are shown in the blue block of Fig.1. The module features

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

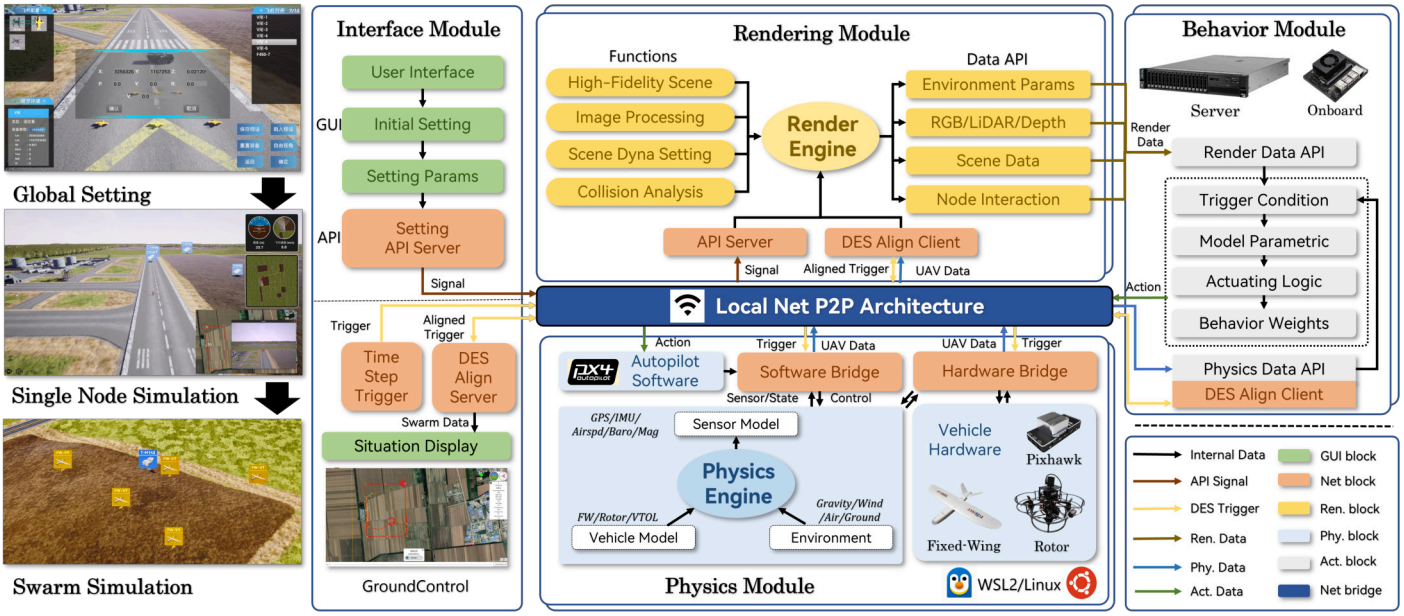


Fig. 1. Architecture of the SIGMA simulator.

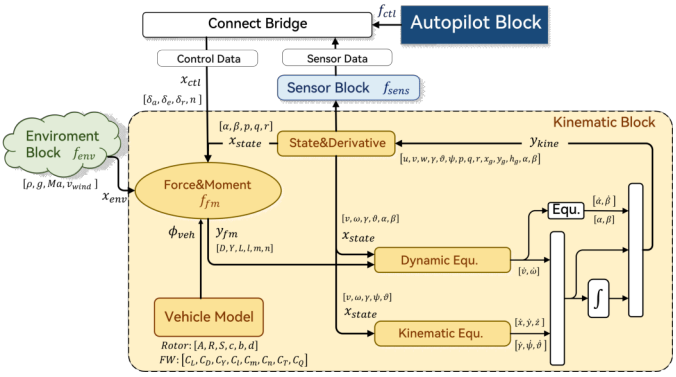


Fig. 2. The structure of general aircraft model in the physical module.

two key components—Hardware Bridge and Software Bridge—acting as interfaces between the autopilot system and the physics engine. Compatible with both Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) simulations, the two modes share a common data flow architecture but differ in the computational platforms and data interfaces. The data flow and computational logic of the physical module are detailed in Fig.2.

The physical module includes three blocks: kinematic block, environmental block, and sensor block. Once the modules above interact with the autopilot firmware, it can effectively simulate the aircraft's control system and flight environment. The mathematical model for this module is as (1):

$$\begin{cases} \boldsymbol{y}_{\text{env}} = \boldsymbol{f}_{\text{env}}(\boldsymbol{x}_{\text{state}}, \phi_{\text{env}}), \\ \boldsymbol{y}_{\text{sens}} = \boldsymbol{f}_{\text{sens}}(\boldsymbol{x}_{\text{state}}, \boldsymbol{y}_{\text{env}}, \phi_{\text{sens}}), \\ \boldsymbol{y}_{\text{ctl}} = \boldsymbol{f}_{\text{ctl}}(\boldsymbol{x}_{\text{state}}, \boldsymbol{x}_{\text{dec}}, \boldsymbol{y}_{\text{sens}}), \\ \boldsymbol{y}_{\text{state}} = \boldsymbol{f}_{\text{kine}}(\boldsymbol{x}_{\text{state}}, \boldsymbol{y}_{\text{ctl}}, \boldsymbol{y}_{\text{env}}, \phi_{\text{veh}}) \end{cases} \tag{1}$$

Where $\boldsymbol{f}_* \triangleq \{\boldsymbol{f}_{\text{env}}, \boldsymbol{f}_{\text{sens}}, \boldsymbol{f}_{\text{kine}}, \boldsymbol{f}_{\text{ctl}}\}$ denotes the functions of the environment, sensor, aerodynamic block and autopilot. $\phi_* \triangleq \{\phi_{env}, \phi_{sens}, \phi_{veh}\}$ denotes the parameters of these blocks. $\boldsymbol{x}_{state}$ contains aircraft motion states (e.g., position, velocity and attitude). $\boldsymbol{x}_{dec}$ denotes the control expected value of the decision module. $\boldsymbol{y}_{env}$ includes the environment variables (e.g., wind speed, gravity) based on the position of the aircraft. $\boldsymbol{y}_{ctl}$ denotes actuator states (e.g., rotor rotating speeds and deflection angles). Finally, the state of the aircraft at the next moment is obtained, denoted as $\boldsymbol{y}_{state}$. The above steps are repeated until the aircraft has completed its entire mission lifecycle. The physical module in SIGMA adopts a general aircraft modeling to capable of supporting multiple UAV types. The details of (1) are provided in Section III.B.

The Physics module supports both SIL and HIL modes. SIL is more suitable for iterative training tasks due to its flexibility and convenience. HIL connects hardware to the simulator, which can evaluate the hardware performance and reliability of the algorithms. In the HIL mode, the behavior module is directly deployed on the onboard computer, which serves as the edge platform and connects the autopilot and other hardware to the physical module via hardware interfaces (e.g., USB, I2C, UART) and communication protocols (e.g., MAVLink, RTPS). More importantly, it allows integration with data link hardware to simulate communication delays.

3) Rendering module: The rendering module integrates a DES alignment client to coordinate swarm states and supports the analysis of swarm interactions within the ABMS framework. The components of the rendering module are shown in the yellow block of Fig.1. Leveraging the aligned trigger value, the client retrieves state variables from all nodes in parallel, thereby ensuring consistency in swarm state representation. The aligned data is visu-

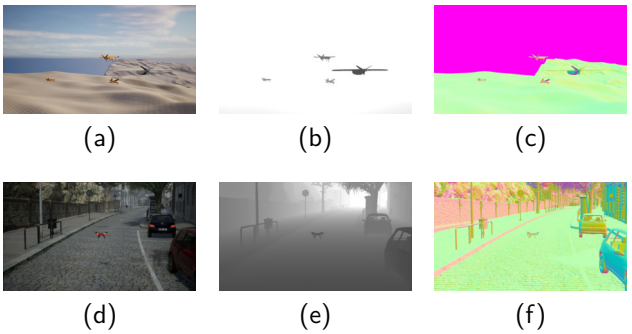


Fig. 3. SIGMA rendering effects: (a), (b), and (c) show the high-altitude scene's RGB, depth, and normal rendering result. (d), (e) and (f) show the city scene's RGB, depth, and normal rendering result.

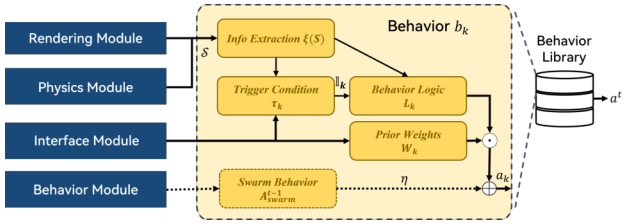


Fig. 4. The structure of the behavior module.

alized in real time using Epic Games®/Unreal Engine 4, supporting effective monitoring and evaluation of swarm behavior.

The module offers data APIs to facilitate the development of intelligent algorithms and incorporates collision detection and radiation mechanisms to support functions such as collision analysis, flight constraints, and spacing evaluation. An event-driven architecture integrates the behavior, physics, and rendering modules, enabling responsive system-level interactions—for example, upon collision, the physical module is deactivated and relevant data (e.g., object, location, timestamp) is relayed to the behavior module.

The rendering module simulates sensor data—such as vision, radar, infrared, and depth—using visual inverse projection and material post-processing techniques [26], as illustrated in Fig.3. To enhance visual fidelity, the engine employs digital twin modeling [27], fusing elevation data with satellite imagery to reproduce environments under varying weather, lighting, and temporal conditions. These data are essential for vision-based intelligent algorithms.

4) Behavior module: As an ABMS simulator, SIGMA's Behavior module is crucial for the emergence of collective behavior from individual UAV actions. The components of the behavior module are shown in the gray block of Fig.1. This module provides a standardized behavior library template, enabling the construction of an extensible decision-making framework based on real-time environmental perception and task state information.

The operational logic of the behavior module is illustrated in Fig.4. To support autonomy, cooperation, and adaptive evolution in ABMS, each node maintains an independent behavior library $B = \{b_k\}_{k=1}^{K}$, where behavior $b_k \triangleq \{\xi_k(S), \tau_k(S), L_k(S), W_k\}$ comprises an information extraction function, trigger condition, behavior logic, and priority weight. Here, S denotes the node's perceptual input, including internal states (e.g., position, velocity) and external data (e.g., inter-node distances, obstacle maps). During the simulation phase, each node sequentially processes all behaviors. For each behavior, the current state variables are transformed into behavior decisions based on its trigger condition and logic. The resulting decisions are weighted according to the behavior priority weights, generating the node's final flight control command.

During the decision-making process, each node first processes its state information to determine whether a specific behavior should be activated as (2):

$$T_k = \begin{cases} 1, & \text{if } \tau_k(\xi_k(S)) > t_M \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Where $\xi(*)$ denotes the information extraction function, $\tau(*)$ denotesthe behavior mapping function, and t_M indicates the behavior response threshold, T_k denotes the behavior activation value. When the trigger condition is met, the behavior library generates an action primitive $a_k = L_k(S)$ based on the execution logic function. Considering that some algorithms require consistency with the behaviors of other agents in the swarm, the decision process can optionally incorporate other agents' actions at the previous step as a weighting factor. The behavior decision can be represented as (3):

$$a^t = \arg\max_{b_k \in B} \left[T_k \cdot ((R(a_k^t, S) \cdot W_k + \eta \cdot sync(a_k^t, A^t)) \right] \tag{3}$$

Where $R(*)$ denotes the reward or evaluation function that generates behavior based on the current state, as defined within the behavior logic. η denotes the cooperation gain coefficient, A denotes the swarm behavior, and $sync(*)$ represents the behavior consistency function. The behavior module ensures Pareto optimality between collision avoidance and task completion through multi-objective optimization, ultimately driving the swarm to exhibit intelligent emergent behavior with both autonomy and cooperation.

Inspired by AnyLogic's design [28], SIGMA offers standardized behavior library templates. After generating the behavior library, users can directly select each node's current behavior library via the GUI of the Interface module. This behavior module allows SIGMA to handle both individual behavior logic and swarm evolution simultaneously. Once defined, the same behavior module can be ported to various environments without modification. Additionally, the behavior templates are fully developed based on ROS/MAVlink protocols, enabling easy deployment to edge devices during real-world testing by simply modifying external sensor interfaces like cameras.

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

B. General Aircraft Model

As shown in TABLE I, current UAV simulators demonstrate model-specific advantages. X-Plane excels in high-fidelity commercial aircraft simulation, JSBSim is oriented toward fixed-wing dynamics, AirSim emphasizes vision-based control for multirotors, While Gazebo enables high-fidelity rotorcraft simulation via RotorS, its simplified fixed-wing models limit applicability to real-world platforms. These simulators often lack flexibility across diverse UAV types. Therefore, SIGMA implements a general aircraft model based on a 6-DoF framework, with enhancements in modeling versatility, environmental fidelity, and sensor noise representation.

1) Kinematic block: Based on the dynamics and kinematics, the physics engine uses the control value $\boldsymbol{x}_{ctl}$, state variable $\boldsymbol{x}_{state}$, environment forces $\boldsymbol{x}_{env}$ and the aircrafts parameters ϕ_{veh} as inputs to solve for the state quantities and their derivatives $\boldsymbol{y}_{\text{kine}} \triangleq \left[\boldsymbol{p}^e, \boldsymbol{a}^e, \boldsymbol{v}^e, \boldsymbol{\omega}^e, \dot{\boldsymbol{v}}^e, \dot{\boldsymbol{w}}^e\right]^T$, which represent the position, attitude, velocity, angular velocity and their derivatives in the world coordinate system respectively. Noteworthy, in this paper, the right-superscript symbols "e" and "b" denote the North-East-Down (NED) earth frame [29] and the Head-Right-Down body frame, respectively. In sensor simulation mode, the state quantities are converted into sensor parameters through the sensor module and input into the self-driving module. Thus, the mathematical description of the relationships between the system modules is as (4)

$$\begin{cases} \boldsymbol{y}_{\text{fm}} = \boldsymbol{f}_{\text{fm}}(\boldsymbol{x}_{\text{ctl}}, \boldsymbol{x}_{\text{state}}, \boldsymbol{x}_{env}, \phi_{veh}), \\ \boldsymbol{y}_{\text{kine}} = \boldsymbol{f}_{\text{dy}}(\boldsymbol{y}_{\text{fm}}, \boldsymbol{x}_{\text{state}}) + \boldsymbol{f}_{\text{kine}}(\boldsymbol{x}_{state}), \end{cases} \tag{4}$$

Where $\boldsymbol{y}_{\text{fm}} \triangleq [D, Y, L, l, m, n]^T$ represents the output of the aircraft's forces and moments, and $\boldsymbol{x}_{\text{state}} \triangleq [u, v, w, \phi, \theta, \psi, p, q, r, x_g, y_g, h_g]^T$ is the aircraft's current velocity, position, and the attitude states. $\boldsymbol{x}_{\text{ctl}} \triangleq [\delta_a, \delta_e, \delta_r, \sum n]^T$ includes the deflections of the control surfaces and throttle setting. Where $\boldsymbol{f}_{\text{fm}}$, $\boldsymbol{f}_{\text{dy}}$ and $\boldsymbol{f}_{\text{kine}}$ represent the functions of forces and moments equations, dynamics equations and kinematics equations. In the field of aircraft model identification, a general model has been established for rigid body aircraft. This model can figure out the state of the aircraft at any moment based on flight dynamics and Euler's equations as (5)

$$\begin{cases} \sum \boldsymbol{F}_e = m \cdot \mathbf{1}_{\boldsymbol{V}} \dfrac{\delta \boldsymbol{V}}{\delta t} + \boldsymbol{\Omega} \times m\boldsymbol{V}, \\ \sum \boldsymbol{M}_e = m \cdot \mathbf{1}_{\boldsymbol{T}} \dfrac{\delta \boldsymbol{T}}{\delta t} + \boldsymbol{\Omega} \times \boldsymbol{T}, \end{cases} \tag{5}$$

Where $\boldsymbol{F}_e$ and $\boldsymbol{M}_e$ is the forces and moments in the world frame, $\boldsymbol{V} = [u, v, w]^T$ is the flight velocity, $\boldsymbol{\Omega} = [p, q, r]^T$ is the angular velocity. The momentum $\boldsymbol{T}$ is given by $\boldsymbol{T} = \int d\boldsymbol{T} = \int (\boldsymbol{r} \times \boldsymbol{V})\, \delta_m$ Where $\mathbf{1}_{\boldsymbol{V}} \frac{\delta \boldsymbol{V}}{\delta t}$ and $\mathbf{1}_{\boldsymbol{T}} \frac{\delta \boldsymbol{T}}{\delta t}$ donate the unit vectors along $\boldsymbol{V}$ and $\boldsymbol{L}$ directions, respectively. Therefore, based on the aircraft's aerodynamic model and propulsion system, the state quantities $\boldsymbol{x}_{\text{state}}$ and control quantities $\boldsymbol{x}_{\text{ctl}}$ are input into the physical engine. The engine then solves the aircraft's motion state at any given moment according to the corresponding nonlinear relationships. Environmental parameters, aerodynamic coefficients, and structural parameters are used as inputs to compute the forces and moments $\boldsymbol{y}_{\text{fm}}$ of the aircraft [30]. The formulas for calculating forces and moments are as follows (6)

$$\begin{cases} F_b = F_R + F_F + G, \\ M_b = M_R + M_F, \end{cases} \tag{6}$$

Where F_b and M_b represents the forces and moments in the body frame. The subscripts R and F denote the rotor and fixed-wing components, G represents the current gravity. The specific definitions of rotors' forces and moments can be expressed as (7).

$$\begin{cases} F_R = \sum_{i=0}^{n} \left(C_{T0} + k_\theta \theta_i + k_\lambda \lambda_i\right) \rho A R^2 \omega_i^2, \\ M_R = \sum_{i=0}^{n} \left(C_{Q0} + \dfrac{C_T^{1.5}}{\sqrt{2}} + \mu \cdot \mathbf{f}_{adv}\right) \rho A R^3 \omega_i^2 d_i, \end{cases} \tag{7}$$

Where $n, C_T, C_Q, \rho, A, R, \omega, d$ represent the motors' number, thrust coefficient, torque coefficient, air density, blade area, blade radius, angular velocity, and position distance. To address the simplifications in the JSBSim motor model and the lack of propeller aerodynamic characteristics, a semi-empirical model is introduced. Where, $k_\theta, \theta, k_\lambda, \lambda, \mu, f_{adv}$ represent the pitch angle sensitivity coefficient, pitch angle, inflow ratio correction coefficient, inflow ratio, advance ratio, and the advance ratio correction function, respectively.

In addition to the power model, SIGMA also aims to retain the excellent aerodynamic structure calculation capabilities of JSBSim. The relevant force and moment formula is (8)

$$\begin{bmatrix} F_{FL} \\ F_{FD} \\ F_{FY} \\ M_{Fl} \\ M_{Fm} \\ M_{Fn} \end{bmatrix} = \frac{1}{2} \rho V_a^2 S \begin{bmatrix} C_L(\alpha, \delta_a, \delta_e) \\ C_D(\alpha, \delta_a, \delta_e) \\ C_Y(\beta, p, r, \delta_r) \\ C_l(\beta, p, r, \delta_a, \delta_r) \\ C_m(\alpha, q, \delta_e) \\ C_n(\beta, p, r, \delta_a, \delta_r) \end{bmatrix} \tag{8}$$

Where c is the mean aerodynamic chord of the wing, b is the wing span, S is the wing area, C_* represents the aerodynamic coefficients for lift L, drag D, side force Y, roll moment l, pitch moment m, yaw moment n. α, β, p, r denote the aircraft's current angle of attack, sideslip angle, roll rate, and yaw rate. $\delta_a, \delta_e, \delta_r$ are the aileron, elevator, and rudder deflections for fixed wings. By substituting (4) and (5) into (3) and using $\boldsymbol{R}_e^b \in \mathbb{R}^3$, $\boldsymbol{F}_e$ and $\boldsymbol{M}_e$ can be obtained. Then, using $\boldsymbol{f}_{dy}$ and $\boldsymbol{f}_{kine}$, the next state of the aircraft can be solved.

To ensure aircraft compatibility, all aircraft share the same computation process and equations. The physical module's processing time (<2ms) is much shorter than the 10ms trigger interval, indicating that SIGMA's efficiency is determined solely by the number of nodes, regardless of the aircraft model.

2) Environment block: The environment block needs to obtain environmental forces in the earth coordinate based on the aircraft's position information $\boldsymbol{p}^{\mathrm{e}} \triangleq [\lambda, \psi, h]^T$ which includes information on air density, wind, magnetic fields, and other factors. The world coordinate system generally uses the WGS84 model, where λ, ψ, h represent longitude, latitude, and altitude in the world system. The WGS84 model can be used to determine gravitational information for the accelerometer at that location. Standard atmospheric models (ISA) and world magnetic models (WMM) provided by tools like JSBSim and Matlab describe air density and magnetic fields.

Meteorological satellite data from "Gaode Maps" API provides initial information like temperature, wind speed, and direction for specific coordinates. This data allows for calculating the average wind speed V_0 and direction θ_0 for local regions. Inputting this local data into a wind field model yields wind speed information for any coordinate point.

$$V(\lambda, \psi, h) = V_0 \cdot e^{-\alpha t} + d(\lambda, \psi, h, \theta_0) \cdot e^{-\beta t} \tag{9}$$

Where α and β are parameters controlling the temporal decay of the wind field, (x, y, z, θ_0) refers to the local wind field model built into the Unreal Engine, which adheres to the dry turbulence model [31]. This setup allows for the simulation of dynamic wind fields affecting the aircraft. The rendering engine can pass this information in real-time to the environmental module within the subsystem, enabling flight simulation under dynamic wind conditions.

3) Sensor block: The sensor block in the physics module supplies raw data to the autopilot's extended Kalman filter (EKF) for state estimation. The simulation system uses a nonlinear mapping method to convert state quantities into simulated sensor data, and introduces a noise model to simulate the uncertainty and interference in real measurements.

First, the next state $\boldsymbol{x}_{state}$ is inversely calculated through the observation model of the sensors to obtain the simulated input $\boldsymbol{y}_{sens}$. Then, the sensor block applies specific sensor error and noise to simulate the non-ideal characteristics of the actual sensor. General noises include measurement noise $n_e \sim (\mu_n, \sigma_n^2)$, bias error $b_e \sim (\mu_b, \sigma_b^2)$, and sensitivity error $m_e \sim (\mu_m, \sigma_m^2)$, aiming to more accurately reproduce sensor performance under actual flight conditions. (μ_*, σ_*^2) are the mean and covariance of the Gaussian noise. Finally, by combining these calculation results with the error model, the corresponding sensor data output is generated. The mathematical description of the sensor model is (10).

$$x' = x_o + n_e + b_e + m_e \cdot x_o + v_e(t) \tag{10}$$

Where x' and x_o represent the measured value and the ideal sensor value. It's usually easy to refer to the table for aircraft sensor errors. SIGMA introduces a modulated harmonic term $v_e(t)$ into the rotor model to simulate high-frequency vibrations of the airframe.

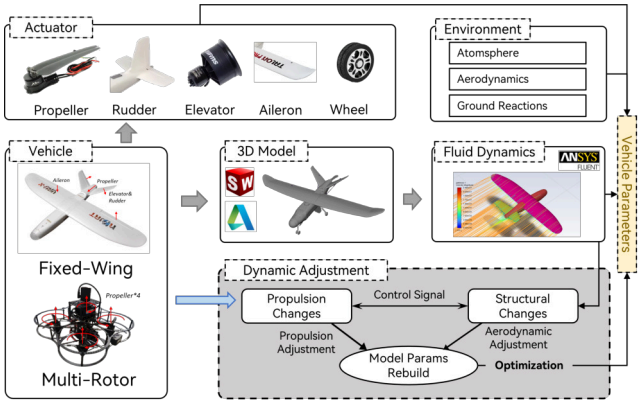


Fig. 5. Aerodynamic modeling flowchart.

$$v_e(t) = \sum_{k=1}^{K} A_k(t) \cdot \sin(2\pi f_k t + \phi_k) + \eta(t) \tag{11}$$

Where $A_k(t)$ denotes the time-varying amplitude, which is correlated with rotor speed; f_k is the harmonic frequency, $\phi_k \sim (0, 2\pi)$ represents the random initial phase angle. $\eta(t) \sim (\mu_\eta, \sigma_\eta^2)$ denotes the band-pass resonant noise.

IV. Simulator Implementation methods

A simulation platform designed for swarm intelligence algorithms must support quantitative simulation. It should also provide rich data acquisition and control interfaces to meet the growing demand for large-scale data collection in the development of intelligent algorithms. SIGMA has proposed various strategies and methods to meet the requirements of swarm intelligence algorithms in terms of high-fidelity, scalability, and compatibillity.

A. Modeling and Automatic Tuning Methods

High-fidelity methods can narrow the gap between simulation and real-world environments, allowing algorithms to perform well after migrating from simulation to reality. The modeling method based on virtual-physical integration is compatible with the latest idea of Digital Twin. Our simulator's high-fidelity modeling methods can be divided into two steps: uniform modeling process and automatic tuning based on genetic algorithm (GA) [32].

1) Uniform modeling process: The physics module of our simulator supports various aircraft types, including fixed-wing, rotor, and vertical takeoff and landing(VTOL). A uniform modeling process is proposed, as shown in Fig.5. First, the aircraft's power system (motors, servos, wheels, etc.) is modeled. Typically, actuator parameters ϕ_{act} at various power levels can be directly derived using data tables and interpolation algorithms. After that, it is necessary to build a 3D model of the aircraft based on the structural parameters ϕ_s (e.g., wingspan b and chord length c) and obtain other parameters such as the center of mass, aerodynamic center,

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

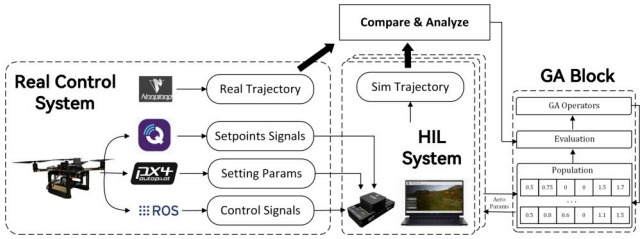


Fig. 6. Automatic tuning based on GA.

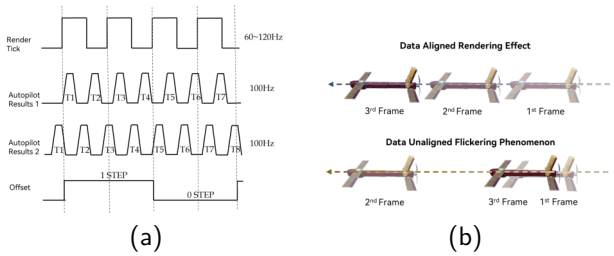


Fig. 7. Flickering phenomenon of distribute swarm simulator: (a) reason for flickering; (b) rendering effect of data aligned and unaligned.

moment of inertia, and mass. Finally, input the 3D model to Computational Fluid Dynamics (CFD) software (e.g., ANSYS®/Fluent) to calculate the aerodynamic coefficients $\phi_c = [C_L, C_D, C_Y, C_l, C_m, C_n, C_T, C_Q]^T$. The parameters above ϕ_s and ϕ_c are the basic parameters of Equations(7) and (8). Through this process, the initial parameters of the aircraft are determined.

2) GA-based automatic tuning: We propose an AutoML optimization method based on a genetic algorithm (GA) to automatically tune the aircraft parameters, as shown in Fig.6. This method involves three main steps. First, perform a cruise flight experiment in offboard mode, collecting autopilot control signals and external positioning flight data. Second, the control signals are input into the simulation system at the same timestamps, recording the system's state and trajectory. Third, compare the simulated and actual trajectories to evaluate performance using a trajectory fitting function. The GA Block then optimizes the parameters based on the results.

In the GA Block, the actuator system ϕ_{act} and aerodynamic parameters ϕ_c are used as chromosomes of the population, and the parameters are fine-tuned through the arithmetic crossover strategy. The optimization formula is (12).

$$\begin{cases} p_i = \delta x_i + (1-\delta)\, y_i, \\ q_i = \delta y_i + (1-\delta)\, x_i, \end{cases} \tag{12}$$

Where p,q are the generated offspring, x,y are individuals, and δ is a weight parameter randomly selected in the range of [0,1] to control the degree of mixing of gene values. The advantage of the arithmetic crossover is that it can introduce moderate variation while retaining most of the characteristics of the parent generation, thereby enhancing the diversity of the offspring. This block uses the relationship between fast Dynamic-Time-Warping (FastDTW [33]) and motion distance as the evaluation function r, as shown as (13).

$$r = \frac{D^e\left(t_{ex}, t_{sim}\right) T}{s} \tag{13}$$

Where D^e represents the DTW calculation function based on Euclidean distance, t_{ex}, t_{sim} are the experiment flight and simulation trajectories, T is the flight time, and s is the flight distance. The calculation formula of DTW is (14).

$$D^e_{(i,j)} = D_{i,j} + \min(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}) \tag{14}$$

TABLE II
DISTRIBUTED SYSTEM TIME ALIGNMENT METHODS COMPARISON.

Aspect	**HLC**	**Lock-step**	**BiDES (ours)**
Consistency	Causal Consistency	Strong Consistency	Eventual Consistency
Network Model	non-blocking	blocking	non-blocking
Advantage	High clock precision	Low computational load	Scalable and lightweight design
Dis-advantage	Poor latency tolerance	Single point of failure	Complex mechanism
Complexity	$O(N)$	$O(1)$	$O(N)$

Where i,j represent the reference points of the two trajectories, $D_{i,j}$ is the relative Euclidean distance between the reference points. DTW performs well in measuring the fit between two trajectories or time series when the time axis is offset or distorted.

B. Time Alignment Methods

The distributed architecture of SIGMA balances computational load across multiple devices. In the absence of time alignment, frequency drift among nodes may cause visual flicker, thereby undermining relative positioning in swarm flight. Fig.7 illustrates the underlying causes and impacts of this phenomenon.

To address this issue, various time alignment methods have been proposed. Among them, decentralized Logical-Clock and centralized Lock-Step methods are the most commonly used. The characteristics of these two approaches are summarized in TABLE II.

Logical-Clock methods are extensively used in distributed systems, such as cloud computing and the Internet of Things (IoT). Communication-layer alignment is typically achieved via NTP or PTP protocols. Algorithmic-layer consistency is maintained through Lamport clocks or Hybrid Logical Clocks (HLC). These methods guarantee causal consistency, but may fail under high network latency or infrequent node communication [34].

In UAV swarm simulation, platforms such as ARGoS, RFlySim, and Gazebo commonly employ centralized Lock-Step time alignment, which advances all agents in fixed steps via global blocking communication. While this approach is easy to implement and ensures strong consistency, it suffers from poor scalability. As the swarm size increases, communication bottlenecks become more pronounced, leading to a reduction in simulation frequency.

Algorithm 1 Swarm State Alignment Algorithm.

Input: $\mathbf{S} \leftarrow \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_k}\}$, latest trigger timestamp: $t_{\max}$
▷ $\mathbf{S}$ Contains k drone state sequences, each sequence contains multi $(state, trigger)$ tuples
Output: Swarm state vector $\mathbf{V}_s$
Initialize $\mathbf{V}_s \leftarrow []$
Set $t_{\text{tar}} \leftarrow t_{\max}$
for $i \leftarrow 1$ **to** k **do**
    $\mathbf{s} \leftarrow \mathbf{S}[i]$
    $s_l, t_l \leftarrow \mathbf{s}[-1]$
    Update $t_{\text{tar}} \leftarrow \min(t_{\text{tar}}, t_l)$
end for
for $i \leftarrow k$ **down to** 1 **do**
    $\mathbf{s} \leftarrow \mathbf{S}[i]$
    $n_{max} \leftarrow$ length of $\mathbf{s}$
    for $j \leftarrow n_{max}$ **down to** 1 **do**
        $s_l, t_l \leftarrow \mathbf{s}[j]$
        if $t_l = t_{tar}$ **then**
            Add state s_l to $\mathbf{V}_s$
        end if
    end for
end for
Return $\mathbf{V}_s$

This issue has not yet drawn widespread attention. However, with the growing demands of the low-altitude economy and autonomous combat applications, it is essential to develop a more scalable time alignment algorithm.

To address these limitations, SIGMA introduces a Bidirectional optimistic Discrete Event Simulation(BiDES) method, with improvements at both the communication and algorithmic layers. At the communication layer, a lightweight MQTT protocol is used, with a Mosquitto broker enabling efficient and reliable message exchange.

At the algorithmic layer, the alignment workflow (Fig. 8) proceeds as follows:

1) The Time Step Trigger periodically broadcasts timestamps over the LAN, subscribed to by all physical modules.
2) Upon receipt, each physics module executes simulation steps and broadcasts UAV states tagged with the corresponding frame.
3) The DES Alignment Server runs the swarm state alignment algorithm (Algorithm 1) to determine the aligned trigger.
4) The Rendering Module retrieves the swarm state matrix at the aligned trigger to generate visualization and interaction data.
5) The Behavior Module aligns individual and swarm states with interaction outputs, computes behavioral commands, and sends them to the autopilot.
6) If the gap between the aligned trigger and the latest time trigger exceeds the threshold, the physics module will activate the time-wrap mechanism to roll back the state of nodes.

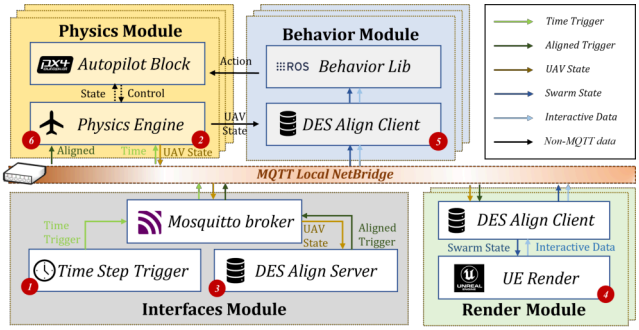


Fig. 8. BiDES time alignment method workflow.

The computational complexity of the SIGMA method is primarily determined by Algorithm 1, which has a complexity of $O(k*n)$, where n is the number of nodes and k is the length of each node's state sequence. Since k is a constant, the complexity can be simplified to $O(N)$. The method only processes simple sequences during alignment, the overall processing speed is fast. For example, when n=1000 and k=30, the computation time is less than 8ms.

Compared to the Lock-Step method, SIGMA's physics engine employs an event-triggered computation mechanism, decoupling computational load from aircraft model complexity. Its performance depends on simulation step size and node count. To avoid overload with increasing nodes, SIGMA integrates dynamic resource scheduling, allowing visualized node allocation and continuous monitoring of latency and throughput. When a node's delay exceeds a threshold, the system dynamically adjusts by reducing the interface module's trigger frequency and increasing the simulation step size, ensuring stable operation.

C. Multi-Agent Learning Toolbox

The multi-agent learning toolbox is proposed to meet the requirements of automated evolutionary learning and data collection of swarm intelligence algorithms [35]. Compared with Gazebo and AirSim, SIGMA is optimized in terms of parallel training and data collection to enhance the training and convergence speed of swarm intelligence algorithms. The workflow diagram of the Toolbox designed for the swarm intelligence algorithm in SIGMA is shown in Fig. 9.

1) Data interaction interface: SIGMA designs three API interfaces for the behavior module as shown in as shown in the yellow block of Fig.9: Order API, which interacts with the interface module and supports basic functions such as simulation reset and parameter adjustments; Physics API, deeply integrated with the autopilot, fully consistent with the real-world interaction interface; Rendering API, which interacts with the UE module, allowing access to swarm interaction data and multimodal sensor data.Before the simulation begins, users must configure environmental parameters (e.g., weather, time, terrain) and specify the behavior logic and model

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

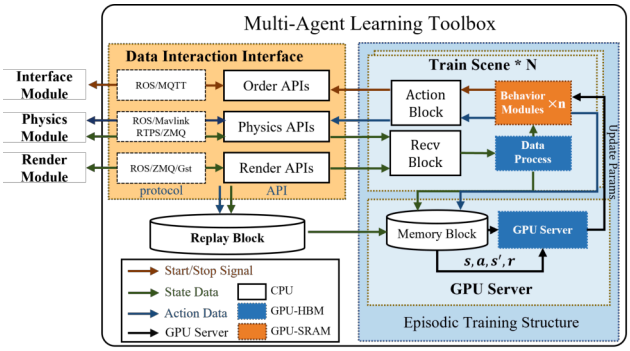


Fig. 9. Multi-agent learning toolbox workflow.

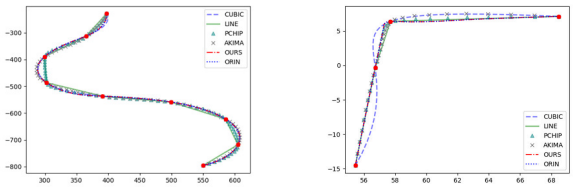


Fig. 10. Comparison of trajectory interpolation results: (a) and (b) show the fixed-wing and rotor trajectories, respectively.

TABLE III
INTERPOLATION ALGORITHMS' RESULTS (UNIT:M)

	Fixed-wing			Multi-Rotor		
	mean error	max error	RMSE	mean error	max error	RMSE
Line	3.311	11.626	4.092	0.410	1.598	0.598
Cubic	2.187	10.204	2.796	0.365	1.117	0.434
PCHIP	1.389	6.711	1.895	0.333	0.886	0.389
Akima	0.609	4.089	0.993	0.259	1.023	0.336
Ours	**0.256**	**1.798**	**0.433**	**0.102**	**0.354**	**0.121**

parameters of physical nodes. In addition to initial configuration, SIGMA allows these environmental states to be dynamically modified at runtime through exposed control interfaces.

To maintain the ROS compatibility advantages of Gazebo, SIGMA has developed a ROS2 integration plugin for UE4. This plugin integrates all the above APIs into the ROS2/FastDDS framework. At the code level, SIGMA ensures simulation-to-hardware consistency, allowing ROS2-based algorithms to be deployed directly onto physical UAV platforms and markedly reducing Sim2Real migration costs. It should be noted that, given node-count constraints and the limitations of a centralized architecture, SIGMA does not support the ROS1 framework.

2) Replay memory block: Photo-realistic simulators construct high-fidelity 3D environments in virtual domains and generate precisely annotated high-quality synthetic datasets (e.g., nuScenes, Virtual-KITTI). This method can address the issues of low-quality and high-cost image data acquisition, such as depth and semantic information. Although platforms such as AirSim and Gazebo can stream RGB, depth, and LiDAR data in real time, high-frequency transmission in multi-node setups consumes excessive bandwidth and undermines synchronization and efficiency. To alleviate these constraints, SIGMA introduces a "Replay memory block" that employs incremental storage and trajectory interpolation to collect swarm data as shown in Fig.9.

Incremental storage: In the simulation, the rendering module doesn't record the complete motion state of the swarm but captures only the changes in key information. The information includes weather, time, and the position and velocity variations of aircraft, ground vehicles, and obstacles. This incremental storage method can significantly reduce the log storage size of the simulation.

Data interpolation: Since the sampling frequency is generally lower than the rendering frequency, the simulator designs a Cubic Hermite Spline interpolation method during replay. This method incorporates a dynamic window sampling technique into the cubic spline algorithm, enabling it to effectively handle both continuous, smooth trajectories (fixed-wings) and abrupt changes in path (rotors) during motion. Denote that the default sampling window size is w, and this system adopts the window adjustment strategy to sample landmark points. The sampling rule of the window is (15).

$$W = w * (\omega \cdot \frac{\delta_{max}}{\|\delta\|_2} \cdot \frac{|\Delta\theta|_{max} + \pi}{\pi}) \tag{15}$$

Where $\delta_i = v_i - v_{i-1}$ is the difference in velocity vectors between each sampling point. $|\Delta\theta|_{max}$ is the maximum change in attitude angle within the default window. ω is the window scaling ratio constant. W is the dynamic sampling window width. Fig.10 compares this function with PCHIP, Akima, and Cubic interpolation functions. The subfigures (a) and (b)respectively show the actual flight logs and corresponding interpolation results for the TALON and P450 UAVs. Red dots indicate the recorded log coordinates. The fitting results are shown in TABLE III. This method demonstrates significant robustness, making it especially well-suited for aircraft trajectories.

In addition to data collection, the replay memory plays a key role in evaluating algorithm performance for specific tasks. After the simulation, users can access the stored logs to assess performance based on task objectives. As evaluation methods vary across tasks, this component is not built into the SIGMA system. Instead, users are encouraged to develop their evaluation modules to select the most suitable algorithm.

3) Episodic training structure: SIGMA only supports CTDE (Centralized Training with Decentralized Execution) paradigms similar to Genesis. Its architecture and implementation logic are illustrated in the blue-highlighted part of Fig.9. The CTDE training template is inspired by FasterTransformer [36]. Each parallel training scenario includes a recv block, which collects environment data and converts it into a GPU-compatible format for each agent's behavior module. After receiving deci-

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

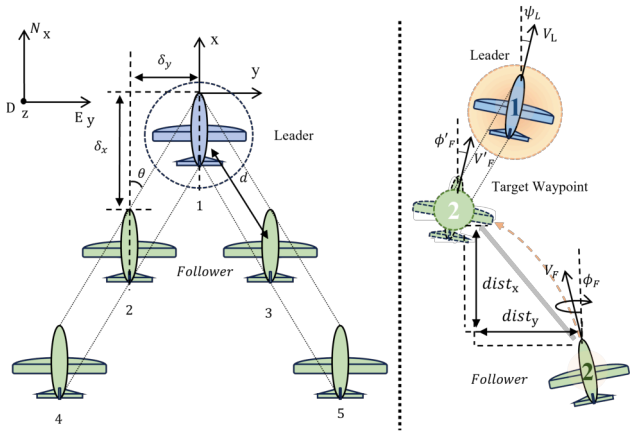


Fig. 11. Formation rules diagram.

sions from all agents, the action block interfaces with external modules to produce coordinated group behavior. All scenario data is simultaneously transmitted to the GPU server for centralized training. The behavior module leverages SRAM's high throughput to enable low-latency decision-making, while the GPU server utilizes high-bandwidth HBM to support large-scale parameter synchronization and optimization. For example, TensorRT or torch.compile is employed to accelerate model inference, while the DeepSpeed library is used on the server side to enable parallel training.

To enhance training efficiency, multiple scenarios are executed in parallel with identical initial configurations. To ensure physical-level data isolation, each scenario is equipped with an independent data synchronization server and a dedicated MQTT broker mounted on a separate port. The parallel scenario architecture enables parallel training across diverse environments, each running independently, thereby significantly enhancing scalability.

Compared to traditional platforms like Gazebo and AirSim, SIGMA achieves substantial improvements in training efficiency and scalability through a fully optimized architecture. Leveraging a heterogeneous GPU architecture, SIGMA utilizes high-throughput SRAM (1TB/s) for real-time inference and high-bandwidth HBM (3.2TB/s) for accelerated parameter updates. Training and inference threads run in parallel, accelerating the overall training and convergence speed of the algorithm.

V. Verification and Application

A. Platform Verification

1) Swarm Fidelity Verification: This paper proposes a design, implementation, and validation of rule-based formation for fixed-wing UAV System [37] to validate the swarm fidelity of SIGMA. The leader-followers rules of the five-node swarm are shown in Fig. 11.

The leader cruises along preset waypoints at a fixed speed. The followers dynamically adjust their positions based on the leader's pose. The preset node deviation of the formation (δ_x, δ_y) is (16):

$$\begin{cases} \delta_x = (-d) * \left\lfloor \dfrac{id}{2} \right\rfloor * sin\theta \\ \delta_y = (-1)^{id+1} * d * \left\lfloor \dfrac{id}{2} \right\rfloor * \cos\theta, \end{cases} \tag{16}$$

Where δ_x and δ_y represent the expected relative positions of each node in the aircraft coordinate system. d is the preset inter-aircraft distance, and θ is the desired formation angle. The expected coordinate of the world coordinate system can be obtained based on the leader's heading angle ψ_L as follows (17).

$$\begin{cases} \Delta N = x_L^w + \delta_x \cdot cos\psi_L - \delta_y \cdot sin\psi_L - x_F^w \\ \Delta E = y_L^w + \delta_x \cdot sin\psi_L + \delta_y \cdot cos\psi_L - y_F^w, \end{cases} \tag{17}$$

Where (x_L^w, y_L^w) and (x_F^w, y_F^w) represents the leader and follower's position in the world coordinate system, respectively. $(\Delta N, \Delta E)$ represent the target movement distances of the followers in the north and east directions, respectively. Based on the follower's heading angle ψ_F, the target position $\left(x_F^b, y_F^b\right)$ in the body coordinate can be calculated as (18).

$$\begin{cases} x_F^b = \Delta N \cdot cos\psi_F + \Delta E \cdot sin\psi_F \\ y_F^b = -\Delta N \cdot sin\psi_F + \Delta E \cdot cos\psi_F, \end{cases} \tag{18}$$

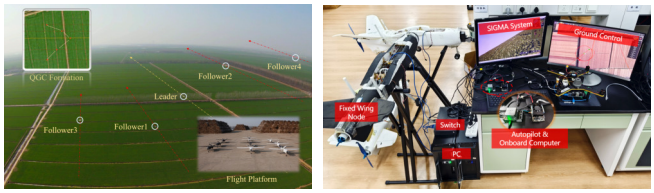
To enhance the impact of swarm fidelity, the desired waypoints are converted into the roll and speed as (19).

$$\begin{cases} \phi_F = \phi_L + P_\phi * y_F^b + P_\psi * (\psi_L - \psi_F) \\ v_F = v_{cru} + P_v * x_F^b \end{cases} \tag{19}$$

Where ϕ_* denotes the roll angle, v_* represents the flight speed, v_{cur} is the cruise speed, and P_* indicates the response parameters for each variable. The roll angle is used to control lateral spacing. The speed is adjusted using pitch and throttle to control longitudinal spacing. The leader-follower rule is widely used in UAV swarm formation. Due to control lag and aerodynamic fidelity, fixed-wing swarms may show differences in convergence time and flight trajectory. This characteristic makes the rule well-suited for evaluating the swarm fidelity of the SIGMA simulator.

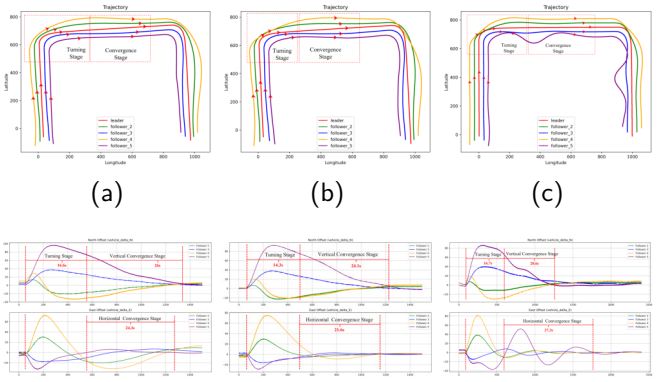
The formation control rules were validated across SIGMA, Gazebo, and real-world platforms, with identical decision-making code, communication hardware, and data interfaces to ensure consistency. The real-world and HIL simulation platforms are shown in Fig.12. The flight experiment is conducted using a TALON commercial UAV. The parameters of the formation algorithm are: P_ϕ=0.015,P_ψ=0.4,P_v=0.3,θ=45°,d=30m,v_{cur}=20m/s. The formation flight follows a rectangular trajectory of (1 km, 0.75km).

To align flight data across different platforms, swarm data is collected by the ground station at 30Hz. Fig.13 shows the swarm turning trajectory and convergence curves. When compared to the real-world experimental trajectory at 20km flight distance, the DTW values for



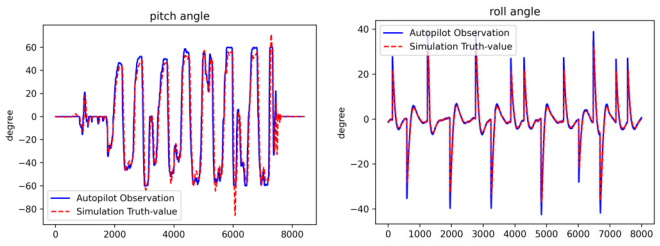
(a) (b)

Fig. 12. Real-world and simulation platform. (a) Real-world platform and experiment effect. (b) HIL simulation platform.



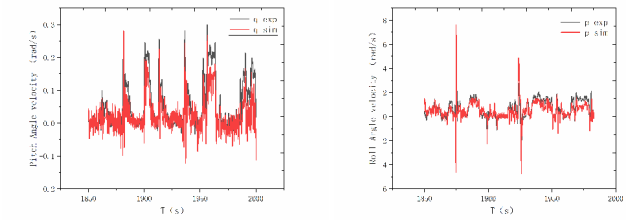
(a) (b) (c) (d) (e) (f)

Fig. 13. Trajectory and convergence curve of experiment, SIGMA and Gazebo. (a) (d) Experiment; (b) (e) SIGMA; (c) (f) Gazebo.



(a) (b)

Fig. 14. P450 Rotors response function fitting results. (a) Pitch result. (b) Roll result.



(a) (b)

Fig. 15. TALON Fixed-wing response function fitting results. (a) Pitch result. (b) Roll result.

SIGMA and Gazebo simulations are 12.7m and 35.3m, respectively. SIGMA shows convergence characteristics close to the real-world platform, with horizontal and vertical convergence time differences of -1.9s and -2.7s. In contrast, Gazebo exhibits larger deviations of -5.4s and +13.0s. Experimental results demonstrate the discrepancies between Gazebo simulations and real-world platform, which can be attributed to its simplified aerodynamic model and aggressive control response. In contrast, SIGMA's formation trajectory and convergence trend align more closely with the behavior of real aircraft, demonstrating higher swarm fidelity than Gazebo.

2) Model Fidelity Verification: SIGMA introduces a uniform modeling process and automatic tuning method for identification and optimization to ensure consistency between virtual and real models. Due to the high cost of wind tunnel experiments for evaluating the aerodynamic model, this paper uses a response function fitting method [30] for testing the model's fidelity.

The experiment tests the fit respose of attitude loop of the high-maneuverability control input, which is divided into three steps: First, the initial parameters of the aircraft actuator and aerodynamic model are obtained following the unified modeling process(Fig.5). Afterwards, multiple sets of experimental flight log files should be input into the automatic tuning method (Fig.6) to optimize the model parameters. Finally, a set of high-maneuverability flight logs in a windless or wind tunnel environment is used as verification data and input into the simulation system to test the fitting response of the optimized aerodynamic model. Due to the differences in modeling approaches and parameters, it's difficult to conduct comparative experiments with Gazebo or AirSim.

In this study, the commercial aircraft P450 and TALON were used as examples to validate the rotor and fixed-wing model fidelity of the SIGMA system. Fig.14 shows the pitch and roll channel results for the P450 rotor aircraft. Under the same control signal, the Root Mean Square(RMS) errors between the simulation and experimental outputs are 0.1042 rad for pitch angle and 0.036 rad for roll angle. The degree of fitting R^2 is 0.93. Fig.15 shows the pitch and roll channel results for the TALON fixed-wing aircraft. The RMS errors are 0.062 rad/s for pitch angle velocity, 0.073 rad/s for roll angle velocity. The degree of fitting R^2 is 0.91. These experiment results demonstrate that the simulator's modeling method can reach a high match degree, with an average degree of fit exceeding 90% (where 80% is the minimum accuracy requirement [38]).

3) Scalability Verification: A distributed simulation platform is constructed using 11 computers, which are equipped with an i7-12700F CPU and a 4070Ti GPU. Ten of them host 20 physics modules and behavior modules(totaling 200), while one node handles the rendering module and interface module. Using this setup, 200-node versions of SIGMA, Gazebo, and AirSim were deployed for performance evaluation, with comparisons conducted across three key metrics: communication latency, Age of Information (AoI), and memory usage, to assess their

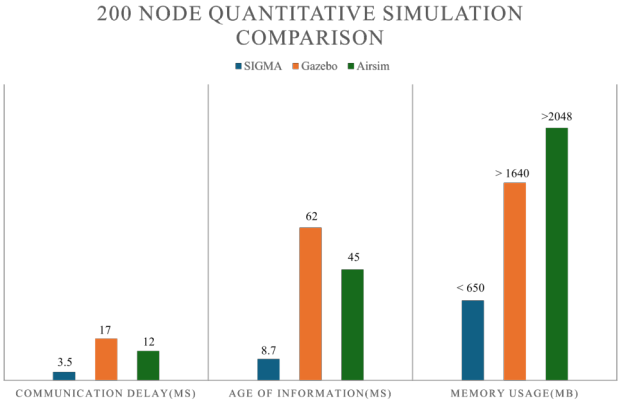


Fig. 16. Simulation comparison results for 200 nodes.

efficiency and scalability under large-scale distributed simulation conditions.

AoI is defined as the time elapsed since the generation of the latest received update, which is a key metric for assessing synchronization timeliness and communication efficiency across nodes. It is widely adopted in communication, sensing, and distributed systems [39]. The comparison results of the three simulators are shown in Fig.16.

SIGMA adopts MQTT with QoS 1 for time alignment, where each node publishes 150-bit state data at 100 Hz. In a 2-hour test, SIGMA maintains an average communication latency of 3.5ms, with an AoI of 8.7ms (±1.7ms), memory usage below 650 MB per device, and stable rendering above 60Hz. In contrast, Gazebo utilizes ROS2/FastDDS combined with Lock-Step methods. While its memory usage remains moderate(∼1.6GB), its communication latency exhibits significant fluctuations and frequent blocking at 200 nodes, with an average latency of 17∼45ms. The AoI increases sharply with node count, from 12ms at 60 nodes to 62ms(±15ms) at 200 nodes, and becomes unstable beyond 100 nodes, leading to performance degradation. AirSim employs UE4's RPC mechanism for scene synchronization. While its communication is relatively stable, this approach incurs high memory overhead (far beyond 2GB), particularly on the rendering side, where scene complexity further amplifies memory pressure. The AoI ranges from 23ms to 45ms, with observed fluctuations mainly attributed to the strong coupling between computing and rendering frequencies [40].

To further evaluate the scalability of SIGMA, this paper expanded the node count to 1000. At this scale, the system generates a network throughput of 208.4 Mbps, with the average AoI over two hours for all nodes maintained at 19ms±3.7ms. Dynamic adaptive mechanism enables stable operation at 50Hz. Furthermore, when testing with different aerodynamic configurations (rotor/fixed-wing). The variation in node computation time and memory usage is less than 2%. The impact of the aircraft type can be considered negligible.

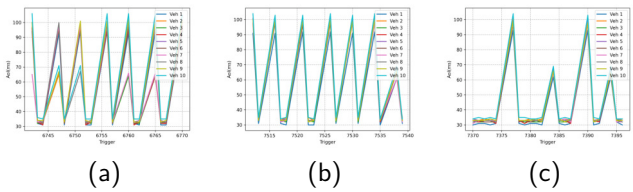


Fig. 17. Time alignment results: (a) No alignment (b) HLC (c) BiDES

4) Latency Tolerance Verification: SIGMA's BiDES algorithm demonstrates adaptability in high-latency communication environments. This paper designs a 10-node flight simulation with a random packet loss rate of 30% and communication delays of 30(±10)ms. In this situation, Gazebo and AirSim exhibit severe frame rate fluctuations, rendering them non-functional. Therefore, this paper compares the BiDES algorithm with the HLC algorithm. To prevent the influence of dynamic adaptive mechanisms, the simulation locks the alignment frequency at 30Hz and the physics node update frequency at 100Hz. Under this configuration, the AOI of each node must be integer multiples of 30ms. The alignment performance is shown in Fig.17.

When the alignment algorithm is not loaded, the raw data shows in Fig.17(a), which shows a displacement difference ranging from 30ms to 90ms, occurring randomly with noticeable flicker. The HLC method performance is illustrated in Fig.17(b). The method is generally effective, though its strict adherence to Lamport clock causality prevents dynamic adaptation to network conditions. The mean AoI of HLC reaches approximately 70ms, resulting in severe frame rate degradation. The BiDES strategy performance is shown in Fig.17(c). Leveraging optimistic alignment architecture and dynamic adaptation mechanisms, it achieves an average AoI of 38ms. While exhibiting a reduced frame rate, the system maintains stable 20Hz operation.

It should be noted that, although SIGMA exhibited a certain level of latency tolerance, its frame rate degraded significantly even when operating under the aforementioned network conditions. Since simulation systems are typically deployed in high-bandwidth local area networks, more extreme cases were not tested.

Based on the quantitative simulations, SIGMA shows clear advantages in fidelity and scalability for UAV swarm scenarios. The comparison with Gazebo and AirSim is summarized in TABLE IV.

B. Application

In the validation section, multiple quantitative simulations were conducted to verify SIGMA's fidelity and scalability. Thanks to these capabilities, the SIGMA system provides a robust parallel environment, which serves as an excellent platform for swarm intelligence algorithm training and practical application simulation.

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

TABLE IV
COMPARISON RESULTS OF SIGMA, GAZEBO, AND AIRSIM

Features	SIGMA (Ours)	Gazebo	AirSim
Model Fidelity	6-DOF model(High)	Simplified aerodynamic model(Middle)	Simplified rotor model(Low)
degree of fit	90%	–	–
Node Upper Limit	>500	60∼100	20∼40
Memory Usage	Low	Middle	High
Latency Tolerance	Operates reliably under high latency	Flickering issues under high latency	Blocking issues under high latency
Overall Strengths	High scalablity, high fidelity	user-friendly interface, rich functionality	Focus on rendering, prone to blocking

1) Multi-Agent Algorithms Training and Evaluation: In addition to the rule-based method used in the swarm flight experiment, SIGMA also supports parallel training of algorithms like MARL and NEAT [35]. Using SIGMA's episodic training structure, the study conducted parallel training across 500 synchronized scenes with MATD3 [7] and MAPPO [8] to achieve a 10-node tight formation flight. Compared to single-scene training in platforms like Gazebo and AirSim, SIGMA reduced the training time from 30 hours to just 4 hours.

In addition to accelerating training, SIGMA's Replay Memory feature enables users to uniformly evaluate and analyze the effectiveness of different algorithms for the same task. The initial position and velocity of each node are consistent across all episodes. The host's flight trajectory is random, and the reinforcement learning algorithm generates the desired attitude and throttle for the aircraft based on the host's position and the expected distance between nodes. The trajectory consistency results and convergence curves for MATD3 and MAPPO algorithms are shown in Fig. 18.

The results show that both MATD3 and MAPPO algorithms can converge to accomplish the dense swarm formation task. However, MAPPO suffers from overshooting in the x-direction and significant oscillations in the z-direction, while MATD3 maintains better stability and control precision. Additionally, MAPPO requires more training time to converge. Overall, MATD3 demonstrates stronger performance and proves more suitable for this formation task.

2) Practical Application: Beyond training, SIGMA's full-lifecycle simulation supports emerging fields like low-altitude economy and unmanned battlefield. It accurately models complete flight processes while handling core tasks like traffic control, collision avoidance, and path planning. In the unmanned battlefield, SIGMA shows strong potential by supporting the shift from single-platform to multi-agent operations. To illustrate its practical capability, this paper presents a full-process simulation of a swarm mission

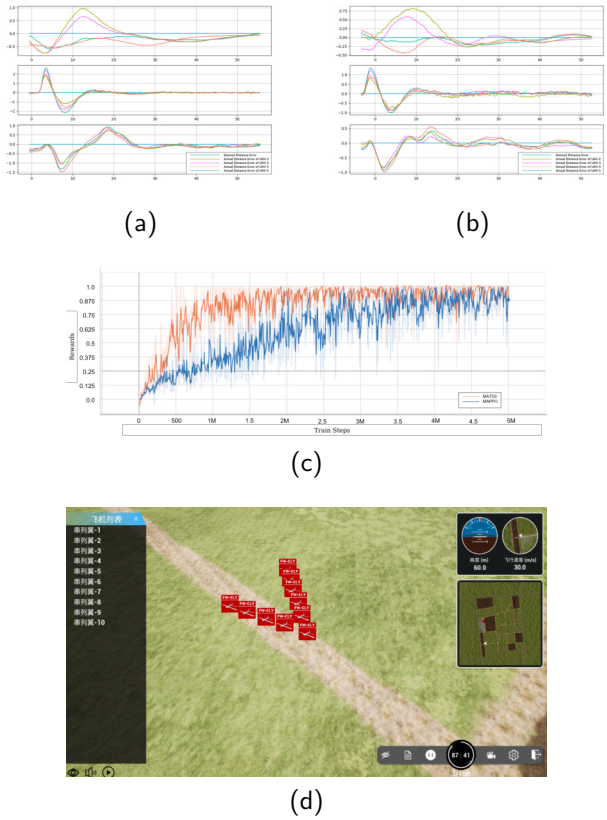


Fig. 18. MARL formation results: (a) (b) illustrate the trajectory consistency results in x,y,z directions of MAPPO and MATD3 algorithms, respectively. (c) shows the convergence of the reward function of the above algorithm.(d) shows the formation effects.

OODA loop in an unmanned battlefield scenario (Fig.19). During the configuration phase, the mission regions and swarm size are defined, with node configurations and behaviors visually specified and differentiated through icons and colors. Once the simulation begins, swarm assembly, formation, and regional search tasks are carried out in sequence. Upon detecting a ground target, each node conducts visual guidance enabled by SIGMA's real-time image transmission. SIGMA seamless integration with external command systems allows both tactical drills and large-scale combat simulations, making it a powerful tool for advancing unmanned warfare strategies.

In the field of UAV algorithm development and education, SIGMA offers a lightweight version—SIGMA(Free)—based on WSL, designed to run smoothly on laptops or standard desktops. It retains core simulation accuracy and distributed synchronization while significantly reducing hardware requirements. Users can quickly install and launch typical scenarios like multi-UAV formation, task planning, and obstacle avoidance with one command via WSL in Windows. This makes it ideal for universities, research labs, and startups by lowering costs and shortening development time. SIGMA(Free) shares the same modular design as the full version, allowing seamless migration to large-scale

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

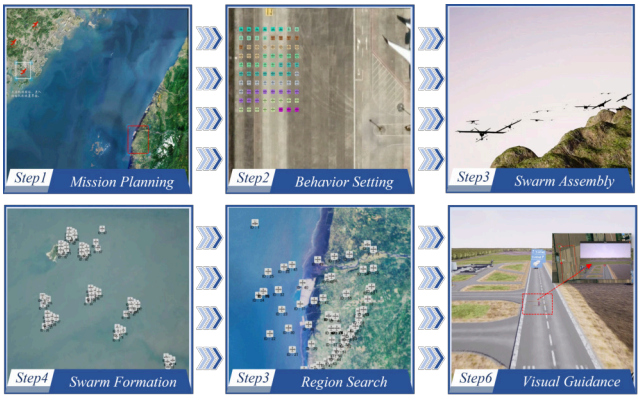


Fig. 19. Full-process simulation demonstration of Assembly-Migration-Search-Guidance.

simulations without rewriting core interfaces. Project link: https://bit-sigma.gitbook.io/tutorials-for-sigma-free

VI. Conclusion and Future Work

This paper presents SIGMA, a high-fidelity simulation platform specifically designed for UAV swarm intelligence algorithms. Built upon an ABMS distributed architecture, SIGMA is capable of supporting large-scale, high-precision swarm simulations. To address the core requirements of fidelity, scalability, and compatibility in swarm algorithm development, SIGMA introduces three key innovations: First, a modeling and automatic tuning pipeline is developed by integrating simulated and real-world data to enhance model fidelity. Second, a BiDES method is employed based on optimistic alignment architecture to achieve efficient time alignment across distributed nodes. Finally, a multi-agent learning toolbox is provided to support large-scale data acquisition and parallel training for reinforcement learning and other intelligent algorithms.

A series of quantitative experiments confirm SIGMA's strengths in simulation fidelity and scalability. Compared to Gazebo and AirSim, SIGMA enhances node scalability by the BiDES method and enables high-fidelity simulation in high-density scenarios through workload distribution based on the modular architecture of ABMS. Its parallel architecture enables more efficient analysis of the emergence of swarm behavior and faster convergence of multi-agent learning under the CTDE framework, helping reduce research costs, shorten development cycles, and improve data efficiency in UAV swarm research.

In future work, SIGMA could be enhanced by integrating with high-performance physics engines such as Taichi to further leverage GPU-based parallelization. This would enable deeper integration with learning paradigms like reinforcement learning, imitation learning, and embodied intelligence. Additionally, incorporating 3D Gaussian Splatting (3DGS) for improved rendering realism and integrating multi-sensor fusion modules would help bridge the gap between simulation and real-world deployment, positioning SIGMA as a comprehensive platform for both algorithm development and operational scenario rehearsal in UAV swarm systems.

References

[1] Y. Zhou, B. Rao, and W. Wang, "Uav swarm intelligence: Recent advances and future trends," *IEEE Access*, vol. 8, pp. 183 856–183 878, 2020.

[2] J. Kennedy, "Swarm intelligence," in *Handb. Nat. Inspired Innov. Comput.* Springer, 2006, pp. 187–219.

[3] S.-L. Jiang, Q. Liu, I. D. L. Bogle, and Z. Zheng, "A self-learning based dynamic multi-objective evolutionary algorithm for resilient scheduling problems in steelmaking plants," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 832–845, 2022.

[4] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, 2018.

[5] K. Li and Q.-S. Jia, "Multi-agent reinforcement learning with decentralized distribution correction," *IEEE Trans. Autom. Sci. Eng.*, 2024.

[6] S. Tao, Y. Xia, L. Ye, C. Yan, and R. Gao, "Db-aco: A deadline-budget constrained ant colony optimization for workflow scheduling in clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1564–1579, 2023.

[7] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," *arXiv:1910.01465*, 2019.

[8] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, and A. e. a. Bayen, "The surprising effectiveness of ppo in cooperative multi-agent games," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24 611–24 624, 2022.

[9] Z. Tu, Z. Fan, W. Zhang, and W. Liu, "Development of deep reinforcement learning co-simulation platforms for power system control," *IEEE Trans. Autom. Sci. Eng.*, 2024.

[10] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ IROS*, vol. 3. Ieee, 2004, pp. 2149–2154.

[11] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field Serv. Robot.* Springer, 2018, pp. 621–635.

[12] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 1, p. 5, 2004.

[13] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, and M. e. a. Brambilla, "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intell.*, vol. 6, pp. 271–295, 2012.

[14] J. J. Tai, J. Wong, M. Innocente, N. Horri, J. Brusey, and S. K. Phang, "Pyflyt–uav simulation environments for reinforcement learning research," *arXiv:2304.01305*, 2023.

[15] 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)*. IEEE, 2019, pp. 6941–6948.

[16] C. Cui, X. Zhou, M. Wang, F. Gao, and C. Xu, "Fastsim: A modular and plug-and-play simulator for aerial robots," *IEEE Robot. Autom. Lett.*, 2024.

[17] 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, p. 106727, 2021.

[18] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *CoRL*. PMLR, 2018, pp. 270–282.

[19] J. Huang, Y. Cui, L. Zhang, W. Tong, Y. Shi, and Z. Liu, "An overview of agent-based models for transport simulation and analysis," *J. Adv. Transp.*, vol. 2022, no. 1, p. 1252534, 2022.

[20] A. I. Ameur, O. S. Oubbati, A. Lakas, A. Rachedi, and M. B. Yagoubi, "Efficient vehicular data sharing using aerial p2p backbone," *IEEE Trans. Intell. Veh.*, 2024.

[21] C. Dissanayake and C. Algama, "A review on message complexity of the algorithms for clock synchronization in distributed systems," *arXiv:2404.15467*, 2024.

[22] K.-X. Cui, G.-R. Duan, and M.-Z. Hou, "Discrete-time model reference tracking control for a class of combined spacecraft: A high-order fully actuated system approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 4, pp. 6966–6977, 2023.

[23] A. I. Hentati, L. Krichen, M. Fourati, and L. C. Fourati, "Simulation tools, environments and frameworks for uav systems performance analysis," in *IWCMC*. IEEE, 2018, pp. 1495–1500.

[24] F. Kong, X. Liu, B. Tang, J. Lin, Y. Ren, and Y. e. a. Cai, "Marsim: A light-weight point-realistic simulator for lidar-based uavs," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2954–2961, 2023.

[25] 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, pp. 902–919, 2018.

[26] C. Floerkemeier and S. Sarma, "Rfidsim—a physical and logical layer simulation engine for passive rfid," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 33–43, 2008.

[27] V. V. Lehtola, M. Koeva, S. Oude Elberink, P. Raposo, J.-P. Virtanen, and F. e. a. Vahdatikhaki, "Digital twin of a city: Review of technology serving city needs," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 114, p. 102915, 2022.

[28] A. Borshchev, "Multi-method modelling: Anylogic," *Simul. Model. Pract. Theory.*, pp. 248–279, 2014.

[29] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *CoRL*. PMLR, 2021, pp. 1147–1157.

[30] E. Irmawan, A. Harjoko, and A. Dharmawan, "Model, control, and realistic visual 3d simulation of vtol fixed-wing transition flight considering ground effect," *Drones*, vol. 7, no. 5, p. 330, 2023.

[31] B. Zhou, Y. Li, and S. Miao, "A scale-adaptive turbulence model for the dry convective boundary layer," *J. Atmos. Sci.*, vol. 78, no. 5, pp. 1715–1733, 2021.

[32] T.-C. Chiang, A.-C. Huang, and L.-C. Fu, "Modeling, scheduling, and performance evaluation for wafer fabrication: a queueing colored petri-net and ga-based approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 3, pp. 330–338, 2006.

[33] R. Wu and E. J. Keogh, "Fastdtw is approximate and generally slower than the algorithm it approximates," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3779–3785, 2020.

[34] K. Agalianos, S. Ponis, E. Aretoulaki, G. Plakas, and O. Efthymiou, "Discrete event simulation and digital twins: review and challenges for logistics," *Procedia Manuf.*, vol. 51, pp. 1636–1641, 2020.

[35] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[36] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *arXiv:1911.02150*, 2019.

[37] E. P. de Freitas, L. A. L. da Costa, C. F. E. de Melo, M. Basso, M. R. Vizzotto, and M. S. C. e. a. Corrêa, "Design, implementation and validation of a multipurpose localization service for cooperative multi-uav systems," in *ICUAS*. IEEE, 2020, pp. 295–302.

[38] M.-d. Shen, S.-b. Chen, and X.-d. Ding, "The effectiveness of digital twins in promoting precision health across the entire population: a systematic review," *NPJ Digit. Med.*, vol. 7, no. 1, p. 145, 2024.

[39] K. Messaoudi, O. S. Oubbati, A. Rachedi, and T. Bendouma, "Uav-ugv-based system for aoi minimization in iot networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*. IEEE, 2023, pp. 4743–4748.

[40] Y. Liu, J. Fu, Z. Zhao, Y. Yang, L. Peng, and e. a. Qu, Taiguo, "Intelligent logistics service quality assurance mechanism based on federated collaborative cache in 5g+ edge computing environment," in *IWIC*. Springer, 2023, pp. 157–169.

Biography Section

Sheng Zhang received the B.E. and M.S. degrees in mechatronical engineering from the Beijing Institute of Technology, Beijing, China, in 2019 and 2022, respectively.

His current research interests include digital twin, swarm intelligence, and machine learning.

Juan Li received the B.S. degree in statistics and the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2013 and 2019, respectively.

She is currently an Associate Professor with the School of Mechatronical Engineering, Beijing Institute of Technology. Her current research interests include multiobjective evolutionary optimization and swarm intelligence.

Chang Liu received the B.E., M.S. and Ph.D. degrees in mechatronical engineering from the Beijing Institute of Technology, Beijing, China, in 2007, 2009, and 2014, respectively.

He is currently an Associate Researcher with the School of Delta Region Academy of Beijing Institute of Technology. His current research interests include micro unmanned systems and swarm intelligence.

Lei Fu received the B.E. and M.S. degrees in mechatronical engineering from the Beijing Institute of Technology, Beijing, China, in 2019 and 2022, respectively.

His current research interests include aircraft control and swarm intelligence.

Zehao Bai received the B.E. degrees in mechanical engineering from China University of Mining and Technology, Beijing, China in 2022.

His current research interests include digital twin and simulation technology.

Jie Li received the B.E. degree in electronic engineering from Harbin Engineering University, Harbin, China, in 1991, and the M.S. and Ph.D. degrees in mechatronical engineering from the Beijing Institute of Technology, Beijing, China, in 1994 and 2001, respectively.

He is currently a Professor with the School of Mechatronical Engineering from the Beijing Institute of Technology. His current research interests include the design of micro unmanned systems and swarm intelligence.