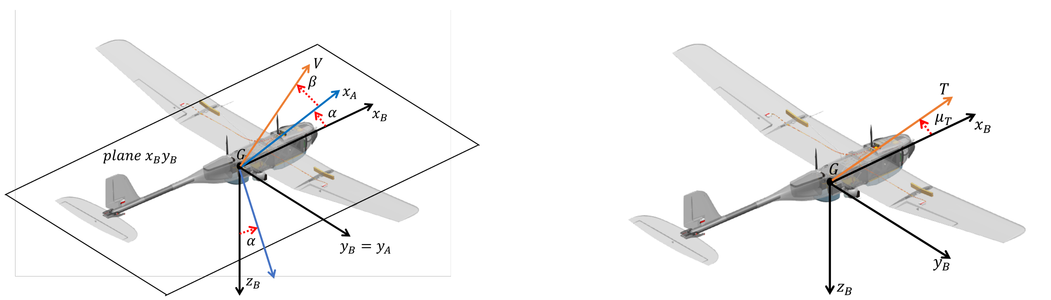
A Details of Platform

A.1 Flight Dynamics Model

The 6-DoF atmospheric dynamics of a rigid aircraft are described by a set of standard nonlinear ordinary differential equations, which are not detailed here for brevity; interested readers are referred to [9] [16]. This model differentiates between a ground-based inertial frame and an aircraft-based reference frame. The ground-based frame $\mathcal{F}_E = \{O_E; x_E, y_E, z_E\}$ is inertial, ignoring Earth's rotational effects, which is a valid assumption for low-altitude flight. The frame's origin is fixed at point O_E on the ground, with x_E pointing north, y_E east, and z_E downwards. This is also known as the NED (North-East-Down) frame. The aircraft body-fixed frame $\mathcal{F}_B = \{G; x_B, y_B, z_B\}$ originates at the aircraft's center of gravity G. Here, x_B aligns with the fuselage pointing forward, y_B points rightward, and z_B downward.

The motion equations are derived from Newton's second law for an air vehicle, resulting in six core scalar equations (conservation of linear and angular momentum in $\mathcal{F}_B$), flight path equations (for tracking the aircraft's center-of-gravity relative to $\mathcal{F}_E$), and rigid-body kinematic equations (defining the aircraft's attitude quaternion to describe the body axes orientation relative to the inertial ground frame).



(a) Aerodynamic angles, aerodynamic (or stability) frame

(b) Thrust vector, thrust magnitude T , thrust line angle μ_T

Figure 8: Fixed-Wing aircraft flight dynamics model

The conservation of linear momentum equations (CLMEs) for a rigid aircraft with constant mass can be expressed by the following three fundamental scalar equations 1:

$$\dot{u} = rv - qw + \frac{1}{m}\left(W_x + F_x^{(A)} + F_x^{(T)}\right) \tag{1a}$$

$$\dot{v} = -ru + pw + \frac{1}{m}\left(W_y + F_y^{(A)} + F_y^{(T)}\right) \tag{1b}$$

$$\dot{w} = qu - pv + \frac{1}{m}\left(W_z + F_z^{(A)} + F_z^{(T)}\right) \tag{1c}$$

where $\mathbf{W}$ represents the aircraft's weight, $F^{(A)}$ denotes the aerodynamic forces, and $F^{(T)}$ stands for the thrust forces. These forces are decomposed into body frame components $\mathcal{F}_B$ for simplicity in deriving Eqs. 1a, 1b, 1c.

The weight force, always aligned with the inertial z_E axis, is mg and its components in the body frame are given by:

$$\begin{Bmatrix} W_x \\ W_y \\ W_z \end{Bmatrix} = [T_{BE}] \begin{Bmatrix} 0 \\ 0 \\ mg \end{Bmatrix} = \begin{Bmatrix} 2(q_z q_x - q_0 q_y) \\ 2(q_y q_z + q_0 q_x) \\ q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{Bmatrix} mg \tag{2}$$

The matrix $[T_{BE}]$ describes the direction cosines for the instantaneous attitude of frame $\mathcal{F}_B$ relative to frame $\mathcal{F}_E$. Its entries are functions of the aircraft's attitude quaternion components (q_0, q_x, q_y, q_z) 3:

$$[T_{BE}] = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 2(q_x q_y - q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_0 q_x) \\ 2(q_x q_z + q_0 q_y) & 2(q_y q_z - q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3}$$

The aerodynamic force $F^{(A)}$ acting on the aircraft, projected onto frame $\mathcal{F}_B$, is given by 4:

$$\begin{Bmatrix} F_x^{(A)} \\ F_y^{(A)} \\ F_z^{(A)} \end{Bmatrix} = [T_{BW}] \begin{Bmatrix} -D \\ -C \\ -L \end{Bmatrix} \tag{4}$$

$$\begin{Bmatrix} F_x^{(A)} \\ F_y^{(A)} \\ F_z^{(A)} \end{Bmatrix} = \begin{bmatrix} -D\cos\alpha\cos\beta + L\sin\alpha + C\cos\alpha\sin\beta \\ -C\cos\beta - D\sin\beta \\ -D\sin\alpha\cos\beta - L\cos\alpha + C\sin\alpha\sin\beta \end{bmatrix} \tag{5}$$

The aerodynamic drag D, cross force C, and lift L account for the effects of external airflow. The coordinate transformation matrix $[T_{BW}]$ from the standard wind frame $\mathcal{F}_W = \{G; x_W, y_W, z_W\}$ to $\mathcal{F}_B$ is given by:

$$[T_{BW}] = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Equations 1a, 1b, 1c are expressed in closed form since the aerodynamic angles (α, β) and force components (D, C, L) are functions of the aircraft's state variables and external conditions. According to Figure 8a, the state variables (u, v, w), which are components of the aircraft's velocity vector $\mathbf{V}$ in $\mathcal{F}_B$, are related to (α, β) as follows:

$$u = V\cos\beta\cos\alpha \tag{7a}$$

$$v = V\sin\beta \tag{7b}$$

$$w = V\cos\beta\sin\alpha \tag{7c}$$

where

$$V = \sqrt{u^2 + v^2 + w^2} \tag{8}$$

The instantaneous angles of attack and sideslip are given by:

$$\alpha = \tan^{-1}\frac{w}{u}, \quad \beta = \sin^{-1}\frac{v}{\sqrt{u^2 + v^2 + w^2}} \tag{9}$$

The aerodynamic forces are described using their aerodynamic coefficients in the following standard formulas:

$$D = \frac{1}{2}\rho V^2 S C_D, \quad C = \frac{1}{2}\rho V^2 S C_C, \quad L = \frac{1}{2}\rho V^2 S C_L \tag{10}$$

where the air density ρ depends on the flight altitude $h = -z_{E,G}$ and other atmospheric properties like the sound speed a [53]. S represents a reference area, while the coefficients (C_D, C_C, C_L) vary with the aircraft's state and external inputs.

Finally, as shown in Figure 8b, the thrust force $F^{(T)}$ of magnitude T is expressed in the body-frame components as follows:

$$\begin{Bmatrix} F_x^{(T)} \\ F_y^{(T)} \\ F_z^{(T)} \end{Bmatrix} = \delta_T T_{\text{max}}(h, M) \begin{Bmatrix} \cos\mu_T \\ 0 \\ \sin\mu_T \end{Bmatrix} \tag{11}$$

where μ_T is a constant angle between the thrust line and the reference axis x_B in the aircraft's symmetry plane. The thrust $T = \delta_T T_{\text{max}}(h, M)$, where δ_T is the throttle setting (an external input), and $T_{\text{max}}(h, M)$ is the maximum thrust available, dependent on altitude and Mach number $M = V/a$.

The conservation of angular momentum equations (CAMEs) for a rigid aircraft with constant mass are given by [9]:

$$\dot{p} = (C_1 r + C_2 p)q + C_3 L + C_4 N \tag{12a}$$

$$\dot{q} = C_5 pr - C_6(p^2 - r^2) + C_7 M \tag{12b}$$

$$\dot{r} = (C_8 p - C_2 r)q + C_4 L + C_9 N \tag{12c}$$

where

$$C_1 = \frac{1}{\Gamma}[(I_{yy} - I_{zz})I_{zz} - I_{xz}^2], \tag{13a}$$

$$C_2 = \frac{1}{\Gamma}[(I_{xx} - I_{yy} + I_{zz})I_{xz}], \tag{13b}$$

$$C_3 = \frac{I_{zz}}{\Gamma}, \quad C_4 = \frac{I_{xz}}{\Gamma}, \quad C_5 = \frac{I_{zz} - I_{xx}}{I_{yy}}, \tag{13c}$$

$$C_6 = \frac{I_{xz}}{I_{yy}}, \quad C_7 = \frac{1}{I_{yy}}, \tag{13d}$$

$$C_8 = \frac{1}{\Gamma}[(I_{xx} - I_{yy})I_{xx} + I_{xz}^2], \quad C_9 = \frac{I_{xx}}{\Gamma} \tag{13e}$$

and $\Gamma = I_{xx}I_{zz} - I_{xz}^2$ are constants derived from the aircraft's inertia matrix relative to the axes of $\mathcal{F}_B$.

The systems of equations 1, 12 for CLMEs and CAMEs projected onto the moving frame $\mathcal{F}_B$ must be supplemented with additional equations to fully describe the aircraft dynamics and evolve its state over time. One such set of equations is the flight path equations (FPEs), which describe the aircraft's trajectory relative to the Earth-based inertial frame. These equations yield the instantaneous position $\{x_{E,G}(t), y_{E,G}(t), z_{E,G}(t)\}$ of the aircraft's center of gravity G in $\mathcal{F}_E$. The 2D version $\{x_{E,G}(t), y_{E,G}(t)\}$ of the FPEs defines the ground track relative to the aircraft's flight path.

The flight path equations (FPEs) are derived by transforming the vector $\mathbf{V}$ from frame $\mathcal{F}_B$ to frame $\mathcal{F}_E$:

$$\begin{Bmatrix} \dot{x}_{E,G} \\ \dot{y}_{E,G} \\ \dot{z}_{E,G} \end{Bmatrix} = [T_{EB}] \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{14}$$

with $[T_{EB}] = [T_{BE}]^{\mathrm{T}}$ as defined in equation 3. The matrix form of the FPEs is:

$$
\begin{Bmatrix} \dot{x}_{E,G} \\ \dot{y}_{E,G} \\ \dot{z}_{E,G} \end{Bmatrix} = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 2(q_x q_y - q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_0 q_x) \\ 2(q_x q_z + q_0 q_y) & 2(q_y q_z - q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{15}
$$

The inputs for the FPEs are the aircraft's attitude quaternion components along with the components (u, v, w), which are derived from the combined CLMEs and CAMEs system.

The rigid-body kinematic equations (KEs) using the aircraft's attitude quaternion components [9] are expressed in matrix form as:

$$
\begin{Bmatrix} \dot{q_0} \\ \dot{q_x} \\ \dot{q_y} \\ \dot{q_z} \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{Bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{Bmatrix} \tag{16}
$$

The inputs to these KEs are the angular velocity components (p, q, r) in $\mathcal{F}_B$, and solving these equations provides the kinematic state variables (q_0, q_x, q_y, q_z).

The system comprising (CLMEs)-(CAMEs)-(FPEs)-(KEs), i.e., 1, 12, 15, and 16, represents a complete set of 13 coupled nonlinear differential equations that describe the 6-DoF rigid-body dynamics of atmospheric flight. These equations are in closed form once the aerodynamic and propulsive external forces and moments are fully modeled as functions of the 13 state variables:

$$
\mathbf{x} = [u, v, w, p, q, r, x_{E,G}, y_{E,G}, z_{E,G}, q_0, q_x, q_y, q_z]^{\mathrm{T}} \tag{17}
$$

This state vector $\mathbf{x}$, along with various external inputs grouped into an input vector, commonly referred to as $\mathbf{u}$, fully characterizes the system.

The F-16 public domain model utilized in this study includes a sophisticated and high-fidelity flight control system (FCS). The FCS, which incorporates state feedback from the aircraft dynamics block, consists of the following channels: (i) Roll command δ_a (affecting right aileron deflection angle δ_a and antisymmetric left aileron deflection), (ii) Pitch command δ_e (controlling elevon deflection angle δ_e), (iii) Yaw command δ_r (manipulating rudder deflection angle δ_r), (iv) Throttle lever command δ_T (adjusting throttle setting δ_T and enabling jet engine afterburner).

A.2 Task Scenarios

The task scenarios can be categorized by objectives into *Heading, Control, and Tracking*. (1) *Heading*: The objective is to control the fixed-wing aircraft to reach a predetermined altitude, yaw angle, and speed within a specified time. This task serves as the foundation for multi-aircraft collaboration and pursuit tasks. (2) *Control*: The objective is to control the fixed-wing aircraft to reach a predetermined pitch angle, yaw angle, and speed within a specified time. This task serves as the fundamental control basis for fixed-wing aircraft trajectory tracking. (3) *Tracking*: The objective is to control the fixed-wing aircraft to reach a predetermined coordinate position (in the geocentric coordinate system) within a specified time. This work designs a hierarchical control algorithm for this task. The lower-level controller is capable of completing the Control task, while the upper-level planner algorithm aims to achieve the overall task objective. This task forms the basis for performing aerobatic maneuvers with fixed-wing aircraft.

The task scenarios can also be categorized by flight conditions into HighSpeed, HighAltitude, Windy, and Noisy. (1) *HighSpeed*: Control of high-maneuverability flight of fixed-wing aircraft under high-speed conditions (speed exceeding Mach 1). (2) *HighAltitude*: Control of high-maneuverability flight of fixed-wing aircraft under high-altitude conditions (altitude exceeding 30,000 feet). (3) *Windy*: Control of high-maneuverability flight of fixed-wing aircraft under windy conditions. (4) *Noisy*:

Control of high-maneuverability flight of fixed-wing aircraft when there is noise in the observation measurements.

We design different environment rewards for different task objectives. For the Heading and Tracking tasks, the environment reward is the negative Euclidean norm (L2 norm) error between the current state and the target state. For the Control task, the environment reward is the negative optimal rotation angle from the current attitude to the target attitude. We also designed various termination conditions and terminal rewards for different tasks, as shown in Table 5.

Table 5: Termination conditions and terminal rewards for different tasks.

Name	Key Value	Description	Terminal reward
ExtremeState	AOA, AOS	AOA and AOS exceeding limit ranges.	-200
HighSpeed	TAS	speed exceeding Mach 3.	-200
LowAltitude	altitude	altitude falling below 2500 feet.	-200
LowSpeed	TAS	speed falling below Mach 0.01.	-200
overload	G	G exceeding 10.	-200
UnreachTarget	$\mathbf{x_t} - \mathbf{x_{target}}$	the target is not reached.	-200
ResetTarget	$\mathbf{x_t} - \mathbf{x_{target}}$	the target is successfully reached.	200

A.3 Baseline Libraries

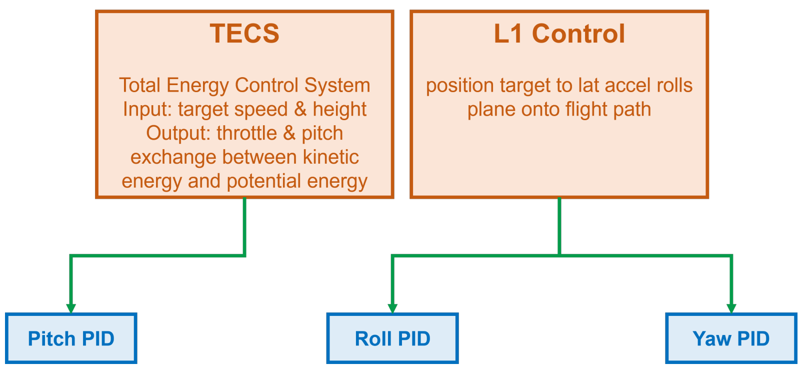


Figure 9: The control system structure for traditional methods.

Traditional Methods These are based on open-source fixed-wing aircraft control algorithms from the Ardupilot platform, using a hierarchical control approach. The upper layer includes the TECS controller [51], which manages the aircraft's total flight energy by adjusting throttle and pitch to maintain desired altitude and speed, and the L1 controller [52], which manages the flight path by adjusting roll and yaw to follow waypoints or desired path characteristics. The lower layer consists of an attitude loop controller using a dual-loop PID algorithm to control the aircraft's surfaces and achieve three-axis attitude tracking. The control system structure for traditional methods is shown in Figure 9.

RL Methods We use PPO for Heading and Control tasks in fixed-wing aircraft. For the Tracking task, we use a hierarchical RL method: the upper-level algorithm converts the target location into desired pitch, yaw, and speed, while the lower level uses the trained PPO algorithm to control the aircraft's surfaces. The structure for hierarchical RL method is shown in Figure 10.

The PPO algorithm's parameter settings are as follows: the learning rate is set to 3×10^{-4}, the number of PPO epochs is 16, the clipping parameter is 0.2, the maximum gradient norm is 2, and the entropy coefficient is 1×10^{-3}. Additionally, the hidden layer sizes for the neural networks are set to "128 128", and the recurrent hidden layer size is 128 with a single recurrent layer.

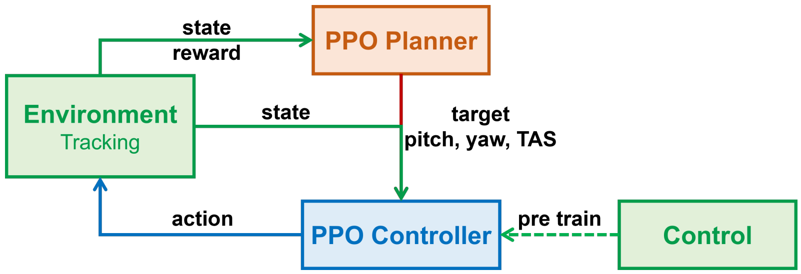


Figure 10: The structure for hierarchical RL method.

A.4 Evaluation Metrics

We provide two types of performance evaluation metrics to assess the algorithm's performance of fixed-wing aircraft control: maneuverability indicators and safety indicators. The complete set of evaluation metrics is shown in Table 6.

Table 6: Performance metrics to assess the algorithm's performance of fixed-wing aircraft control.

Type	Name	Description
Maneuverability Indicators	G	Average G-force during flight.
	TAS	Average True Air Speed during flight.
	RoC	Average Rate of Climb during flight.
	AOA	Average Angle of Attack during flight.
	AOS	Average Angle of Sideslip during flight.
	t	Average time to complete the task objective.
	P	Average roll rate around the body-fixed x-axis.
	Q	Average pitch rate around the body-fixed y-axis.
	R	Average yaw rate around the body-fixed z-axis.
Safety Indicators	ASM	Altitude Safety Margin.
	SSM	Speed Safety Margin.
	OSM	Overload Safety Margin.
	AOASM	Angle of Attack Safety Margin.
	AOSSM	Angle of Sideslip Safety Margin.
	FSM	Smoothness of the aircraft's flight state.

A.5 Code Structure

The overall code framework and workflow of the platform are illustrated in Figure 11. We also provide a complete algorithmic process for training, testing, and evaluating RL algorithms on the platform. Once the appropriate parameters are selected, the platform can automatically execute the algorithm training, testing, and evaluation processes.

Algorithm 1 NeuralPlaneTrainer

Require: User-designed learnable agent A, user-specified FDM M, user-specified task scenario T
Ensure: Trained Agent A, training records

1. Initialize environment with FDM and task scenario $Env = \text{Env\_Initialize}(M, T)$;
2. **while** max learning steps Not reached **do**
3. $\quad A$.train_episode(Env);
4. $\quad$ Record training data;
5. $\quad$ Plot training figures;
6. **end while**
7. Summarize and visualize the training records in *Logger* and return the trained agent;

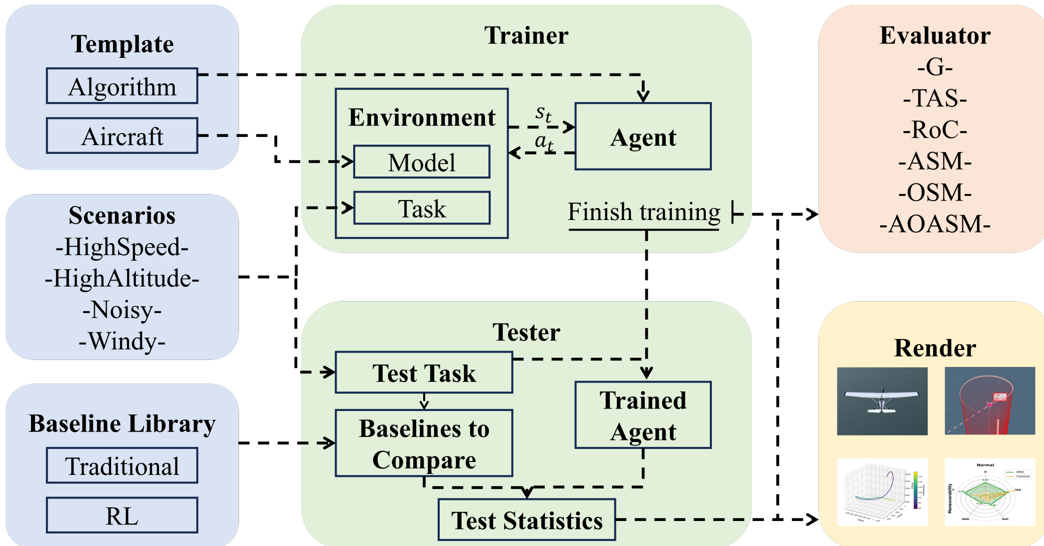


Figure 11: The overall code framework and workflow of NeuralPlane.

Algorithm 2 TrainingEpisode

Require: User-designed learnable agent A, Constructed environment Env
Ensure: Training records

$state = Env$.reset();
while termination condition Not achieved **do**
    $action = A$.get_action$(state)$;
    $next\_state, reward, done, info = Env$.step$(action)$;
    Store transition $\langle state, action, reward, done, next\_state \rangle$;
    Update agent A;
    Record training data and plot figures;
    $state = next\_state$;
end while
Summarize training records and return;

Algorithm 3 NeuralPlaneTester and NeuralPlaneEvaluator

Require: User-specified algorithm set B including baselines and user's trained agent, user-specified FDM M, User-specified task scenario T
Ensure: Testing results

Initialize environment with FDM and task scenario $Env =$ Env_Initialize(M, T);
for each algorithm $alg \in B$ **do**
    alg.rollout_episode(Env);
    Record testing data;
    Plot testing figures;
end for
Summarize testing results and call *Evaluator* for standardized metrics and visualization;

The pseudo-code for the Trainer is detailed in Algorithm 1. With a user-designed learnable agent A, a user-specified FDM M, and a user-specified task scenario T, the NeuralPlane first initializes the environment by determining task scenario and FDM. MetaBox then iteratively trains on each instance until the maximum learning steps are reached. For each instance, agent A calls the `train_episode()` function to interact with Env and perform the training. All training logs are managed by the *Logger*.

Next, we focus on the `train_episode()` function. In Algorithm 2, we present a straightforward example of implementing RL training algorithms within `train_episode()`. Starting from Env initialization, in each step, agent A provides Env with actions based on the current state, receives the next state, reward, and other information, and updates the policy accordingly. Within the `env.step()` interface, actions are translated into configurations applied to the aircraft. Rewards and subsequent states are calculated, with logging information summarized concurrently.

For the Tester and Evaluator shown in Algorithm 3, the environment is first initialized to evaluate each algorithm in the set (including several baseline agents and the user's trained agent). The `rollout_episode()` interface is similar to `train_episode()`, but it does not include the policy update procedures. Finally, NeuralPlane evaluates the algorithm's performance and provides visualization of fixed-wing aircraft flight trajectories based on the flight data generated from testing.

B Details of Experiments

B.1 Experimental Parameters

Before researchers can use NeuralPlane to complete the full workflow of algorithm training, testing, evaluation, and replay, two preliminary steps must be completed: 1) Determine the fixed-wing aircraft dynamics model to be used, the task objectives that the algorithm will control the aircraft to achieve, and the operational conditions. This step is used to initialize the basic parameters of NeuralPlane's core simulation environment. 2) Determine the maximum number of training steps (M), the number of parallel rollouts during training (n), and the number of steps per rollout in one iteration (m). After setting these parameters, the platform can automatically execute the complete process. The experimental parameter settings for different task scenarios are shown in Table 7.

Table 7: The experimental parameter settings for different task scenarios

Name	n	m	M	env	scenario	model
Heading	3000	3000	1.35×10^9	Control	heading	F16
Control	3000	3000	2.25×10^9	Control	control	F16
Tracking	10000	100	3×10^8	Planning	tracking	F16

B.2 Additional Experimental Results

We conducte multiple experiments across all task scenarios, thoroughly demonstrating NeuralPlane's superiority in supporting RL algorithm training and showcasing the powerful capabilities of RL algorithms in fixed-wing aircraft control. Some experimental results are shown in Figure 12, 13, 14, with all results available at `https://anonymous.4open.science/r/NeuralPlane`. The results also indicate that in some high-difficulty scenarios, the control effectiveness of RL algorithms needs improvement, highlighting the platform's value and potential for RL research.

We also test the RL algorithms across all task scenarios and perform a visual evaluation of their performance. Some visualization results are shown in Figure 15, with all experimental results available at `https://anonymous.4open.science/r/NeuralPlane`.

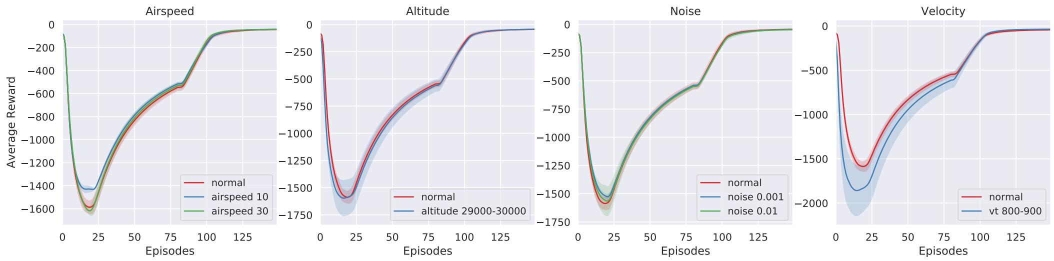


Figure 12: Training curves of PPO in different task scenarios. **From left to right**, the task conditions are different wind speeds, different flight altitudes, different environmental noise levels, and different flight speeds, with the task objective being the Heading task in all cases.

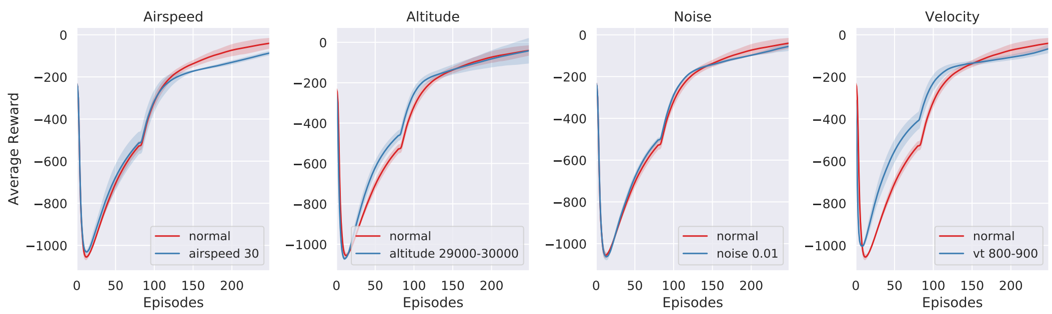


Figure 13: Training curves of PPO in different task scenarios. **From left to right**, the task conditions are different wind speeds, different flight altitudes, different environmental noise levels, and different flight speeds, with the task objective being the Control task in all cases.

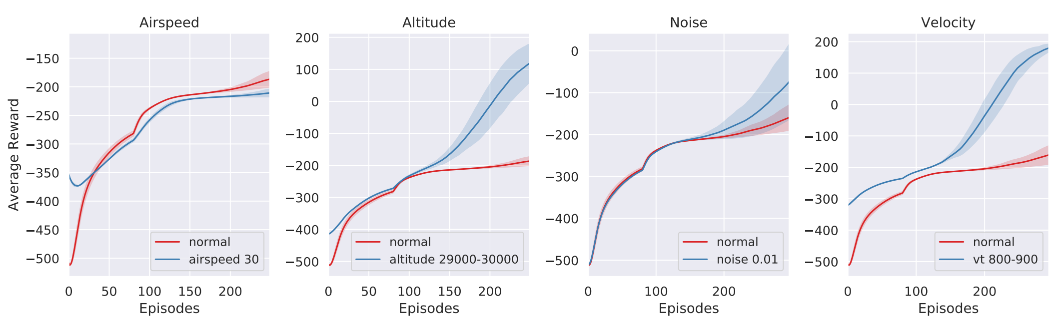
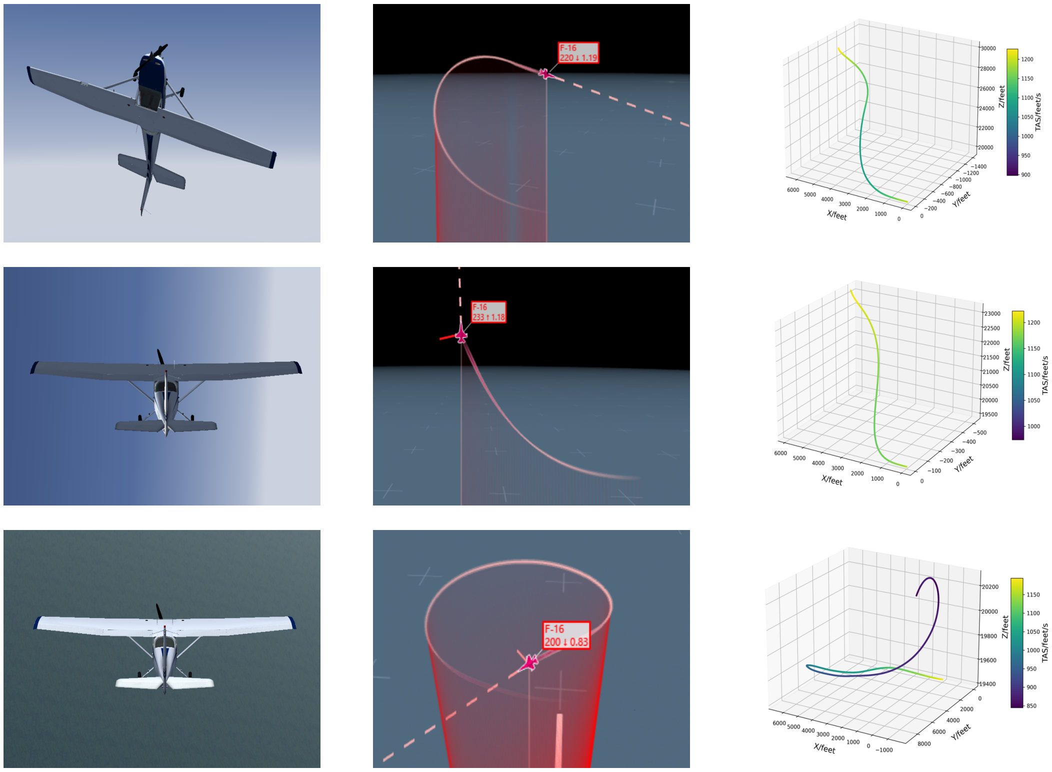


Figure 14: Training curves of PPO in different task scenarios. **From left to right**, the task conditions are different wind speeds, different flight altitudes, different environmental noise levels, and different flight speeds, with the task objective being the Tracking task in all cases.



(a) FlightGear (b) Tacview (c) Track chart

Figure 15: Visualization of fixed-wing aircraft flight trajectories. **Top**: The results of the Heading task. **Middle**: The results of the Control task. **Bottom**: The results of the Tracking task.

C Used Assets

NeuralPlane is an open-source tool available at `https://anonymous.4open.science/r/NeuralPlane`. It is licensed under the LGPL-3.0 license. Table 8 lists the resources and assets used in NeuralPlane along with their respective licenses. We strictly adhere to these licenses during the development of NeuralPlane.

Table 8: Used assets and their licenses

Type	Asset	Codebase	License
Baseline	PPO [27]	CloseAirCombat [54]	LGPL-3.0 license
Platform to Compare	Ardupilot [22] JSBSim [21] QPlane [23]	Ardupilot [22] JSBSim [21] QPlane [23]	LGPL-3.0 license LGPL-2.1 license -