Aerospace Science and Technology 114 (2021) 106727

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

RFlySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations

Xunhua Dai [a], Chenxu Ke [b], Quan Quan [b,*], Kai-Yuan Cai [b]

[a] School of Computer Science and Engineering, Central South University, Changsha 410000, China
[b] School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, China

ARTICLE INFO

Article history:
Received 10 January 2020
Received in revised form 3 February 2021
Accepted 6 April 2021
Available online 20 April 2021
Communicated by Roberto Sabatini

Keywords:
Automatic test
Safety
Autopilot
HIL
UAV
Simulation

ABSTRACT

Autopilot systems on unmanned aerial vehicles (UAVs) are safety-critical systems whose requirements on reliability and safety are ever-increasing. However, testing a complex autopilot control system is an expensive and time-consuming task, which requires massive outdoor flight tests during the whole development stage. This paper presents an indoor automatic test platform for autopilot systems aiming to significantly improve the development efficiency and safety level of UAVs. First, a unified modeling framework is proposed for different types of aerial vehicles to make it convenient to share common modeling experience and failure modes. Then, a real-time simulation platform is developed by using automatic code generation and FPGA-based hardware-in-the-loop simulation methods to ensure simulation credibility on software and hardware levels. Finally, an automatic test framework is proposed to traverse test cases during real-time flight simulation and assess the test results. In the verification part, the accuracy and credibility of the simulation platform are verified by comparing the obtained results with experimental results, and several successful applications on multicopters demonstrate the practicability of the proposed platform.

1. Introduction

Unmanned aerial vehicles (UAVs), including fixed-wing [1], tilt-wing [2], and multicopters [3], are becoming increasingly popular in both civil and military fields [4]. For all types of UAVs, safety is always the most basic requirement, and the concern over potential safety issues remains the biggest challenge for their practical applications. For most small-scale UAVs, there is usually not enough space or payload to carry more hardware redundancy (such as backup engines, actuators, or motors) due to the limitation of cost and performance, so the autopilot control systems are becoming increasingly complex to ensure both reliable operation under normal conditions and safety decision under failure scenarios. To verify the safety of a UAV autopilot system, continuous outdoor flight tests are required during the whole development stage, but traditional test methods are usually too expensive and inefficient to cover all normal and failure cases. As a result, more efficient real-time simulation and test methods [5] for autopilot systems of UAVs are urgently needed for the ever-increasing system complexity as well as high safety requirements in complex environments.

According to [6], more than 80% of the development tasks of an autopilot system are in the decision-making layer to guarantee safety under various possible faults. However, most potential fatal faults are rare to be encountered in normal operations, so massive repeated experimental tests are essential to ensure that the autopilot system can correctly detect and handle unexpected faults. In recent years, many experimental test methods are proposed to comprehensively test and assess the safety of UAV systems. For example, in [7], the experimental test method for UAV autopilot systems is studied based on the airworthiness framework of manned aerial vehicles. However, since the amount and types of UAVs are much larger than manned aerial vehicles and the development circles are required to be shorter, the experimental test methods in the current airworthiness framework are becoming increasingly inefficient for UAVs. Besides, for small-scale commercial UAVs, the experiments (e.g., outdoor flight tests) are usually high-cost, dangerous, and regulatory restricted. Therefore, new simulation test and safety assessment methods (e.g., the real-time simulation methods [8], high-precision modeling and system identification methods [9], model-based safety assessment methods [10]) are becoming the trend for UAVs. Although experimental tests cannot be completely abandoned, simulation testing techniques are undertaking more and more safety testing and assessment tasks [11].

* Corresponding author.
E-mail address: qq_buaa@buaa.edu.cn (Q. Quan).

https://doi.org/10.1016/j.ast.2021.106727
1270-9638/

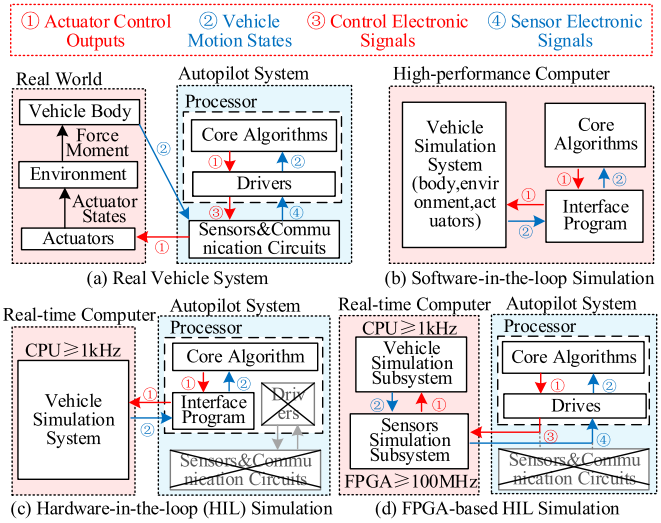


Fig. 1. Comparisons between common simulation methods.

Simulation methods for UAV autopilot systems can be divided into Software-In-the-Loop (SIL) simulation and Hardware-In-the-Loop (HIL) simulation [12]. As shown in Fig. 1(b), by running the control algorithms in the same computer with the vehicle simulation model, SIL simulation can quickly test the control algorithms with the simulation speed much faster than the real world, but it is built on the expense of losing simulation credibility (control algorithms are not running on real onboard hardware). To improve the simulation credibility, as shown in Fig. 1(c), HIL simulation is proposed by using real autopilot system hardware and real-time simulation computers to reflect the real operating environment of the control algorithms. However, traditional HIL simulation platforms usually require to modify the source code to disable drivers and add interface programs of the autopilot system to exchange sensor data and control signals to close a loop, which may affect the operating environment and performance of the original system (unstable and unreliable).

In recent years, with the utilization of Field Programmable Gate Array (FPGA) [13], real-time simulation computers (e.g., OPAL-RT®/OP series and NI®/PXI series) start to have the simulation performance of nanosecond-level real-time update frequency [8,14]. This makes it possible to simulate almost everything (including vehicle motion, sensor chips, electric circuits, and high-frequency interfaces) outside the main processor of the autopilot systems. The system structure of the FPGA-based simulation platform is presented in Fig. 1(d) which is closet to a real UAV system as shown in Fig. 1(a). Besides, the tested autopilot systems can be treated as black boxes in HIL simulation systems with no need for accessing the code or adding interface programs. Therefore, by simply replacing the sensor models, the HIL simulation system can also be applied to perform comprehensive tests for different brands of autopilot systems.

For all simulation methods, the primary challenge is to ensure simulation credibility [15], namely making people believe the simulation results are as real as experiments in the real world. The simulation credibility (compared with real systems) is mainly determined by two aspects: platform credibility and model credibility. The platform credibility can be further divided into hardware credibility and software credibility. As previously mentioned, the hardware credibility can be guaranteed by the FPGA-based HIL simulation method, while the software credibility and the model credibility are still challenges for simulation methods. The Model-Based Design (MBD) [16] method is an effective means to solve the above credibility problems by using modular visual programming technology and automatic code generation technology to standardize the modeling, developing, and testing procedures of complex autopilot systems. The MBD method can ensure the software credibility by eliminating the disturbance factors such as manual programming negligence and nonstandard development process. For example, the MathWorks®/MATLAB (the most widely used MBD software) can ensure the generated code meet the requirements of standards and guidelines such as DO-178C. In MBD methods, the whole simulation systems can be divided into many small subsystems (modules), such as kinematic modules, GPS modules, ground modules, and propeller modules. Certification authorities can verify and validate these modules to build a standard product model database for companies to develop the vehicle prototype and the corresponding vehicle simulation system. Then, the model credibility can be guaranteed by using well-validated standard component models. What is more, with MBD methods, the same component model can be applied to different types of vehicles and autopilot systems, which may significantly improve the design, verification, validation, and certification process of the unmanned vehicle filed. For ensuring model credibility, in our previous research [17], a credibility assessment method is proposed based on experiments to assess the model credibility of HIL simulation platforms from multiple aspects, such as key performance indices, time-domain characteristics, and the frequency-domain characteristics. By combining the above methods, the credibility of the simulation platform can be guaranteed from the model, development process, and platform hardware aspects.

In this paper, an automatic test platform is proposed for autopilot systems of UAVs based on FPGA-based HIL simulation, MBD methods, and high-fidelity 3D engines, aiming to significantly improve the test efficiency and safety level of autopilot software systems on unmanned vehicles. In summary, the main research contents and the corresponding contributions are listed as follows.

(i) Unified Modeling Method. There are so many in common among different types of unmanned vehicles. They should not be treated separately as more and more composite vehicles (e.g., multicopter + fixed-wing and car + fixed-wing) emerged. Therefore, a unified modeling method is proposed for different types of unmanned vehicles along with parameter measurement and identification methods to validate the obtained model and ensure simulation credibility.

(ii) Real-time HIL Test Platform with MBD. A real-time HIL test platform is built for the testing and assessment of autopilot systems. The platform is capable of simulating real-world flight situations (e.g., 3D environment, wind disturbance, air density, faults) that can be described by analytic or numerical models, and it has advantages in obtaining the true states and controlling the testing variables for quantitative assessment of test results. The utilization of MBD methods can ensure that the testing results are credible and standard-compliant.

(iii) Automatic Test Framework. An automatic test framework is proposed to traverse test cases during real-time flight simulation and assess the test results by using the proposed Platform. The proposed simulation-based safety assessment framework is a novel and original method, which is a huge improvement compared with the experiment-based assessment framework, and it is urgently needed by the UAV field as the increasing complexity of autopilot systems.

There are also many simulation test platforms released in recent years for UAV autopilot systems, such as the Airsim [18] from Microsoft® and the FlightGoggles [19] from Google®. Both of them provide SIL and HIL simulations as presented in Figs. 1(b)(c) with the latest three-dimensional (3D) engines (e.g., Unreal Engine 4 and Unity) to generate high-fidelity rendered visual data, which have been proven to be convenient and efficient in accelerating the development speed of upper-level algorithms such as Com-

puter Vision (CV) and Artificial Intelligence (AI). Similar to them, the platform proposed in this paper also uses the latest 3D engine (Unreal Engine 4) to ensure fidelity in visual scene rendering. Compared with Airsim, FlightGoggles, and other simulation platforms, the proposed automatic test platform uses: i) the FPGA-based HIL simulation as shown in Fig. 1(d) to provide higher real-time performance and hardware credibility; ii) the MBD methods to ensure the simulation credibility of the vehicle dynamics models; iii) the automatic testing framework to improve the test efficiency. With the proposed simulation-based safety assessment platform, fault and accident cases can be automatically generated for the autopilot to improve safety design and fail-safe algorithms. Then, the fault test data and safety design experience can be shared with other vehicle types and companies to improve the development of autonomous driving technology. In summary, the significant advantages of the proposed test platform are listed in four aspects:

(i) Extensibility. By changing the parameters (e.g., weight, size, and aerodynamic coefficients) of specific subsystem modules, it is easy to extend a simulation system to other vehicle systems with similar structures. Moreover, by replacing a whole subsystem module (e.g., a propeller module to a tire module), the vehicle simulation system can be extended to other types of vehicles.

(ii) Comprehensiveness. The current simulation systems mainly focus on functional testing, i.e., whether the vehicles can work properly in normal situations. However, unmanned vehicles are safety-critical systems, and most of the effects are focused on safety testing, i.e., whether the vehicles can work safely when accidents or faults happen. With the modular programming method, the fault modes, the aging process, and the probabilistic reliability property can be modeled for each subsystem module to improve the comprehensiveness of the simulation platform. Mathematically, the fault injection simulations (or other safety simulations) can be realized by online changing the module parameter or functional expressions of a subsystem module while the simulation program is running.

(iii) Verification. In practice, it is difficult to verify and validate the simulation accuracy and credibility of a complex simulation system. However, it is relatively simple to verify a small subsystem. Therefore, the modular programming method can divide a complex simulation system into many small subsystems, and verify it from lower levels to the top level. More importantly, if all subsystems used in a simulation system are well-verified modules from certification authorities, the verification efficiency can be significantly improved.

(iv) Standardization. A standard certification framework is urgently needed for unmanned vehicles to improve testing and certification efficiency. The modular programming method is a feasible way to solve this problem with the certification framework presented in Fig. 2. In this framework, the manufactures should provide the product hardware along with a simulation model that should be fully verified and certified by authority agencies based on the simulation data and experimental data. That coincides with the idea of Digital Twin [20] for the efficient design and testing of complex systems. Then, the vehicle companies can use the certified models for simulation system development and prototype design. Finally, the simulation results and experimental results can be applied for the certification of the unmanned vehicle.

2. Unified modeling method

Although different UAVs have different shapes, configurations, or flight environments, they have a similar system structure presented in Fig. 3 and share many common model features and fault modes. The common faults include actuator faults (e.g., blocked, failed, or unhealthy), sensor and communication faults (e.g., loss of signal, delays, GPS failed, and transmitting interference), environment faults (e.g., obstacles, collisions, and wind disturbances) and vehicle model faults (e.g., vibration and loss of weight). Thus, a unified framework compatible with different types of UAVs will be beneficial to share fault mode information and safety design experience to improve the safety level of the whole unmanned vehicle field. Besides, the modeling framework is also compatible with the latest idea of Digital Twin [20], which advocates each component/product/subsystem should provide a high-fidelity numerical model (digital twin) to make it possible to simulate and predict the behavior in the whole lifecycle. Then, it can also help to increase the exchange of design experience among different companies, manufacturers, and certification authorities, and decrease the repetitive work during testing and assessment processes, which is also beneficial to the rapid development requirements and better response to the rules and regulations of governments.

To make the maximum utilization of these common features, a unified modeling framework is developed with the system structure presented in Fig. 4 for all UAV systems. The core idea of the unified modeling framework is to decompose the whole vehicle system into many subsystems (e.g., motors), and each subsystem should have standard input/output/parameter interfaces to make it easier to extend to different types of UAVs. In this section, the unified modeling framework for the simulation system in Fig. 4 will be introduced in detail. The hardware structure and development process for the simulation system in Fig. 4 will be introduced in Section 3. Since the modeling methods for dynamics, flight environment and disturbances, forces and moments, and sub-system of UAVs are well-studied in many references [21–23], this paper will focus more on the modeling framework of each subsystem instead of the modeling method details.

2.1. Overall vehicle model

2.1.1. Model abstraction

In practice, a complex dynamic system usually consists of many small subsystem systems (e.g., body system and propulsion systems). Every subsystem includes input signals $\mathbf{u}_*$, output signals $\mathbf{y}_*$, parameters $\mathbf{\Phi}_*$, dynamic states $\mathbf{x}_*$, and dynamic state and output functions $\mathbf{f}_*(\cdot)$ and $\mathbf{h}_*(\cdot)$, which can be mathematically described by the following dynamic equations

$$\begin{cases} \dot{\mathbf{x}}_* = \mathbf{f}_*(\mathbf{x}_*, \mathbf{\Phi}_*, \mathbf{u}_*) \\ \mathbf{y}_* = \mathbf{h}_*(\mathbf{x}_*, \mathbf{\Phi}_*, \mathbf{u}_*) \end{cases}. \tag{1}$$

For simplicity, a dynamic subsystem in Eq. (1) is further simplified to the following input/output form as

$$\mathbf{y}_* = \mathbf{S}_*(\mathbf{u}_*). \tag{2}$$

The system description form in Eq. (2) will be applied in the following content to describe the connection relationships among different subsystems. The subscript symbol "$*$" in Eqs. (1)(2) can be replaced by different abbreviation words to represent different dynamic systems, such as the control software system $\mathbf{S}_{\text{ctrl}}$ and the sensor simulation subsystem $\mathbf{S}_{\text{sens}}$.

2.1.2. Main framework of simulation system

As presented in Fig. 4, the whole simulation system can be divided into three main subsystems: the vehicle simulation subsystem $\mathbf{S}_{\text{vehi}}$ (generating vehicle states according to the control signals), the 3D environment simulation subsystem $\mathbf{S}_{\text{3d}}$ (generating vision data according to the vehicle states), and the sensor simulation subsystem $\mathbf{S}_{\text{sens}}$ (generating sensor signals according to the vehicle states and vision data). Besides, there is an autopilot system $\mathbf{S}_{\text{ctrl}}$ (generating control signals according to the sensor data) to be tested. The above connection relationships among the above four systems are mathematically described by

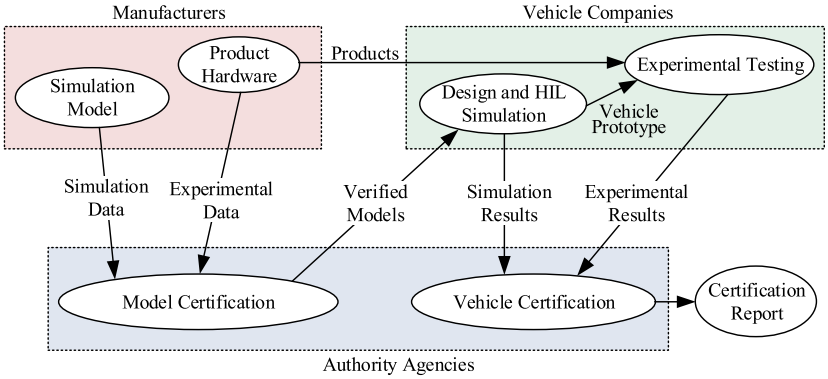


Fig. 2. Certification framework for unmanned vehicles.

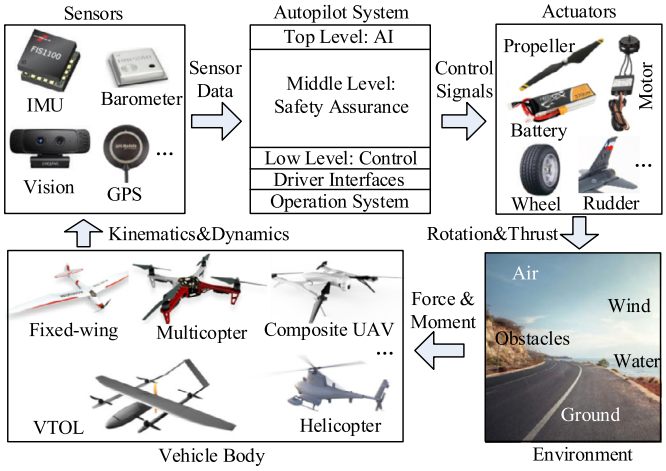


Fig. 3. System structure of unmanned aerial vehicles.

$$\begin{aligned}
&\mathbf{y}_{\text{ctrl}} = \mathbf{S}_{\text{ctrl}}\left(\mathbf{u}_{\text{ctrl}}\right),\ \mathbf{u}_{\text{ctrl}} = \mathbf{y}_{\text{sens}}\\
&\mathbf{y}_{\text{vehi}} = \mathbf{S}_{\text{vehi}}\left(\mathbf{u}_{\text{vehi}}\right),\ \mathbf{u}_{\text{vehi}} = \{\mathbf{y}_{\text{ctrl}}, \mathbf{y}_{\text{3d}}\}\\
&\mathbf{y}_{\text{3d}} = \mathbf{S}_{\text{3d}}\left(\mathbf{u}_{\text{3d}}\right),\ \mathbf{u}_{\text{3d}} = \mathbf{y}_{\text{vehi}}\\
&\mathbf{y}_{\text{sens}} = \mathbf{S}_{\text{sens}}\left(\mathbf{u}_{\text{sens}}\right),\ \mathbf{u}_{\text{sens}} = \{\mathbf{y}_{\text{vehi}}, \mathbf{y}_{\text{3d}}\}
\end{aligned} \tag{3}$$

which is consistent with the connection relationships in Fig. 4. In the following, the unified modeling methods for the above three main subsystems will be introduced sequentially.

2.2. Vehicle simulation subsystem

The vehicle simulation subsystem $\mathbf{S}_{\text{vehi}}$ in Fig. 4 can be further divided into four main subsystems: the actuator subsystem $\mathbf{S}_{\text{act}}$, the environment subsystem $\mathbf{S}_{\text{env}}$, the force&moment subsystem $\mathbf{S}_{\text{fm}}$, and the vehicle body subsystem $\mathbf{S}_{\text{body}}$. As shown in Fig. 5, the connection relationships of the four subsystems are mathematically described as

$$\begin{aligned}
&\mathbf{y}_{\text{body}} = \mathbf{S}_{\text{body}}\left(\mathbf{u}_{\text{ctrl}}\right),\ \mathbf{u}_{\text{ctrl}} = \mathbf{y}_{\text{fm}}\\
&\mathbf{y}_{\text{fm}} = \mathbf{S}_{\text{fm}}\left(\mathbf{u}_{\text{fm}}\right),\ \mathbf{u}_{\text{fm}} = \left\{\mathbf{y}_{\text{body}}, \mathbf{y}_{\text{act}}, \mathbf{y}_{\text{env}}\right\}\\
&\mathbf{y}_{\text{env}} = \mathbf{S}_{\text{env}}\left(\mathbf{u}_{\text{env}}\right),\ \mathbf{u}_{\text{env}} = \left\{\mathbf{y}_{\text{body}}, \mathbf{y}_{\text{3d}}\right\}\\
&\mathbf{y}_{\text{act}} = \mathbf{S}_{\text{act}}\left(\mathbf{u}_{\text{act}}\right),\ \mathbf{u}_{\text{act}} = \left\{\mathbf{y}_{\text{ctrl}}, \mathbf{y}_{\text{body}}, \mathbf{y}_{\text{env}}\right\}
\end{aligned} \tag{4}$$

where $\mathbf{y}_{\text{body}}$ contains vehicle motion states (e.g., position, velocity, and attitude), $\mathbf{y}_{\text{fm}}$ denotes all the forces and moments acting on the vehicle, $\mathbf{y}_{\text{env}}$ includes environment parameters (e.g., gravity, air density, terrain, and obstacle distribution), $\mathbf{y}_{\text{act}}$ denotes actuator states (e.g., rotating speed of rotors, and deflection angle of control surface). By combining the output signals in Eq. (4), the output set $\mathbf{y}_{\text{vehi}}$ for the vehicle simulation subsystem $\mathbf{S}_{\text{vehi}}$ is given by

$$\mathbf{y}_{\text{vehi}} \triangleq \left\{\mathbf{y}_{\text{body}}, \mathbf{y}_{\text{fm}}, \mathbf{y}_{\text{env}}, \mathbf{y}_{\text{act}}\right\}.$$

The key modeling methods for the four subsystems in Eq. (4) will be introduced as follows.

2.2.1. Vehicle body subsystem

As shown in Fig. 5, the vehicle body subsystem $\mathbf{S}_{\text{body}}$ computes the vehicle states $\mathbf{y}_{\text{body}}$ according to the force and moment $\mathbf{y}_{\text{fm}}$ acting on the vehicle. In practice, based on the flat-earth assumption (ignoring the curvature of the earth in a small range) and the rigid-body assumption (the body is rigid and not flexible), most vehicle body subsystem $\mathbf{S}_{\text{body}}$ can be described by nonlinear dynamic equations [24, pp. 25-54], [25, pp. 99-143] as

$$\begin{aligned}
\dot{\mathbf{x}}_{\text{body}} &= \mathbf{f}_{\text{body}}\left(\mathbf{x}_{\text{body}}, \mathbf{\Phi}_{\text{body}}, \mathbf{u}_{\text{body}}\right)\\
\mathbf{y}_{\text{body}} &= \mathbf{h}_{\text{body}}\left(\mathbf{x}_{\text{body}}, \mathbf{\Phi}_{\text{body}}, \mathbf{u}_{\text{body}}\right)
\end{aligned} \tag{5}$$

where the state set is $\mathbf{x}_{\text{body}} \triangleq \left\{{}^{\text{e}}\mathbf{p}, {}^{\text{b}}\mathbf{v}, \mathbf{R}_{\text{b}}^{\text{e}}, {}^{\text{b}}\mathbf{w}\right\}$, the input set is $\mathbf{u}_{\text{body}} = \mathbf{y}_{\text{fm}} \triangleq \left\{{}^{\text{b}}\mathbf{F}, {}^{\text{b}}\mathbf{M}\right\}$, the parameter set is $\mathbf{\Phi}_{\text{body}} \triangleq \{\mathbf{J}, m\}$, and the output set is $\mathbf{y}_{\text{body}} \triangleq \left\{\mathbf{x}_{\text{body}}, {}^{\text{e}}\mathbf{v}, {}^{\text{b}}\dot{\mathbf{v}}, {}^{\text{b}}\boldsymbol{\omega}, \mathbf{R}_{\text{b}}^{\text{e}}, \cdots\right\}$. Noteworthy, the right-superscript symbols "e" and "b" denote the North-East-Down (NED) earth frame and the Head-Right-Down body frame [24, pp. 25-54] respectively; ${}^{\text{e}}\mathbf{p} \in \mathbb{R}^3$ is the position vector defined in the earth frame; ${}^{\text{b}}\mathbf{v} \in \mathbb{R}^3$ and ${}^{\text{b}}\boldsymbol{\omega} \in \mathbb{R}^3$ are the velocity vector and the angular velocity vector defined in the body frame; ${}^{\text{b}}\mathbf{F} \in \mathbb{R}^{\Lambda}$ and ${}^{\text{b}}\mathbf{M} \in \mathbb{R}^{\Lambda}$ are the force vector and the moment vector defined in the body frame; $\mathbf{R}_{\text{b}}^{\text{e}} \in \mathbb{R}^{3\times 3}$ is the rotation matrix to transform a vector from body frame to earth frame; $\mathbf{J} \in \mathbb{R}^{3\times 3}$ and m are the moment of inertia matrix and the mass of the vehicle. Then, the dynamic equations in Eq. (5) can be applied to describe the vehicle body module $\mathbf{y}_{\text{body}} = \mathbf{S}_{\text{body}}\left(\mathbf{y}_{\text{fm}}\right)$ in Eq. (1).

2.2.2. Environment subsystem

As shown in Fig. 5, the environment subsystem $\mathbf{S}_{\text{env}}$ generates environment parameters $\mathbf{y}_{\text{env}}$ (e.g., air density, temperature, terrain, wind, and magnetic field) based on the position of the vehicle ${}^{\text{e}}\mathbf{p} \in \mathbf{y}_{\text{body}}$. In practice, the World Geodetic System (WGS84) model [26] is widely used to describe the shape of the earth, which can convert the position vector ${}^{\text{e}}\mathbf{p}$ to the earth Latitude-Longitude-Altitude (LLA) global position ${}^{\text{e}}\mathbf{p}_{\text{g}} \triangleq [\mu\ \iota\ h]^{\text{T}}$, where μ, ι (unit: degree) are the latitude and longitude, and h (unit: m) is the altitude. Then, the acceleration of gravity g can be estimated by the WGS model [26] based on the vehicle global position ${}^{\text{e}}\mathbf{p}_{\text{g}}$. Similarly, the air density and temperature are estimated by the International Standard Atmosphere (ISA) model [27], and the magnetic field vector is estimated by the World Magnetic Model (WMM) [28]. Besides, according to the Military Specification MIL-F-8785C [29], the wind velocity disturbance vector ${}^{\text{e}}\mathbf{v}_{\text{wind}} \in \mathbb{R}^3$ (defined in

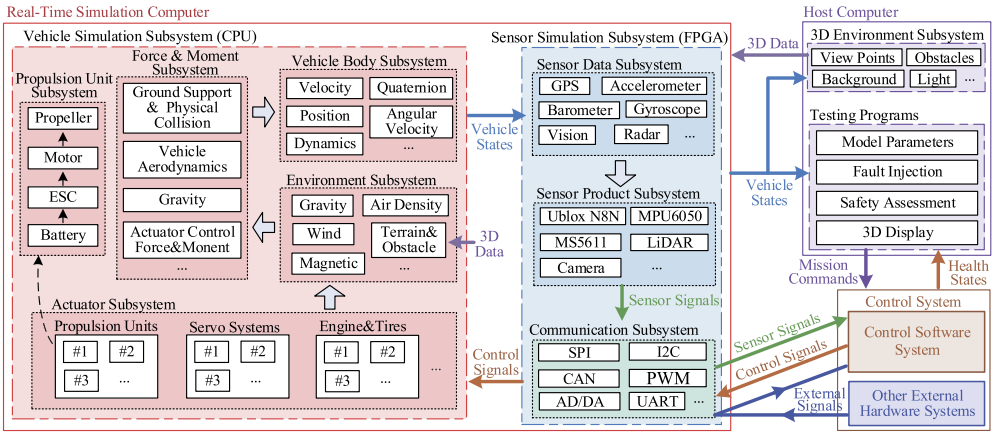


Fig. 4. System structure of the simulation test platform for UAV autopilot systems.

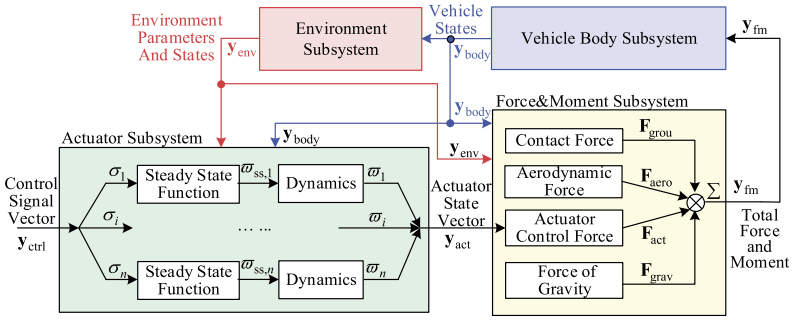


Fig. 5. The structure of the vehicle simulation subsystem.

the earth frame) can be described by the following superposition form

$$^{e}\mathbf{v}_{\text{wind}} = {}^{e}\mathbf{v}_{\text{turb}} + {}^{e}\mathbf{v}_{\text{cons}} + {}^{e}\mathbf{v}_{\text{sheer}} + {}^{e}\mathbf{v}_{\text{gust}} \tag{6}$$

where $^{e}\mathbf{v}_{\text{turb}}$ denotes the atmospheric turbulence field, $^{e}\mathbf{v}_{\text{cons}}$ denotes the prevailing wind field, $^{e}\mathbf{v}_{\text{sheer}}$ denotes the wind shear field, and $^{e}\mathbf{v}_{\text{gust}}$ denotes the wind gust field. There are many widely used mathematical models for the wind components in Eq. (6). For example, the wind turbulence $^{e}\mathbf{v}_{\text{turb}}$ can be described by the Dryden Wind Turbulence Model [29].

2.2.3. Actuator subsystem

As shown in Fig. 5, the actuator subsystem $\mathbf{S}_{\text{act}}$ outputs actuator state $\mathbf{y}_{\text{act}}$ according to the control input $\mathbf{y}_{\text{ctrl}}$ from the autopilot system. In practice, it is difficult to obtain the mathematical model of an actuator system because it is usually composed by complex mechanical components along with programmable control units, such as the Electronic Speed Controller (ESC) for UAV brushless motors, and the Electronic Control Unit (ECU) for car engines and steering systems. These control units have feedback control to ensure that the actuator steady output $\delta_{\text{ss},i}$ satisfy the preprogrammed function of the input control signal $\sigma_i \in \mathbf{y}_{\text{ctrl}}$ under different operating environments. According to [25], a complex actuator system can be linearized to a steady-state process $f_{\text{ss},i}(\cdot)$ and a dynamic response process $G_{\text{ss},i}(s)$ around the rated operation condition. For example, a motor-propeller system with an ESC can be simplified as a first-order or second-order inertial process $G_{\text{ss},i}(s)$ and a steady-state function $f_{\text{ss},i}(\sigma_i)$ as

$$\delta_i = G_{\text{ss},i}(s) \cdot f_{\text{ss},i}(\sigma_i). \tag{7}$$

Noteworthy, $f_{\text{ss},i}(\cdot)$ can be measured by static testing, and $G_{\text{ss},i}(s)$ can be measured by system identification methods through frequency-response testing [30]. By using Eq. (7), it is easy to obtain the actuator output signal $\delta_i(t)$ (propeller rotating speed) under the given control signal $\sigma_i(t)$ (throttle control signal). Then, the control force and torque generated by the state of an actuator δ_i can be obtained by the ground friction model, aerodynamic model, or other mechanical models [25,31,32].

2.2.4. Force & moment subsystem

As shown in Fig. 5, the force&moment subsystem $\mathbf{S}_{\text{fm}}$ outputs the force and moment $\mathbf{y}_{\text{fm}} \triangleq \{{}^{b}\mathbf{F}, {}^{b}\mathbf{M}\}$ to the vehicle body subsystem $\mathbf{S}_{\text{body}}$. The total force $^{b}\mathbf{F}$ and moment $^{b}\mathbf{M}$ acting on a vehicle can be divided into many components from different sources. Taking the force vector $^{b}\mathbf{F} \in \mathbf{y}_{\text{fm}}$ (defined in the body frame) as an example, it can be described by the following superposition form

$${}^{b}\mathbf{F} = {}^{b}\mathbf{F}_{\text{aero}} + {}^{b}\mathbf{F}_{\text{grav}} + {}^{b}\mathbf{F}_{\text{cont}} + \sum {}^{b}\mathbf{F}_{\text{act},i} \tag{8}$$

where $^{b}\mathbf{F}_{\text{aero}} \in \mathbb{R}^3$ denotes the aerodynamic force vector, $^{b}\mathbf{F}_{\text{grav}} \in \mathbb{R}^3$ denotes the force of gravity vector, $^{b}\mathbf{F}_{\text{cont}} \in \mathbb{R}^3$ denotes the contact force vector from ground supporting or physical collision, and $^{b}\mathbf{F}_{\text{act},i} \in \mathbb{R}^3$ denotes the control force vector generated by an actuator. Noteworthy, the above force vectors should be all transformed to the body center and projected to the body frame.

The aerodynamic force vector $^{b}\mathbf{F}_{\text{aero}}$ is a nonlinear function determined by the relative speed of the surrounding air $^{e}\mathbf{v}_{\text{rel}}$ as

$$^{e}\mathbf{v}_{\text{rel}} \triangleq {}^{e}\mathbf{v}_{\text{wind}} - {}^{e}\mathbf{v}$$

where $^{e}\mathbf{v}_{\text{wind}}$ is the wind speed from the environment subsystem in Eq. (6) and $^{e}\mathbf{v}$ is the vehicle speed from the body subsystem in Eq. (5). The high-precision aerodynamic modeling method has been well studied in [24], which is compatible with all types of vehicles such as multicopters [25], helicopters [31] and cars [32].

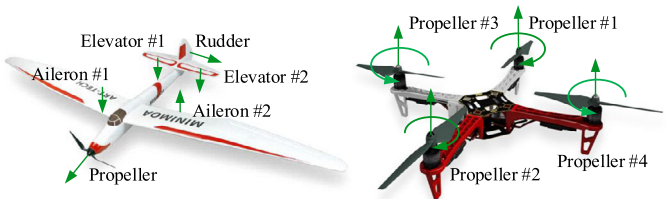


Fig. 6. Actuator force models for different types of UAVs.

The contact force ${}^{\mathrm{b}}\mathbf{F}_{\mathrm{cont}}$ caused by the ground supporting or physical collision can be modeled by simplifying the vehicle body shape to a cuboid or a cylinder and simplifying the contact surface to a spring-loaded system. By adjusting the spring stiffness, it is convenient to simulate physical contact on objects with different surface hardness. The terrain and obstacle information comes from the environment subsystem output $\mathbf{y}_{\mathrm{env}}$, which further comes from the 3D environment subsystem $\mathbf{S}_{3\mathrm{d}}$.

The actuator force vector ${}^{\mathrm{b}}\mathbf{F}_{\mathrm{act},i}$ can be unified described by the following nonlinear expression

$$ {}^{\mathrm{b}}\mathbf{F}_{\mathrm{act},i} = \mathbf{f}_{\mathrm{act},i}\left(\mathbf{\Phi}_{\mathrm{fm}}, \mathbf{y}_{\mathrm{env}}, \mathbf{y}_{\mathrm{body}}, \delta_i\right) \tag{9} $$

where $\delta_i \in \mathbf{y}_{\mathrm{act}}$ is the instantaneous state of an actuator (the rotating speed of a propeller, deflection angle of a servo system, or driving torque of a tire) from the actuator module $\mathbf{S}_{\mathrm{act}}$. Noteworthy, the expression of $\mathbf{f}_{\mathrm{act},i}(\cdot)$ is also related to the vehicle state $\mathbf{y}_{\mathrm{body}}$ and the environment state $\mathbf{y}_{\mathrm{env}}$, and the methods to obtain the force model $\mathbf{f}_{\mathrm{act},i}(\cdot)$ have been well studied in [25,31,32].

As shown in Fig. 6, the different types of vehicle simulation models are mainly distinguished by the actuator types and configurations. In a similar way, the actuator force models for different types of vehicles in Fig. 6 can be easily obtained with proper actuator system modeling methods.

2.3. 3D environment simulation subsystem

The 3D environment subsystem $\mathbf{S}_{3\mathrm{d}}$ in Eq. (3) aims to generate vision data $\mathbf{y}_{3\mathrm{d}}$ based on the vehicle states $\mathbf{y}_{\mathrm{vehi}}$ from the vehicle simulation subsystem. The vision data will be sent to the sensor subsystem to generate data for vision sensors such as cameras, radars, range finder sensors, etc.

Currently, many widely used 3D environment engines can be applied for vehicle vision modeling. For example, the Simulink 3D Animation Toolbox provides interfaces to conveniently access the video stream for image process and controller design in Simulink; the Airsim [18] is developed by Microsoft® to generate high-fidelity visual and physical simulation environment using Epic Games®/Unreal Engine 4 (UE4). Both Simulink 3D Animation Toolbox and Airsim can be applied to different types of vehicles, including aircraft and cars. There are also many 3D simulation environments exclusively developed for specific types of vehicles. For example, the Gazebo HIL simulator [33] for visual simulation of autonomous cars, and the FlightGear [34] for aircraft simulations.

These 3D simulation engines can be applied to develop the 3D environment subsystem to generate vision data (cameras), obstacle distance information (rangefinders, radars), or point cloud data (Lidar). With the development of GPU performance and 3D modeling technology, the obtained vision data $\mathbf{y}_{3\mathrm{d}}$ will be more and more high-fidelity and realistic in the future, which will significantly improve the credibility of visual simulations.

2.4. Sensor simulation subsystem

The sensor subsystem $\mathbf{S}_{\mathrm{sens}}$ in Eq. (3) describes the process to transform the vehicle state $\mathbf{y}_{\mathrm{vehi}}$ and vision data $\mathbf{y}_{3\mathrm{d}}$ to the electric signals $\mathbf{y}_{\mathrm{sens}}$ for the autopilot system. Concretely, it can also be divided into three small subsystems, including the sensor data subsystem $\mathbf{S}_{\mathrm{data}}$, the sensor product subsystem $\mathbf{S}_{\mathrm{prod}}$ and the communication subsystem $\mathbf{y}_{\mathrm{sens}}$. As shown in Fig. 4, their connection relationships are mathematically described as

$$ \begin{aligned} &\mathbf{y}_{\mathrm{data}} = \mathbf{S}_{\mathrm{data}}\left(\mathbf{u}_{\mathrm{data}}\right),\ \mathbf{u}_{\mathrm{data}} = \mathbf{u}_{\mathrm{sens}} = \{\mathbf{y}_{\mathrm{vehi}}, \mathbf{y}_{3\mathrm{d}}\} \\ &\mathbf{y}_{\mathrm{prod}} = \mathbf{S}_{\mathrm{prod}}\left(\mathbf{u}_{\mathrm{prod}}\right),\ \mathbf{u}_{\mathrm{prod}} = \mathbf{y}_{\mathrm{data}} \\ &\mathbf{y}_{\mathrm{sens}} = \mathbf{y}_{\mathrm{com}} = \mathbf{S}_{\mathrm{com}}\left(\mathbf{u}_{\mathrm{com}}\right),\ \mathbf{u}_{\mathrm{com}} = \mathbf{y}_{\mathrm{prod}} \end{aligned} \tag{10} $$

where $\mathbf{y}_{\mathrm{data}}$ contains ideal data for sensors (e.g., the acceleration of accelerometers, the magnetic field of magnetometers, and the image or point cloud data of vision sensors), and $\mathbf{y}_{\mathrm{prod}}$ is the sensor signals after adding detailed product features (e.g., noise level, temperature drift, failure mode, and camera distortion), and $\mathbf{y}_{\mathrm{sens}}$ is the binary electrical signals transmitted to the autopilot system for position and attitude estimation.

2.4.1. Sensor data subsystem

As described in Eq. (10), the sensor data subsystem $\mathbf{S}_{\mathrm{data}}$ generates the sensor data $\mathbf{y}_{\mathrm{data}}$ (applicable for a class of sensor products) based on the vehicle state $\mathbf{y}_{\mathrm{vehi}}$ and vision data $\mathbf{y}_{3\mathrm{d}}$. Transformations are usually required to obtain the sensor data from the vehicle states and vision data. For example, accelerometers measure the specific force (the difference between the acceleration of the aircraft and the gravitational acceleration) [35, p. 122] instead of the vehicle acceleration ${}^{\mathrm{b}}\dot{\mathbf{v}} \in \mathbf{y}_{\mathrm{vehi}}$. Similar computation expressions are applied to other types of sensors, such as the GPS Modules (longitude and latitude obtained from the vehicle position), electronic compasses (the magnetic field intensity obtained from the attitude and global position of the vehicle), optical flow sensors (relative velocity obtained from image stream), etc.

2.4.2. Sensor product subsystem

The sensor product subsystem $\mathbf{S}_{\mathrm{prod}}$ is developed to add product features (e.g., noise, vibration, and calibration) to the sensor data $\mathbf{y}_{\mathrm{data}}$ obtained from the above sensor data subsystem $\mathbf{S}_{\mathrm{data}}$. Given the same sensor data, different sensor products may obtain different results due to product features, so the senor product subsystem is necessary.

In most cases, given the ideal sensor data $\mathbf{x}_{\mathrm{m}} \in \mathbf{y}_{\mathrm{data}}$, the noise feature is mainly reflected in the noise $\mathbf{n}_{\mathrm{a}}$ and bias drift $\mathbf{b}_{\mathrm{a}}$ of the measured value $\mathbf{x}'_{\mathrm{m}}$, which can be described as [25, p. 151]

$$ \begin{cases} \mathbf{x}'_{\mathrm{m}} &= \mathbf{x}_{\mathrm{m}} + \mathbf{b}_{\mathrm{a}} + \mathbf{n}_{\mathrm{a}} \\ \dot{\mathbf{b}}_{\mathrm{a}} &= \mathbf{n}_{\mathrm{b}} \end{cases} \tag{11} $$

where $\mathbf{n}_{\mathrm{a}} \sim N\left(0, \sigma_{\mathrm{a}}^2\right)$ and $\mathbf{n}_{\mathrm{b}} \sim N\left(0, \sigma_{\mathrm{b}}^2\right)$ are zero-mean Gaussian noise vectors for inertial sensors. The standard deviation parameters $\sigma_{\mathrm{a}}, \sigma_{\mathrm{b}}$ can be found in the datasheet document of a sensor product or obtained by system identification with the actual sensor output signals.

If the system is not affected by vibrations, σ_{a} can be modeled as a constant value. When the vibration feature of a sensor is considered, the measuring noise $\mathbf{n}_{\mathrm{a}}$ may also be affected by the vibration from many sources (e.g., engines, motors, and fuselage). Therefore, the standard deviation σ_{a} is not always a constant value, which should be modeled based on the actual system characteristics.

The calibration feature is mainly determined by the working environment of the installation configuration a sensor, which can be described as [25, p. 149]

$$ \mathbf{x}''_{\mathrm{m}} = \mathbf{T}_{\mathrm{e}}\mathbf{K}_{\mathrm{e}}\left(\mathbf{x}'_{\mathrm{m}} + \mathbf{p}_{\mathrm{e}}\right) \tag{12} $$

where $\mathbf{p}_{\mathrm{e}}$ is a constant vector for the position installation deviation, $\mathbf{T}_{\mathrm{e}}$ is a rotation matrix for the installation deviation, $\mathbf{K}_{\mathrm{e}}$ is a diagonal matrix for the scale deviation, and $\mathbf{x}''_{\mathrm{m}} \in \mathbf{y}_{\mathrm{prod}}$ is the final

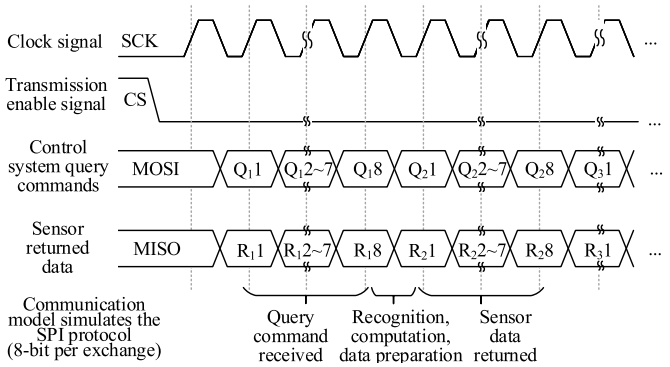


Fig. 7. Communication subsystem model for SPI buses.

output data of a sensor. Eqs. (11)(12) are applicable to most types of sensors to simulate the properties of real sensor products.

There are also many methods to add product features for vision sensors. For example, the methods to add camera features (e.g., blurs, distortions, and noises) to an ideal image are introduced in [36]. Other environmental factors (e.g., lighting, reflection, and fogging) can be simulated by the latest 3D engines, such as UE4.

2.4.3. Communication subsystem

The communication subsystem is developed to transform the sensor data with product features $\mathbf{y}_{\text{prod}}$ to binary electronic signals $\mathbf{y}_{\text{sens}}$ for the autopilot system. The outputs $\mathbf{y}_{\text{prod}}$ of the above sensor product subsystem $\mathbf{S}_{\text{prod}}$ are decimal numerical signals, but binary electronic signals are required for the communication requirements between the autopilot system and other hardware. There are many communication interfaces and protocols widely used in the autopilot systems, such as SPI, Inter-Integrated Circuit (I^2C) [37], Controller Area Network (CAN), Universal Asynchronous Receiver-Transmitter (UART), and Pulse-Width Modulation (PWM).

The mathematical model for a communication interface is usually simple, but it is hard to be simulated by a CPU-based simulation computer due to the extremely high real-time update frequency and bandwidth requirements. Taking the SPI interface as an example, the SPI interface uses four signal wires to exchange information between the master device (the main processor of the autopilot system) and the slave device (onboard sensors). As shown in Fig. 7, the sensor has to finish the command recognition, measured data computation, and output data preparation within a small interval (after the previous byte data received and before the following byte data to be sent). In a real sensor chip, the above process is instantaneously realized by analog circuits whose time consumption can be treated as infinitely small. However, for a real-time simulation system, it usually requires a real-time update frequency at the nanosecond level to ensure that the sensor signals are correctly computed, prepared, and transmitted. For the performance requirements, this paper uses the FPGA system to simulate all sensor communication features (e.g., data transmission, chip recognition, programmable setting functions), which ensures all the sensor hardware related low-level test cases can be simulated by the simulation platform.

3. *Real-time HIL test platform with MBD*

This section presents the hardware structure and development framework of the proposed HIL test platform in Fig. 4 successively.

3.1. Hardware structure of the HIL platform

The platform hardware consists of three parts.

3.1.1. Real-time simulation computer

A real-time simulation computer (also called real-time simulator) is a special type of computer, running Real-Time Operation System (RTOS) to ensure the simulation systems run at the same speed as the actual physical system. The latest COTS simulation computers usually provide a CPU-based system and an FPGA-based system for different requirements of real-time update frequency; the CPU-based system is better at running complex simulation models with moderate frequency requirements (usually smaller than 100 KHz); the FPGA-based system is better at running simple simulation models with extremely high frequency (usually larger than 100 MHz). By combining the advantages of the above two systems, the hardware structure of the proposed test platform is presented in Fig. 4, where the CPU-based system is applied to run the vehicle simulation subsystem presented in Section 2.2 and the FPGA-based system is applied to run the sensor simulation subsystem presented in 2.4.

3.1.2. Autopilot system

The autopilot system is the test object of the proposed test platform. The autopilot system computes control signals for driving the actuators according to the vehicle states measured and estimated by different sensors. To ensure the autopilot system can normally work in the proposed test platform, the original sensors should be blocked (or removed), and the sensor pins on the autopilot system should be reconnected to the FPGA-based system of the real-time simulation computer. Then, the autopilot system can receive the simulated sensor chip signals for full-function operation. The above process only needs to know the brands and models of sensors used in the autopilot system and has no requirement to access the source code or internal hardware structure. Therefore, it is practical to perform black-box testing for different autopilot system products from different UAV companies.

3.1.3. Host computer

A high-fidelity 3D simulation environment is also essential for training or testing the top-level algorithms, including computer vision, machine intelligence, and decision making systems. Therefore, a host computer with high-performance Graphics Processing Units (GPUs) and realistic 3D engines are used in this paper to generate vision signals to the real-time simulation computer. The latest high-end consumer GPUs (such as NVIDIA GTX2080) have been powerful enough to generate high-resolution (larger than 4K) video streaming with an update frequency more than 100 Hz, which is capable of simulating most vision sensors. Besides, the host computer also takes responsibility for running other auxiliary programs such as model parameter configuration program, 3D display program, ground control program, etc. Noteworthy, for different data bandwidth and real-time requirements, the connection and communication among the simulation computer and the host computer can be realized by optical fibers, network cables, serial ports, etc.

3.2. Development framework with MBD

3.2.1. Modular programming

In practice, developing a complex simulation software through hand-coding is a difficult and unreliable task due to too many mathematical operations. Any coding mistake, logical mistake, or unknown vulnerability may lead to wrong or inaccurate simulation results. The types and amounts of UAVs will be far more than manned aerial vehicles, and the development cycles are required to be much shorter, so the traditional hand-coding methods are no longer suitable for developing simulation software for UAVs. As a result, modular (also described as graphical or visual) programming methods have been widely used in many MBD tools (e.g.,

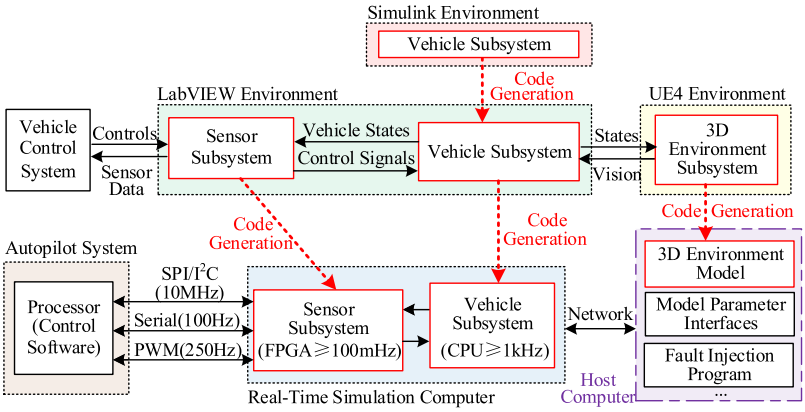


Fig. 8. Code generation and deployment framework for simulation software of HIL testing platforms.

MathWorks®/Simulink and NI®/LabVIEW). The whole simulation system presented in Section 2 has been divided into many simple and independent subsystems, and each subsystem can be easily realized by a visual module or block in the above MBD tools, which makes it easy to develop and test a complicated vehicle simulation system.

3.2.2. Automatic code generation

Modular programming and automatic code generation are two of the most significant features of MBD tools such as Simulink, LabVIEW, and UE4. Simulink is better at developing complex simulation systems (such as vehicle simulation systems), and LabVIEW is better at developing hardware-closed simulation systems (such as sensors, circuits, and communication signals) and deploying the simulation program to the real-time simulator. For example, the Aerospace Blockset in Simulink [38] provides many demos for quickly developing vehicle simulation systems, such as aircraft and multicopter. There are many 3D engines for developing high-fidelity vehicle 3D simulation environments. Among these engines, UE4 provides modular visual programming functions, so the UE4 environment is selected to develop 3D simulation scenes for the HIL platform.

To take full advantage of both MBD tools, the development process for simulation system software is shown in Fig. 8. The development process is divided into the following steps: (i) developing and verifying the vehicle simulation model in Simulink Environment; there are many powerful verifying tools in Simulink such as requirements traceability, code coverage check, document generation, etc., which guarantee the simulation software meets the standards and guidelines such as DO-178C; (ii) compiling the vehicle simulation subsystem into code, and importing it to the LabVIEW environment; (iii) building the sensor subsystem in LabVIEW and building the interfaces to communicate with other systems; (iv) generating code and executable files to deploy them to the real-time simulation computer; (v) developing the 3D simulation environment in UE4 and deploy it to the host computer. The whole development process in Fig. 8 is efficient and reliable because all coding and deploying operations are automatically finished by MBD tools without much human intervention. Therefore, it is convenient to replace some models and rebuild the simulation system for different types of UAVs or autopilot systems.

3.3. Platform implementation

3.3.1. Hardware composition

Based on the hardware structure presented in Fig. 4, a real-time HIL test platform is developed as shown in Fig. 9. The real-time simulation computer adopted in the platform is the NI®/PXI simulator with CPU board: PXIe-8133 (Intel Core I7 Processor, PharLap ETS Real-Time System) and FPGA I/O Module: PXIe-7846R. The host computer is a high-performance workstation PC with professional GPUs. The autopilot system is a Pixhawk autopilot, which is one of the most popular open-source autopilot hardware systems for small-scale UAVs. All the onboard sensors (IMU, magnetometer, barometer, etc.) and external sensors (GPS, rangefinder, camera, etc.) of the Pixhawk hardware are blocked, and the sensor pins are reconnected to the FPGA I/Os to receive the simulated sensor signals through interfaces including SPI, PWM, I^2C, UART, UBX, etc. In the real-time simulation computer, the update frequency of the vehicle simulation model is up to 5 kHz, and the update frequency of the sensor simulation model is up to 100 MHz, whose performance is fast enough for HIL simulations of most commercial autopilot systems. The communication between the host computer and the real-time simulation computer is realized by network cables with TCP and UDP protocols.

3.3.2. Software development

The simulation software of the real-time HIL test platform is developed based on the MBD method in Section 3.2. The Simulink is selected to develop the vehicle simulation model because it is the most professional and widely used software for vehicle dynamic system development; the LabVIEW is selected to develop the sensor simulation model because it is efficient and convenient in real-time simulation system development; the UE4 (version: 4.22) is selected to develop the 3D environment model because it is one of the most realistic 3D engines in the development of simulation systems, games and VR systems. The Simulink, LabVIEW, and UE4 developing environments all provide convenient modular visual programming environments (see Fig. 10) and automatic code generation technology for the development of simulation systems, which are perfect for the implementation of the model-based design method. After the simulation models are all developed, the code generation and deployment framework in Fig. 8 is applied to deploy the simulation software to the real-time HIL test platform presented in Fig. 9.

3.3.3. High-fidelity 3D simulation environment

The fidelity of the 3D simulation model is important for testing the vision-related functions of autopilot systems, such as visual data processing, obstacle avoidance, safety decision-making, etc. With UE4, it is easy to develop high-fidelity 3D scenes for different types of UAVs in different environments. For example, as shown in Fig. 11, we have developed several 3D simulation scenes in UE4 for the HIL test platform. According to the comparisons with experiments, the display effect presented in Fig. 11 has been realistic enough for most UAV systems to simulate the real indoor or outdoor scenes.

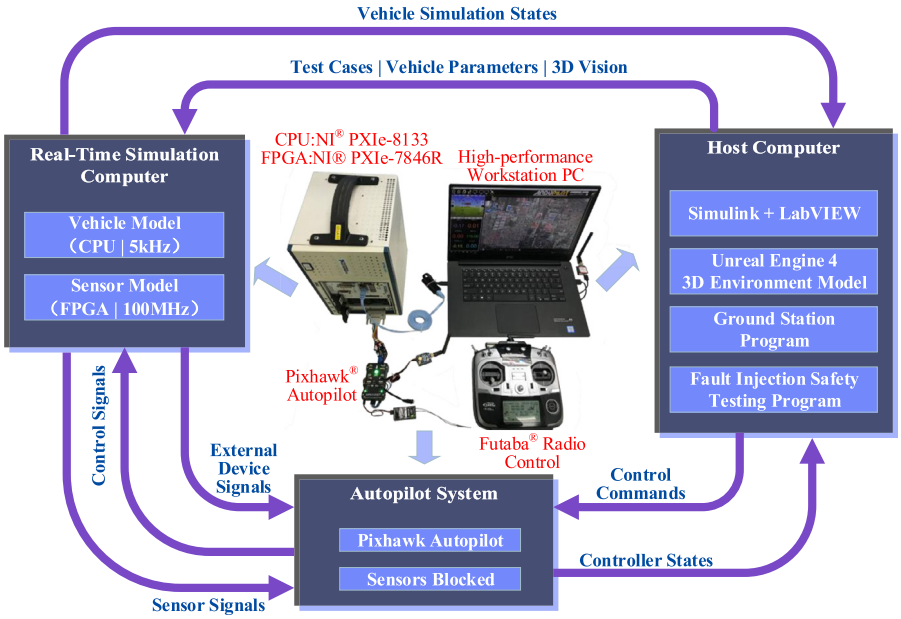


Fig. 9. Hardware composition of the real-time HIL test platform.

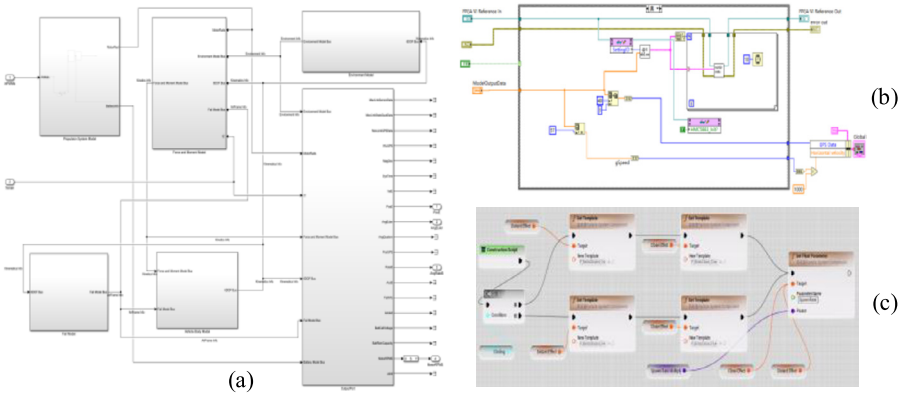


Fig. 10. Modular visual programming environments in (a) Simulink, (b) LabVIEW, and (c) UE4.

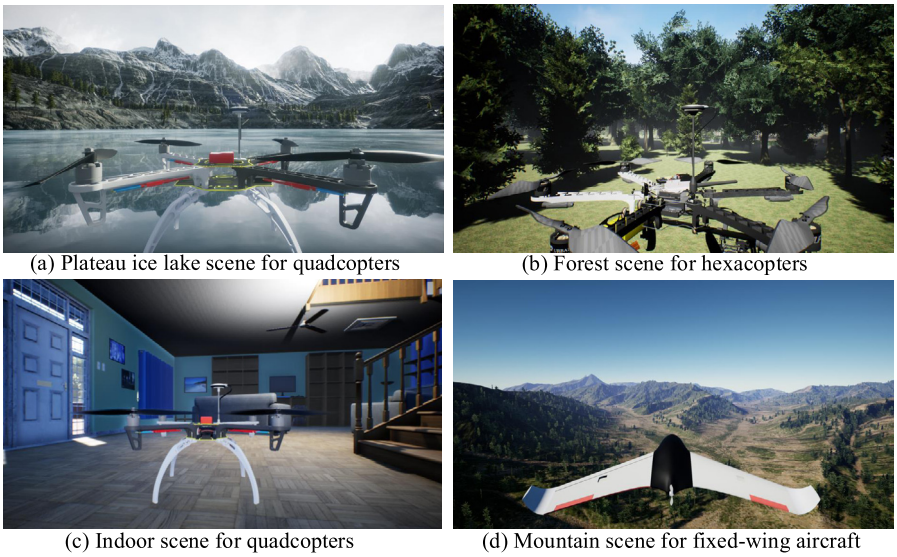


Fig. 11. 3D simulation scenes developed by UE4 for different types of UAVs.

Table 1
A test case demo for the automatic testing framework.

ID	Case description	Control command sequence	Data required	Desired performance
C1	Motor #2 failed during hovering flight	• **5s**: Arm the drone • **10s**: Take off to 10 m above ground and start hovering flight • **20s**: Inject the fault by stopping motor #2 • **25s**: Stop simulation and record flight data	Position, velocity, attitude, angular velocity curves	• Maximum attitude fluctuation $\leq 10°$ • Maximum impact speed on ground ≤ 5 m/s • $\cdots$

3.4. Automatic testing and assessment framework

3.4.1. Fault modeling and test cases

With the fault modes (e.g., sensor failure, wind disturbances, and motor fault) being well modeled, the proposed HIL test platform can also be applied to perform automatic safety testing for autopilot systems. Most faults appear as the change of model parameters or dynamics functions, which can be described by analytic mathematical models or data-driven numerical models. Modeling a test case requires lots of historical data and regulations from UAV authority agencies, which still needs to be studied and is not the main content of this paper. For simplicity, we assume that a test case database has been developed with all concerned failure modes being well modeled based on real experimental data. Each test case should include detailed information as shown in Table 1 for the requirements of automatic testing. The detailed case information should include the vehicle command scripts, the fault injection triggering time, the test stop time, the desired vehicle flight performance, and so on.

Note that, by referring to the airworthiness framework of civil aircraft, the testing items and assessment indexes of UAVs should be obtained from the regulations, guidelines, or standards from the authorities, so that the test results can be applied to the future safety assessment of UAV autopilot systems. Then the test case databases can be obtained from these testing items and indexes, where the logically related test cases can be described by formal modeling language (e.g., UML, SysML), and the process-related testing cases can be modeled in the Simulink Requirements toolbox.

Note that, by referring to the airworthiness framework of civil aircraft, the testing items and assessment indexes of UAVs should be obtained from the regulations, guidelines, or standards from the authorities, so that the test results can be applied to the future safety assessment of UAV autopilot systems. Then the test case databases can be obtained from these testing items and indexes, where the logically related test cases can be described by formal modeling language (e.g., UML, SysML), and the process-related testing cases can be modeled in the Simulink Requirements toolbox.

3.4.2. Automatic testing and assessment framework

An automatic safety testing framework (see Fig. 12) has been developed based on the proposed HIL test platform. In each testing case, the autopilot system and the simulation models are automatically reinitialized to default states, and then the autopilot system automatically controls the vehicle model in the HIL test platform to the desired state. At this moment, a fault case is injected to the HIL test platform to simulate a vehicle failure. After a period of time, the vehicle and autopilot system states are automatically analyzed and recorded to a report. Finally, the next testing case is tested in the same way until all the testing cases are tested and evaluated.

By comparing the flight data (true states from simulation computer and estimated states from autopilot) after fault injection with the desired performance as presented in Table 1, the safety and reliability can be assessed by quantitative or qualitative analysis methods by certification authorities. For example, assuming that the safety specification requires that a multicopter UAV should be able to remain hovering flight with altitude fluctuation $\leq 10°$ after one motor is failed if the tested autopilot drives the multicopter to crash on the ground with maximum attitude fluctuation = 90°, then the safety test is failed and the designers should modify the fail-safe algorithms.

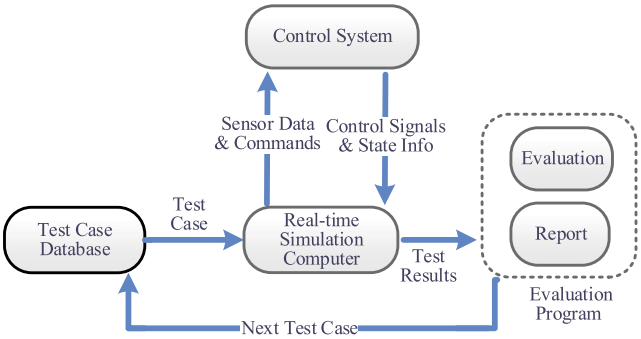


Fig. 12. Automatic safety testing and assessment framework.

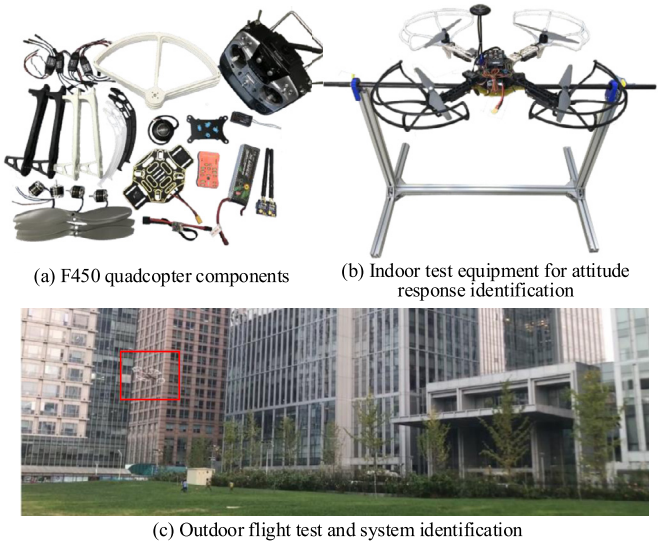
4. Verification and application

In this section, the real-time HIL test platform is applied to open-source autopilots of small UAVs. Then, the simulation credibility is verified through a series of experiments. In the end, several successful applications are presented to verify the feasibility and practicability of the proposed methods in this paper.

4.1. Platform verification

4.1.1. Platform function verification

The Pixhawk autopilot supports for running different types of flight control software systems (e.g., PX4, Ardupilot, and other embedded control software systems) in it, and also supports for controlling different types of unmanned vehicles (e.g., multicopters, fixed-wing aircraft, or even small cars). Besides, the Pixhawk autopilot has a series of available hardware configurations for certain performance requirements, such as Pixhawk 1, Pixhawk 4, etc. To verify the proposed modeling method and test platform, we first apply the proposed unified modeling method in Section 2 to develop the simulation models for different types of UAVs. Then, these simulation models are deployed to the HIL test platform with the MBD framework presented in Section 3. Finally, a series of tests are performed for the Pixhawk systems with various combinations of hardware configurations, software systems, and vehicle types. In our experimental tests, the four advantages of the proposed method (including extensibility, comprehensiveness, verification, and standardization) concluded in Section 3.2 are verified with the proposed modeling method and simulation test platform.



(a) F450 quadcopter components (b) Indoor test equipment for attitude response identification

(c) Outdoor flight test and system identification

Fig. 13. Verification equipment for the proposed HIL test platform.

Owing to the adoption of FPGA for sensor product simulations, the proposed RflySim platform has also been successfully applied to test different UAV autopilot hardware products (e.g., Pixhawk series and self-designed autopilot boards) in our experiments by changing the sensor chip models. Given space limitations, a quadcopter autopilot system will be selected as the representative tested object in this section to perform quantitative verification for the proposed methods. The autopilot system is the most widely used open-source autopilot system for small-scale unmanned vehicles, and the detailed configuration is Pixhawk 1 hardware (MCU: STM32F427, sensor: MPU6000, MS5611, LSM303D, L3GD20H, Ublox-M8N, etc.) with the PX4 control software. The quadcopter UAV is selected because it is the most representative vehicle type that covers the model characteristics (e.g., aerodynamics, ground collision, kinematics, and dynamics) and operating environments (e.g., near-ground, mid-air, indoor, outdoor, hovering flight, and forward flight) of most UAVs.

4.1.2. Experimental verification for simulation credibility

The experimental setup is presented in Fig. 13, where an F450 quadcopter airframe (diagonal length: 450 mm, weight: 1.4 kg, propulsion system: DJI E310, battery: LiPo 3S 4000 mAh) is selected as the experimental subject whose component diagram is shown in Fig. 13(a). To test the dynamics and aerodynamics of quadcopter, an indoor test bench as shown in Fig. 13(b) is developed with the quadcopter fixed to a stiff stick (through the center of mass) with smooth bearings to minimize friction. The quadcopter is free to smoothly rotating along one axis, which makes it possible to perform sweep-frequency testing for system identification and uniform rotation testing for roll damping coefficient measurement. Fig. (c) presents the outdoor flight test scenario, where the body aerodynamic parameters and overall performance of multicopters are measured or obtained by system identification methods. Besides, we had also proposed other experimental equipment [39] to certificate the modeling accuracy of UAV propulsion systems. For example, Fig. 14 presents the result comparison between experiments and simulations of the tested quadcopter propulsion system, where it can be observed that the proposed simulation system has high accuracy from a qualitative point of view.

Based on the above experiment setup, the quantitative simulation credibility assessment method proposed in our previous work [17] is applied in this paper to assess and improve the simulation credibility of the HIL platform. The key idea is to use the same autopilot system (see Fig. 15) in both simulations and experiments, then the simulation errors caused by hardware and software differences of control systems can be controlled to the utmost extent, which significantly improves the credibility of the simulation platform compared with other simulation methods. The experiment results in [17] demonstrate that the proposed FPGA-based HIL simulation platform can reach a high matching degree (the credibility index in [17]) larger than 90% (where 60% presents the minimum accuracy requirement, and 100% presents a perfect match) by analyzing the results between the test platform and real experimental system from the quantitative perspective.

4.1.3. Videos and source code

Four videos have been published to demonstrate the development, experimental verification, and application process in this section for the proposed platform. The first video gives an overall introduction of the proposed platform along with several applications on different multicopter autopilot systems:

https://youtu.be/nXAoLdPzz_I

The second video presents several demos of applying the proposed HIL platform to different types of vehicles with complex traffic environment simulation:

https://youtu.be/xQUnkqH29qU

The third video presents the automatic fault injection, safety testing, and safety assessment demos:

https://youtu.be/MHieyE3hbHY

The fourth video introduces the MBD modeling and development process of the platform with experiments to quantitatively verify its simulation credibility:

https://youtu.be/ChNtkb5rrQs

We have also published the MATLAB/Simulink source code and the detailed modeling tutorial to Github:

https://github.com/RflySim/CopterSim

Readers can use it to rapidly develop SIL or HIL simulation systems for different types of unmanned vehicles by modifying the aerodynamic model and the actuator model.

4.2. Method applications

This subsection presents several successful applications with the proposed test platform to increase the development, testing, and validation efficiency of UAVs.

4.2.1. Rapid prototyping

To take maximum advantage of the proposed test platform with MBD methods, a component model database is developed for the rapid development of electric multicopters. The database covers the common products on the market for multicopter propulsion systems with model parameters obtained by their product specifications and experimental data. With this model database, an online toolbox is released (URL: https://flyeval.com/) based on our previous studies [39–41] for the automatic design and performance estimation of multicopter UAVs, and a screenshot of the online toolbox is presented in Fig. 16. Users can select component products from the database to quickly assemble a multicopter to estimate its flight performance and model parameters. The toolbox has been released for more than two years, and the user feedback indicates that it can significantly improve the multicopter model development efficiency with decent simulation accuracy.

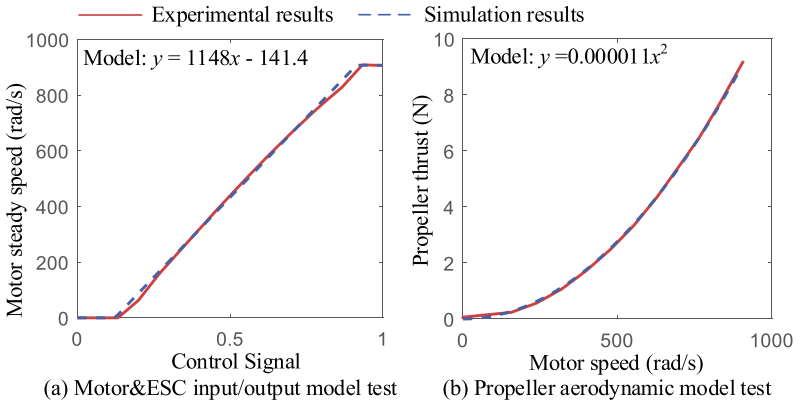


Fig. 14. Modeling accuracy verification of propulsion system with experiments and simulations.

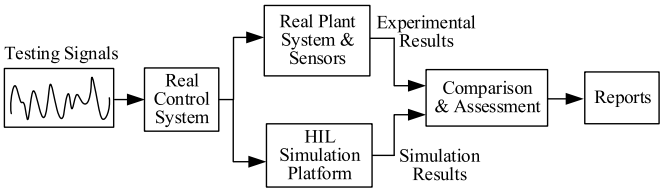


Fig. 15. Simulation credibility assessment experiments [17].

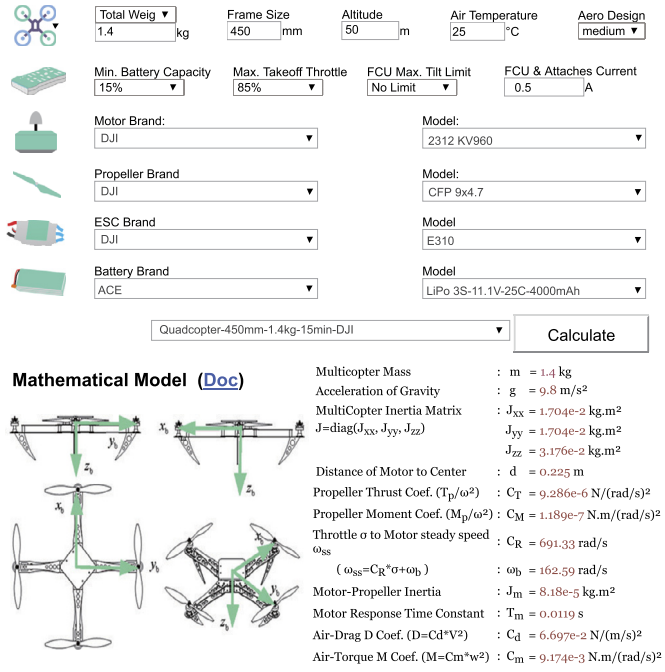


Fig. 16. Screenshot of the online toolbox flyeval.com.

4.2.2. Algorithm comparison

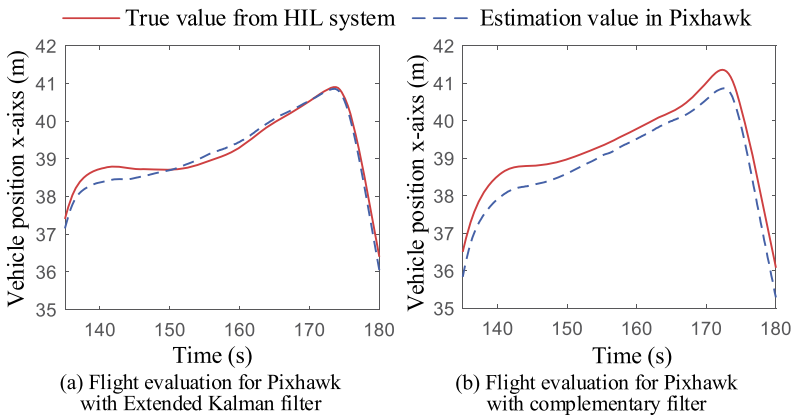
Another important advantage of the proposed HIL test platform is that it can obtain the true states of the simulated vehicle, which is significant for comparing performance differences of control algorithms. In Fig. 17, two simulations are performed on the HIL test platform with two different estimation filter algorithms in the Pixhawk autopilot. They are the Extended Kalman filter algorithm in Fig. 17(a) and the complementary filter algorithm in Fig. 17(b), respectively. It can be observed from the result in Fig. 17 that the extended Kalman filter has a better estimation effect than complementary, which is consistent with the theoretical analysis. This conclusion is hard to obtain through experiments because a higher more precise external measuring devices (e.g., differential GPS or visual positioning systems with centimeter-level precision) are required to measure the true states of the vehicle. These external measuring devices are usually expensive and restrained. For example, the differential GPS is easy to be disturbed by flight environment factors, and its data frequency is too low (usually 5 Hz), and the visual positioning system cannot be used outdoors. Therefore, the proposed test platform is a better way to acquire the true states of the vehicle for comparative analysis and performance assessment.

4.2.3. Normal flight tests

The proposed HIL test platform makes it possible to comprehensively test the autopilot system indoor only with computers, which is significant in reducing cost and time relative to outdoor experiments. Fig. 18 presents an autonomous mission flight test with the proposed HIL test platform, which usually should be performed by outdoor flight tests in traditional test methods. Besides, more comparative simulation tests and experiments demonstrate that all flight tests (indoor or outdoor, manual or automatic) can be tested on the HIL test platform if the vehicle modeling is comprehensive and accurate enough.

4.2.4. Automatic fault injection tests

An automatic fault injection case (one motor blocked) is applied to the experimental set in Fig. 18 during the autonomous mission flight state, where the autopilot control command sequences are similar to Table 1. The recorded data from the simulation computer (truth values) and the autopilot (the estimated states based on sensor data) are shown in Fig. 19, where the altitude, velocity, and roll angle, and autopilot output curves are presented in 0-8 test phases. The results indicate that: i) the quadcopter is losing control (according to the autopilot output curves) and crashed on the ground (according to the altitude curve) when one motor is blocked, which means the fault is not recognized and handled by the fail-safe logic of the autopilot; ii) the ground impact speed and attitude fluctuation are exceed expected safe indexes (5 m/s and 10 degrees for impact speed and attitude requirements in Table 1), so the fault may hurt people on the ground and the safety test is failed; iii) the simulated motor failure effect (losing control and crashed) is consistent with our experimental results whose test process is far more dangerous and high-cost than the HIL simulation; iv) according to our previous study [42], the crash can be avoided if the motor fault is recognized and well controlled by the autopilot, which may help to improve the safety level of the multicopter UAV systems; v) the estimated states by autopilot are quite different from the truth values when the fault is injected in Phase 6-7 of Fig. 19, which means the data logged by the control system may not be accurate after fault injection, and the truth values from



(a) Flight evaluation for Pixhawk with Extended Kalman filter

(b) Flight evaluation for Pixhawk with complementary filter

Fig. 17. Comparing estimation performance of different filter algorithms in turning flight stage.

(a) Simulation Computer & Pixhawk

(b) Front Camera View from 3D Engine

(c) Automatic Mission Flight with Ground Control Station

Fig. 18. Autonomous mission flight test with the proposed HIL test platform.

the proposed HIL simulation platform provide more accurate data for quantitative safety assessment. Since all test and assessment procedures are automatically finished by the proposed automatic test platform, the test efficiency is significantly improved during our actual testing process, and the safety level of UAV systems is improved because more potential failures are revealed and quickly fixed by fail-safe logic of autopilot systems.

5. Conclusion and future work

This paper presents a unified simulation and test platform, aiming to significantly improve the development speed and safety level of unmanned vehicle autopilot systems. A unified modular modeling framework is proposed by abstracting the common features of different types of unmanned vehicles. The application examples demonstrate this framework is efficient in developing a vehicle simulation model with compatibility for the future safety assessment and certification standards. Another key problem solved in the paper is to develop a HIL simulation test platform to ensure simulation credibility. With the model-based design method, the developing process of the simulation software can be automatized and standardized, which ensures different developers can obtain the same simulation software with credibility guaranteed by the automatic code generation tools. With the FPGA-based real-time hardware-in-the-loop simulation technology, the operating environment of the control algorithms is guaranteed by using the same autopilot system in both experiments and simulations. After the credibility of the simulation test platform is well guaranteed, we can focus on verifying and validating the simulation models with the quantitative assessment method proposed in our previous work. The proposed test platform is applied to a multicopter autopilot system, where the accuracy and fidelity of the simulation testing are verified by comparing the results with experiments. The successful applications present the advantages in the multicopter

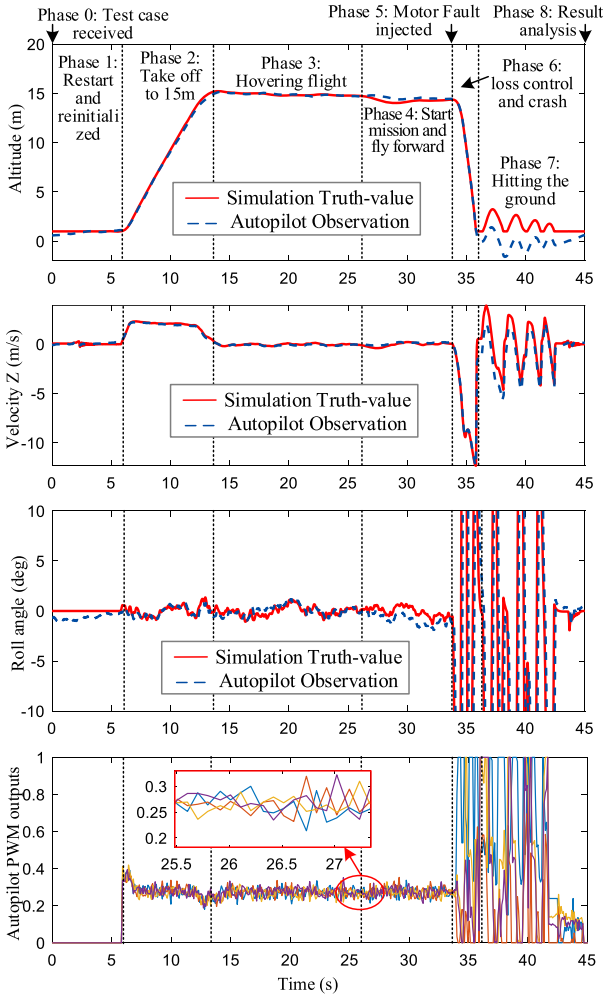


Fig. 19. Automatic test results of one motor failed case during the flight mission.

rapid prototyping, estimation algorithm verification, autonomous flight testing, and automatic safety testing with automatic fault injection and result evaluation of unmanned vehicles.

Since the test platform can provide high-fidelity and credibility simulation results, it will help to improve the training efficiency of artificial intelligence algorithms. Besides, the airworthiness for unmanned aerial vehicles requires more formal, quantitative, and efficient testing methods to assess the safety level, and we will apply the proposed test platform for the safety assessment of UAV systems. It is also an interesting research topic to design test cases for the automatic testing platform to achieve higher testing efficiency and coverage.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This work was supported by the National Natural Science Foundation of China under Grant 61973015, and by the Aeronautics Power Foundation under Grant 6141B09050377.

References

[1] M. Lungu, Backstepping and dynamic inversion combined controller for auto-landing of fixed wing UAVs, Aerosp. Sci. Technol. 96 (2020) 105526, https://doi.org/10.1016/j.ast.2019.105526.
[2] Y. Wang, Y. Zhou, C. Lin, Modeling and control for the mode transition of a novel tilt-wing UAV, Aerosp. Sci. Technol. 91 (2019) 593–606, https://doi.org/10.1016/j.ast.2019.05.046.
[3] N.A. Vu, D.K. Dang, T.L. Dinh, Electric propulsion system sizing methodology for an agriculture multicopter, Aerosp. Sci. Technol. 90 (2019) 314–326, https://doi.org/10.1016/j.ast.2019.04.044.
[4] X. Dai, Q. Quan, J. Ren, K.-Y. Cai, Iterative learning control and initial value estimation for probe-drogue autonomous aerial refueling of UAVs, Aerosp. Sci. Technol. 82–83 (2018) 583–593, https://doi.org/10.1016/j.ast.2018.09.034.
[5] H. Ji, R. Chen, P. Li, Real-time simulation model for helicopter flight task analysis in turbulent atmospheric environment, Aerosp. Sci. Technol. 92 (2019) 289–299, https://doi.org/10.1016/j.ast.2019.05.066.
[6] H. Lipson, M. Kurman, Driverless: Intelligent Cars and the Road Ahead, Mit Press, 2016.
[7] C.M. Belcastro, D.H. Klyde, M.J. Logan, R.L. Newman, J.V. Foster, Experimental flight testing for assessing the safety of unmanned aircraft system safety-critical operations, in: 17th AIAA Aviation Technology, Integration, and Operations Conference, AIAA 2017-3274, 2017.
[8] S.S. Noureen, V. Roy, S.B. Bayne, An overall study of a real-time simulator and application of RT-LAB using MATLAB simpowersystems, in: 2017 IEEE Green Energy and Smart Systems Conference, 2018, pp. 1–5.
[9] M.B. Tischler, System identification methods for aircraft flight control development and validation, in: Advances in Aircraft Flight Control, 2018, pp. 35–69.
[10] O. Lisagor, T. Kelly, R. Niu, Model-based safety assessment: review of the discipline and its challenges, in: ICRMS'2011 - Safety First, Reliability Primary: Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety, 2011, pp. 625–632.
[11] D. Jung, P. Tsiotras, Modeling and hardware-in-the-loop simulation for a small unmanned aerial vehicle, in: AIAA Infotech@Aerospace 2007 Conference and Exhibit, AIAA 2018-2768, May 2007.
[12] D.B. Worth, B.G. Woolley, D.D. Hodson, SwarmSim: a framework for modeling swarming unmanned aerial vehicles using hardware-in-the-loop, J. Defense Model. Simul., Appl. Methodol. Technol. (2017) 1–20.
[13] H. Saad, T. Ould-Bachir, J. Mahseredjian, C. Dufour, S. Dennetiere, S. Nguefeu, Real-time simulation of MMCs using CPU and FPGA, IEEE Trans. Power Electron. 30 (1) (2015) 259–267.
[14] S. Mikkili, A.K. Panda, J. Prattipati, Review of real-time simulator and the steps involved for implementation of a model from MATLAB/SIMULINK to real-time, J. Inst. Eng. (India), Ser. B 96 (2) (2015) 179–196, https://doi.org/10.1007/s40031-014-0128-6.
[15] U.B. Mehta, D.R. Eklund, V.J. Romero, J.A. Pearce, N.S. Keim, Simulation credibility: Advances in verification, validation, and uncertainty quantification, Tech. Rep. jannaf/gl-2016-0001, NASA Ames Research Center, Moffett Field, CA, United States, 01 Nov 2016.
[16] I. MathWorks, Model-based design - MATLAB & Simulink, https://www.mathworks.com/solutions/model-based-design.html. (Accessed 24 May 2019).
[17] X. Dai, C. Ke, Q. Quan, K.-Y. Cai, Simulation credibility assessment methodology with FPGA-based hardware-in-the-loop platform, IEEE Trans. Ind. Electron. 68 (4) (2021) 3282–3291, https://doi.org/10.1109/TIE.2020.2982122.
[18] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: high-fidelity visual and physical simulation for autonomous vehicles, in: Field and Service Robotics, 2018, pp. 621–635.
[19] W. Guerra, E. Tal, V. Murali, G. Ryou, S. Karaman, FlightGoggles: photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019, pp. 6941–6948.
[20] A.M. Miller, R. Alvarez, N. Hartman, Towards an extended model-based definition for the digital twin, Comput-Aided Des. Appl. 15 (6) (2018) 880–891, https://doi.org/10.1080/16864360.2018.1462569.
[21] A. Aminzadeh, M.A. Atashgah, A. Roudbari, Software in the loop framework for the performance assessment of a navigation and control system of an unmanned aerial vehicle, IEEE Aerosp. Electron. Syst. Mag. 33 (1) (2018) 50–57.
[22] M.A. Atashgah, S. Malaek, A simulation environment for path and image generation in an aerial single-camera vision system, proceedings of the institution of mechanical engineers, part G, J. Aerosp. Eng. 225 (5) (2011) 541–558.
[23] M.A. Atashgah, S. Malaek, Prediction of aerial-image motion blurs due to the flying vehicle dynamics and camera characteristics in a virtual environment, proceedings of the institution of mechanical engineers, part G, J. Aerosp. Eng. 227 (7) (2013) 1055–1067.
[24] B.L. Stevens, F.L. Lewis, Aircraft Control and Simulation, 2nd edition, Wiley, New Jersey, 2004.
[25] Q. Quan, Introduction to Multicopter Design and Control, Springer, Singapore, 2017.
[26] R.W. Smith, Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems, Defense Mapping Agency, 1987.

[27] M. Cavcar, The international standard atmosphere (ISA), vol. 30, Anadolu University, Turkey, 2000, p. 9.
[28] A. Chulliat, S. Macmillan, P. Alken, C. Beggan, M. Nair, B. Hamilton, A. Woods, V. Ridley, S. Maus, A. Thomson, The US/UK world magnetic model for 2015-2020, BGS and NOAA, 2015.
[29] D.J. Moorhouse, R.J. Woodcock, Background information and user guide for MIL-F-8785C, military specification-flying qualities of piloted airplanes, Tech. Rep. AFWAL-TR-81-3109, U.S. Air Force Wright Aeronautical Labs, Wright-Patterson AFB, OH, July 1982.
[30] R.K. Remple, M.B. Tischler, Aircraft and rotorcraft system identification: engineering methods with flight-test examples, AIAA J. (2006).
[31] G. Cai, B.M. Chen, T.H. Lee, Unmanned Rotorcraft Systems, Springer Science & Business Media, 2011.
[32] R. Rajamani, Vehicle Dynamics and Control, second edition, Springer Science & Business Media, 2011.
[33] Y. Chen, S. Chen, T. Zhang, S. Zhang, N. Zheng, Autonomous vehicle testing and validation platform: integrated simulation system with hardware in the loop, in: 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 949–956.
[34] A.I. Hentati, L. Krichen, M. Fourati, L.C. Fourati, Simulation tools, environments and frameworks for UAV systems performance analysis, in: 2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018, 2018, pp. 1495–1500.
[35] R.W. Beard, T.W. McLain, Small Unmanned Aircraft: Theory and Practice, Princeton University Press, 2012.
[36] M. Kučiš, P. Zemčík, Simulation of camera features, in: Proceedings of the 16th Central European Seminar on Computer Graphics, 2012, pp. 117–123.
[37] F. Leens, An introduction to I2C and SPI protocols, IEEE Instrum. Meas. Mag. 12 (1) (2009) 8–13.
[38] S. Gage, NASA HL-20 lifting body airframe modeled with Simulink and the aerospace blockset, https://www.mathworks.com/help/aeroblks/nasa-hl-20-lifting-body-airframe.html. (Accessed 24 May 2019).
[39] D. Shi, X. Dai, X. Zhang, Q. Quan, A practical performance evaluation method for electric multicopters, IEEE/ASME Trans. Mechatron. 22 (3) (2017) 1337–1348, https://doi.org/10.1109/TMECH.2017.2675913.
[40] X. Dai, Q. Quan, J. Ren, K.-Y. Cai, An analytical design-optimization method for electric propulsion systems of multicopter UAVs with desired hovering endurance, IEEE/ASME Trans. Mechatron. 24 (1) (2019) 228–239, https://doi.org/10.1109/TMECH.2019.2890901.
[41] X. Dai, Q. Quan, J. Ren, K.-Y. Cai, Efficiency optimization and component selection for propulsion systems of electric multicopters, IEEE Trans. Ind. Electron. 66 (10) (2019) 7800–7809, https://doi.org/10.1109/TIE.2018.2885715.
[42] Q. Quan, X. Dai, S. Wang, Multicopter Design and Control Practice: A Series Experiments based on MATLAB and Pixhawk, Springer, Singapore, 2020.