Journal of Automation and Intelligence 3 (2024) 219–229

Contents lists available at ScienceDirect

Journal of Automation and Intelligence

journal homepage: www.keaipublishing.com/en/journals/journal-of-automation-and-intelligence/

Research article

A data-driven control method for ground locomotion on sloped terrain of a hybrid aerial-ground robot

Xinhang Xu[1], Yizhuo Yang[1], Muqing Cao *, Thien-Minh Nguyen, Kun Cao, Lihua Xie

School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, 639798, Singapore

ARTICLE INFO

Keywords:
Data-driven control
Hybrid aerial-ground robot
Adaptive control
Machine learning
Robotics
Nonlinear control systems

ABSTRACT

In this work, we present a data-driven solution for the attitude control of DoubleBee on slopes. DoubleBee is a novel hybrid aerial-ground robot with two rotors and two active wheels. Inspired by the physics modeling of the system, we add a channel-separated attention head to a deep ReLU neural network to predict disturbances from ground effects, motor torques and rotation axis shift. The proposed neural network is Lipschitz continuous, has fewer parameters and performs better for disturbance estimation than the baseline deep ReLU neural network. Then, we design a sliding mode controller using these predictions and establish its input-to-state stability and error bounds. Experiments show improvements of the proposed neural network in training speed and robustness over a baseline ReLU network, and a 40% reduction in tracking error compared to a baseline PID controller.

1. Introduction

Hybrid aerial-ground robots have gained significant attention in recent years in fields such as routine inspection [1], autonomous exploration [2], and construction operations [3] due to their ability to overcome obstacles that are difficult for traditional ground robots and their longer endurance compared to aerial robots. In our previous work [4], we proposed a new hybrid aerial-ground configuration called DoubleBee, leveraging the similarities and efficiencies of a bicopter and a balancebot. For its ground locomotion, we designed a decoupled mode, where the robot attitude is controlled by the propeller thrust, independent of the robot position controlled by the wheel rotation. This improves flexibility during ground locomotion, allowing it to pass under low obstacles like Fig. 1. DoubleBee achieved the highest energy efficiency on the ground among the existing hybrid aerial-ground robots [4,5]. However, the ground effect induced by the near-ground rotating propellers, the torque output to the wheels, and the influence of gravity pose challenges for precise attitude control.

In our previous work [6], to reduce the impacts of the ground effect, we performed system identification for DoubleBee on flat surfaces, established an explicit model of the ground effect, and designed a controller for decoupled mode attitude control. We ignored the influence of wheel motor torque on attitude control, as translational motion on a flat surface requires a small torque output. However, this results in unsatisfactory performance on the slope. Therefore, in this paper, to enhance DoubleBee's general applicability, we study the attitude control of DoubleBee when ascending or descending slopes.

Several factors complicate the control problem of DoubleBee on a slope. First, a larger torque is needed to counteract the effect of gravity on a slope, which acts as a disturbance torque for attitude control. Additionally, the ground effect changes during dynamic motion on a slope and becomes more difficult to model explicitly. Furthermore, the pitch rotation axis of DoubleBee shifts when the wheels are locked, adding further complexity. These factors can be considered as disturbances to the system.

Controller design for systems under disturbances and uncertainties has been extensively studied. Some early works use H_∞ control [7], extended state observer [8], robust Kalman filter [9], generalized proportional integral observer [10] and many other effective methods to handle the disturbance or uncertainty and improve system robustness. While the aforementioned methods have achieved success, they also have some issues. For example, Kalman-filter-based methods require manually tuning several difficult-to-identify noise covariance matrices. The tuning process can be obscure and difficult to understand, requiring extensive experimental work and knowledge of specialized hardware and software systems.

Recent works have focused on data-driven methods [11–13], using the representation power of neural networks to estimate highly nonlinear disturbances, achieving satisfactory accuracy. However, these

Peer review under responsibility of Chongqing University.

* Corresponding author.
E-mail addresses: xu0021ng@e.ntu.edu.sg (X. Xu), yang0670@e.ntu.edu.sg (Y. Yang), mqcao@ntu.edu.sg (M. Cao), thienminh.nguyen@ntu.edu.sg (T.-M. Nguyen), kun001@e.ntu.edu.sg (K. Cao), elhxie@ntu.edu.sg (L. Xie).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.jai.2024.08.001
Received 15 June 2024; Received in revised form 3 August 2024; Accepted 18 August 2024
Available online 22 August 2024

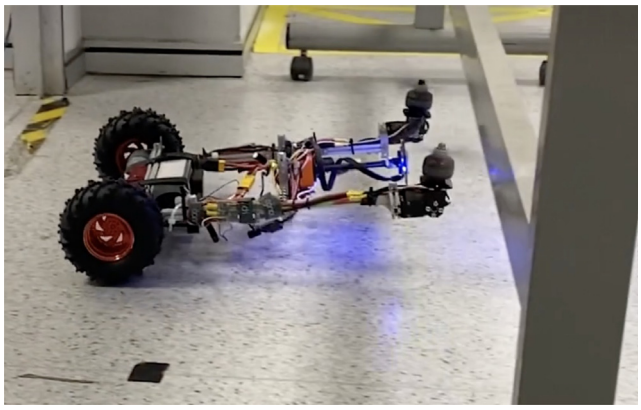

Fig. 1. DoubleBee passes under a low obstacle.

works primarily face the challenge of balancing neural network inference speed and prediction accuracy, and the difficulty in training convergence.

Furthermore, during the motion of DoubleBee along the slope, the disturbances and uncertainties, as well as the rotation axis shift and ground effect (detailed in Section 2.1.2), are closely coupled with the control signal, the thrusts of propellers. Conventional estimator-controller structures are not well-suited for this task due to the interdependence between the control signal and the disturbances. Among existing works, Neural Lander [12] is the most closely related research with real-world experiments and open-source implementation. Neural Lander primarily focuses on near-ground trajectory tracking for unmanned aerial vehicles (UAVs), utilizing a deep ReLU neural network to estimate the ground effect and integrates it into UAV dynamics for accurate trajectory tracking. The Neural Lander employs a cascade control system, wherein a high-level position controller generates a desired signal to a high-frequency, low-level thrust controller. In our case, we need to design the high-frequency low-level attitude controller directly to reduce the effect of disturbance on the rotational dynamics, so Neural Lander is not applicable. Additionally, in maintaining a deep enough neural network to ensure the robustness of the estimator, Neural Lander's position control frequency is limited to 10 Hz, which is relatively low for the attitude control of DoubleBee.

In this work, we design a controller tailored to DoubleBee. Considering that disturbances and uncertainties are essentially governed by physics, we get the inspiration from depthwise separable convolutions [14,15] and attention [16], utilizing dynamics analysis to design a physics-inspired neural network to learn these disturbances more effectively and quickly. Additionally, we design a sliding mode controller for the proposed network with proven input-state stability to control the attitude of DoubleBee on slopes.

The main contributions of the paper are summarized as follows:

- We analyze the dynamics of DoubleBee on slopes and provide a detailed description of the disturbance components.
- We design a neural network that combines the concepts of depthwise separable convolutions [14,15] and attention [16] with prior knowledge of dynamics model. This network is Lipschitz continuous, has fewer parameters and performs better for disturbance estimation than the deep ReLU neural network proposed in [12].
- We design a sliding mode controller with a neural disturbance estimator, prove its input-state stability, and establish an error bound.
- Experimental results verify the performance of the proposed physics-inspired neural network and controller performance, reducing the tracking error from 0.125 rad to 0.076 rad on a 30 deg wooden slope.

The rest of the paper is organized as follows: Section 2 provides a detailed dynamics analysis of DoubleBee on slopes and states the problem. Section 3 introduces our proposed physics-inspired network, analyzes its Lipschitz continuity, and states its advantages over ReLU deep neural networks. Section 4 discusses using neural networks for disturbance estimation in a nonlinear sliding mode controller with linear feedback and provides a theoretical analysis of the controller's stability and error bound. In Section 5, we describe the data collection method and preprocessing method used for training the neural network and experimentally validate the network's inference speed, training speed, robustness, and the role of the attention head. Finally, we compare the proposed controller with a PID controller in a real-world experiment, demonstrating that the proposed controller can better mitigate the effects of disturbances. Section 6 concludes the paper.

2. Dynamics and problem statement

2.1. Dynamics

DoubleBee's locomotion on the slope can be described using three right-hand coordinate systems: the global coordinate system $\mathtt{G} = \{O_\mathtt{G}, \vec{x}_\mathtt{G}, \vec{y}_\mathtt{G}, \vec{z}_\mathtt{G}\}$, the slope coordinate system $\mathtt{S} = \{O_\mathtt{S}, \vec{x}_\mathtt{S}, \vec{y}_\mathtt{S}, \vec{z}_\mathtt{S}\}$ and the body coordinate system $\mathtt{B} = \{O_\mathtt{B}, \vec{x}_\mathtt{B}, \vec{y}_\mathtt{B}, \vec{z}_\mathtt{B}\}$, where O, x, y, z represent the origin and the basis vectors and the subscripts $(\cdot)_\mathtt{G}, (\cdot)_\mathtt{B}, (\cdot)_\mathtt{S}$ indicate the coordinate frame.

We model the slope as a planar surface as shown in Fig. 2. Select a point $O_\mathtt{G}$ at the bottom of the slope as the origin of $\mathtt{G}$, and the projection of the gradient of the slope onto the global horizontal plane as the $\vec{x}_\mathtt{G}$, the global coordinate system $\mathtt{G}$ can be established with the vertical direction opposite to the gravity taken as $\vec{z}_\mathtt{G}$.

The $\mathtt{S}$ shares the origin and the Y-axis with $\mathtt{G}$, while $\vec{x}_\mathtt{S}$ is aligned with the gradient of the slope. For body coordinate system $\mathtt{B}$, we select the midpoint between the rotation center of the two wheels as its origin $O_\mathtt{B}$, as shown in Fig. 2. To establish $\mathtt{B}$, we choose the direction along the longitudinal body of DoubleBee as $\vec{z}_\mathtt{B}$, and choose the direction from the origin to the center of the left wheel as the $\vec{y}_\mathtt{B}$.

Denote the rotation matrices of the slope and the body frames in the global frame as ${}^\mathtt{G}_\mathtt{S}R, {}^\mathtt{G}_\mathtt{B}R \in SO(3)$, which can be decomposed into ${}^\mathtt{G}_\mathtt{S}R = R_y(\theta_S)$ and ${}^\mathtt{G}_\mathtt{B}R = R_z(\psi)R_y(\theta)R_x(\phi)$, where $\theta_S \in \mathbb{R}$ is the slope angle and the scalars ψ, θ, ϕ represent the *yaw*, *pitch*, *roll* angles of $\mathtt{B}$, respectively. $R_x(\cdot), R_y(\cdot)$ and $R_z(\cdot)$ represent the rotation matrix along the $\vec{x}, \vec{y}, \vec{z}$ axes, which are detailed as:

$$
\begin{aligned}
R_z(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
R_y(\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \\
R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}.
\end{aligned} \tag{1}
$$

When DoubleBee moves on the slope, the roll angle ϕ depends on ψ and the slope angle. And if the projection of $\vec{x}_\mathtt{B}$ onto the slope (i.e., the robot moving direction) is not aligned with the gradient of the slope, $\phi \neq 0$, resulting in the coupling between robots' angular velocity and acceleration ω_θ, α_θ, and the robot's attitude ϕ and ψ. In this work, we focus on designing a controller for robot pitch angle θ, and we assume that the robot's yaw ψ is maintained in $\{0, \pm\pi\}$ (such that the robot always moves along the gradient of the slope) by controlling the wheels. Hence, we give the following assumption:

Assumption 1. The DoubleBee is always commanded to move along the gradient by a high-level planner, which means $\phi = 0$ is always satisfied. Additionally, no lateral slipping occurs.

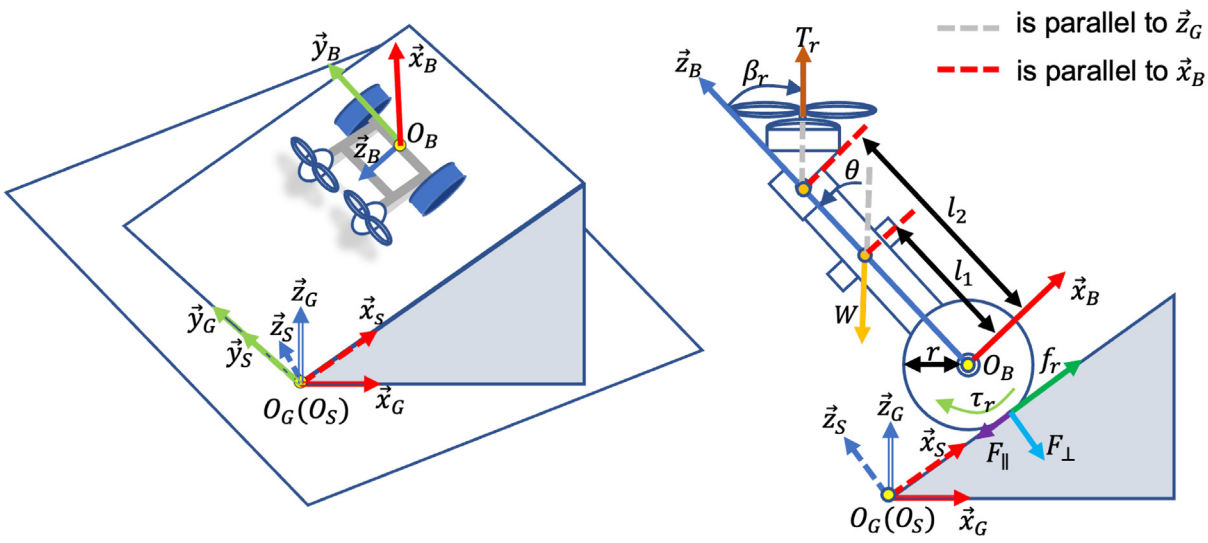

Fig. 2. Diagram of DoubleBee moving on a slope and its side view.

2.1.1. Dynamics of translational states

Denote DoubleBee's global position and global velocity as $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^3$, we consider the dynamics as:

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = R_z(\psi)R_y(\theta_S)\begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} {}_w\alpha_l \\ {}_w\alpha_r \end{bmatrix}, \tag{2}$$

where r is the wheel radius and ${}_w\alpha_l, {}_w\alpha_r$ represent the angular acceleration of the left and right wheels satisfying the following equation:

$$\begin{bmatrix} {}_w\alpha_r \\ {}_w\alpha_l \end{bmatrix} = \frac{1}{J_w}\begin{bmatrix} \tau_r - f_r(F_\perp, \zeta, \omega_r)r + F_\parallel r \\ \tau_l - f_l(F_\perp, \zeta, \omega_l)r + F_\parallel r \end{bmatrix}, \tag{3}$$

where the $\tau_r, \tau_l \in \mathbb{R}$ represent the output torques of the right and left motors driving the wheels, $J_w \in \mathbb{R}$ is the moment of inertia of the left or right wheel about its axis of rotation. The function $f \in \mathbb{R}$ [17] with subscript r, l represents the relationship between the friction and the contact force $F_\perp \in \mathbb{R}$, the wheel angular velocity ω_r, ω_l and the friction-related parameter $\zeta \in \mathbb{R}$.

$F_\parallel$ and $F_\perp$ are the parallel and perpendicular components of the sum of gravity and lift forces. The force due to gravity $\boldsymbol{W} \in \mathbb{R}^3$ is computed as $\boldsymbol{W} = [0,\ 0,\ -mg]^T$ with $m, g \in \mathbb{R}$ being the mass of DoubleBee and the magnitude of gravitational acceleration. The lift due to propellers $\boldsymbol{F_T} \in \mathbb{R}^3$ is computed as:

$$\boldsymbol{F_T} = (T_r R_y(\theta + \beta_r) + T_l R_y(\theta + \beta_l))\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{4}$$

where $T_r, T_l \in \mathbb{R}$ represent the thrust on the right and left sides provided by the rotating propellers, and β_r, β_l represent the tilt angle of the right and left servo.

2.1.2. Dynamics of angular states

Based on Assumption 1, we can define the *angular velocity vector* in the body frame as $\boldsymbol{\omega} = [0,\ \omega_\theta,\ 0]^T$. Similarly, the angular acceleration vector in the body frame is defined as $\boldsymbol{\alpha} = [0,\ \alpha_\theta,\ 0]^T$, where

$$\omega_\theta = \dot{\theta}, \quad \alpha_\theta = \ddot{\theta} = \dot{\omega}_\theta. \tag{5}$$

By Newton's second law for rotation, we can derive the following equation:

$$\begin{aligned} \alpha_\theta = \frac{1}{J_B}\{ &\sum_{i\in\{r,l\}} [(T_i + f_{gi})\sin(\beta_i)l_2 + \tau_i] \\ &+ mg\sin(\theta)l_1 + \tau_J \}, \end{aligned} \tag{6}$$

where the $J_B \in \mathbb{R}$ is the moment of inertia of DoubleBee body about its axis of rotation. $l_1, l_2 \in \mathbb{R}$ denote the distance from the center of gravity to the wheelbase and the perpendicular distance from the motor to the wheelbase. $f_{gi} = f_g(T_i, \theta, \beta_i, \zeta) \in \mathbb{R}$ is a function that computes the extra thrust caused by the ground effect. $\tau_J = \tau_J(T_r, T_l, \theta, \beta_r, \beta_l, \tau_r, \tau_l, \zeta) \in \mathbb{R}$ computes the torque disturbance caused by the rotation axis shift from the ground contact point to the wheel's center, which occurs when the wheels are locked and θ changes. To ensure that the rotation of the propellers on the slope does not affect ϕ and ψ, we enforce a symmetrical output in the control design, i.e., the thrusts of propellers and servo angles of both servos are the same, $T = T_r = T_l$ and $\beta = \beta_r = \beta_l$. Therefore, Eq. (6) can be simplified as:

$$\begin{aligned} \alpha_\theta &= \frac{2Tl_2\sin(\beta)}{J_B} + \frac{mg\sin(\theta)l_1}{J_B} + \alpha_d \\ &= \alpha_T + \alpha_{\boldsymbol{W}} + \alpha_d, \end{aligned} \tag{7}$$

where $\alpha_T = \frac{2Tl_2\sin(\beta)}{J_B}$ and $\alpha_{\boldsymbol{W}} = \frac{mg\sin(\theta)l_1}{J_B}$ represent the angular acceleration caused by the thrust T and gravity $\boldsymbol{W}$, respectively; α_d is the angular acceleration caused by other factors, including the output torque τ_r, τ_l, the disturbance torque τ_J, and the ground effect f_g as follows:

$$\alpha_d = \frac{2f_g\sin(\beta)l_2 + \tau + \tau_J}{J_B}, \tag{8}$$

where $\tau = \tau_r + \tau_l$ denotes the sum of output torque.

2.2. Problem statement

Different from conventional wheeled robots, where attitude control is trivial, DoubleBee exhibits a more sophisticated dynamics. Specifically the pitch angle θ can be manipulated for more complex motion profile, which demands a formal investigation to ensure robust and accurate control performance. To this end, the objective of our research is twofold.

First, we seek to predict the unknown angular acceleration α_d via a physics-inspired neural network with Lipschitz continuity. Second, we integrate the learning model in a closed-loop attitude control system for the pitch θ angle of Doublebee when it moves on the slope. As shown in Fig. 3, a cascade square root (Sqrt) controller is used to provide the reference angular acceleration to a thrust calculator (T Calculator), where the fix-point iteration is applied to compute the desired thrust based on the neural network prediction. The details will be presented in Section 4. We show that the controller is input-to-state stable via both theoretical analysis and real world experiments.

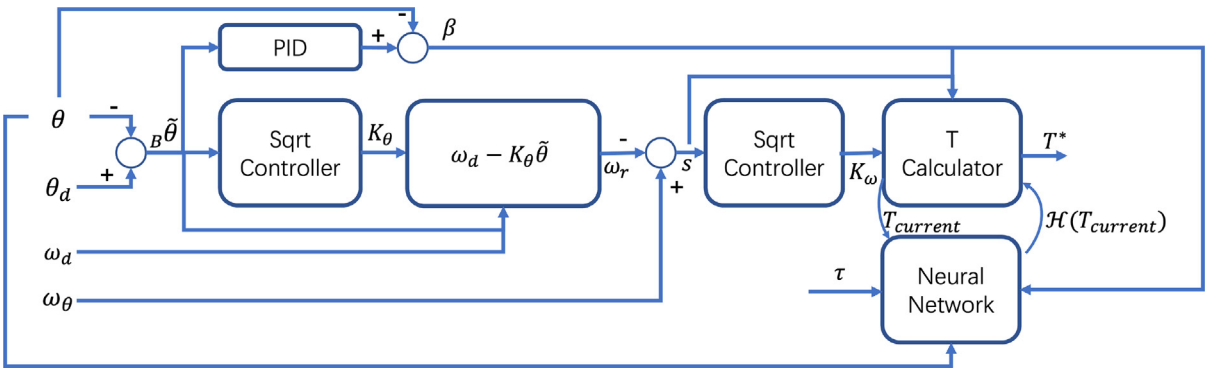

Fig. 3. Controller structure.

3. Physics-inspired neural network design

The neural network is composed of the physics-inspired attention head and a multi-layer perceptron (MLP) with spectral normalization. In this section, we will first introduce the structure of each part of the proposed neural network and analyze its Lipschitz continuity. Then, we will expand the Lipschitz continuity to the proposed neural network.

3.1. Fully connected layer and activation function

In this work, we mainly use a fully connected layer, which uses the Sigmoid function or ReLU function [18] as the activation function. Given an input $\boldsymbol{X} \in \mathbb{R}^N$ and the weight matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$, the output of the full connected layer $\boldsymbol{Y} \in \mathbb{R}^M$ is given as $\boldsymbol{Y} = \boldsymbol{\Phi X}$. Since the output $\boldsymbol{Y}$ is always considered as the input of the activation function, the output of Sigmoid function $\Psi_s(\boldsymbol{Y})$ is given as

$$\Psi_s(\boldsymbol{Y}) = \Psi_s(\boldsymbol{\Phi X}) = \frac{1}{1+e^{-\boldsymbol{\Phi X}}}, \tag{9}$$

and output of ReLU function $\Psi_R(\boldsymbol{Y})$ is given as

$$\Psi_R(\boldsymbol{Y}) = \Psi_R(\boldsymbol{\Phi X}) = \max(0, \boldsymbol{\Phi X}). \tag{10}$$

3.2. Lipschitz continuity&spectral normalization

3.2.1. Lipschitz continuity

Definition 1 (*Lipschitz Continuity and Lipschitz Constant*)**.** A function $h(a)$ is Lipschitz continuous on the domain A if $\exists L_a \in \mathbb{R}^+$, so-called the Lipschitz constant, such that $\|h(a_1) - h(a_2)\|_2 \le L_a \|a_1 - a_2\|_2, \forall a_1, a_2 \in A$.

Denote the operator $\|\cdot\|_L$ that finds the minimal Lipschitz constant, e.g., $\|h(a)\|_L = L_a$, we can get the following lemma.

Lemma 3.1. *If $h_1(a)$ and $h_2(a)$ are Lipschitz continuous on domain A, the inequality $\|h_1 \circ h_2\|_L \le \|h_1\|_L \|h_2\|_L$ always holds.*

Proof. $\forall a_1, a_2 \in A$, based on Definition 1, the following inequality holds:

$$\begin{aligned} &\left\|h_1(h_2(a_1)) - h_1(h_2(a_2))\right\|_2 \\ &\quad \le \|h_1\|_L \left\|h_2(a_1) - h_2(a_2)\right\|_2 \quad \square \\ &\quad \le \|h_1\|_L \|h_2\|_L \|a_1 - a_2\|_2 \end{aligned} \tag{11}$$

3.2.2. Spectral normalization

Spectral normalization is a method for modifying the maximum singular value of the weight matrix $\boldsymbol{\Phi}$ to make the neural network more stable [19,20]. Denote ${}^{SN}_{\lambda}\boldsymbol{\Phi} = S_N(\boldsymbol{\Phi}, \lambda)$ as the desired weight matrix with λ as the maximum singular value of the fully connected layer:

$${}^{SN}_{\lambda}\boldsymbol{\Phi} = S_N(\boldsymbol{\Phi}, \lambda) = \lambda \frac{\boldsymbol{\Phi}}{\sigma(\boldsymbol{\Phi})}, \tag{12}$$

where $\sigma(\boldsymbol{\Phi})$ represents the maximum signature value of weight matrix $\boldsymbol{\Phi}$.

3.3. The physics inspired neural network

Building on the success of depthwise separable convolutions in enhancing neural network performance and efficiency [14] and accelerating deployment on edge computing devices [15], we combine the similar structure with the attention [16] to design our neural network from the perspective of physical modeling. The neural network structure is shown in Fig. 4.

The proposed physics-inspired neural network uses the sample $\{\tau, \theta, \beta, T\}$ as input to predict α_d, which consists of an attention head (layer 1) and an information processing network (layers 2 to I+1).

3.3.1. Attention head

For attention head, we combine human intuition and physics modeling: different physical quantities have varying degrees of influence on the model. Drawing on the idea of depthwise separable convolutions, we utilize four separate fully connected layers to encode information of the input physical quantities. Then, different weights $\lambda_\theta, \lambda_\beta, \lambda_\tau, \lambda_T \in \mathbb{R}^+$ are applied to the extracted feature of θ, β, τ, T to guide the network focus on more important features. The allocation law of λ will be discussed later in inequality (14). Finally, the information of each physical quantities is fused using fully connected layers to determine the output. Spectral normalization is applied to each weight ${}^{SN}_{1}\boldsymbol{\Phi}^2_\theta, {}^{SN}_{1}\boldsymbol{\Phi}^2_\beta, {}^{SN}_{1}\boldsymbol{\Phi}^2_\tau, {}^{SN}_{1}\boldsymbol{\Phi}^2_T \in \mathbb{R}^{D\times 4D}$ of the fully connected layer, ensuring that the maximum singular value is 1, resulting in the superimposed sample $X_h(\tau, \theta, \beta, T)$.

$$\begin{aligned} X_h = {}&{}^{SN}_{1}\boldsymbol{\Phi}^2_\theta \Psi_S\left({}^{SN}_{\lambda_\theta}\boldsymbol{\Phi}^1\theta\right) + {}^{SN}_{1}\boldsymbol{\Phi}^2_\beta \Psi_R\left({}^{SN}_{\lambda_\beta}\boldsymbol{\Phi}^1\beta\right) \\ &+ {}^{SN}_{1}\boldsymbol{\Phi}^2_\tau \Psi_S\left({}^{SN}_{\lambda_\tau}\boldsymbol{\Phi}^1\tau\right) + {}^{SN}_{1}\boldsymbol{\Phi}^2_T \Psi_R\left({}^{SN}_{\lambda_T}\boldsymbol{\Phi}^1 T\right) \end{aligned} \tag{13}$$

We use different activation functions for different channels of the attention head. Based on the fitting ability of ReLU function as Eq. (10), which essentially performs local linear approximation of high-dimensional data, and the continuous nonlinearity of the Sigmoid function as Eq. (9), which bounds the output but may lead to gradient vanishing [21], we use the ReLU function on β, T and Sigmoid function on θ, τ. The reasons are as follows.

Considering Eq. (8), the main components of our estimated target are f_g and τ_J. Referring to the ground effect given in previous work [22], the ground effect f_g is mainly related to the distance from the rotor plane to the ground. From Fig. 2, it is clear that this distance has a nonlinear relationship with the θ, approximately a cosine function. Therefore, it is intuitive to use the Sigmoid function on the θ channel to increase its nonlinearity and achieve better fitting in local regions compared to the linear approximation of the ReLU function. For τ_J, due to the highly nonlinear nature of the torque disturbance

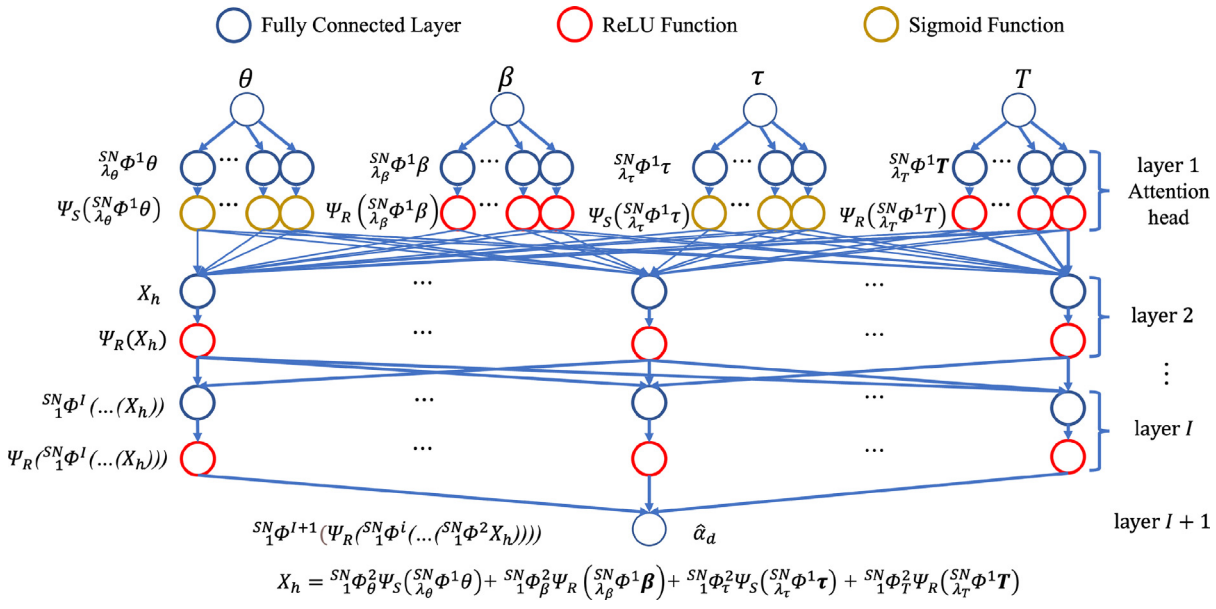

Fig. 4. The structure of physics-inspired neural network.

caused by the rotation axis shift, we also use the Sigmoid function for the τ channel.

For β, we can see from Eq. (23) in controller design in Section 4.2 that it has a linear relationship with θ. Since we already use the Sigmoid function to increase the nonlinearity of θ channel, we apply the concept of Taylor expansion, using ReLU function as the activation function to learn its lower-order expansion, while the higher-order expansion is learned through θ channel. Additionally, since the Sigmoid function involves exponential operations, which are slightly more computationally expensive than the ReLU function, we aim to use fewer Sigmoid functions to improve the inference speed of the network. Therefore, we use the ReLU on the β channel. Similar to the β channel, T is essentially proportional to f_g. Hence we use the ReLU function on the T channel.

Additionally, as shown in Eq. (7), T contributes significantly to counter the disturbance, and also, due to the need for the controller to solve T, we require the neural network to focus more on T. Considering the aforementioned design, we have:

$$\lambda_T \gg \lambda_\theta = \lambda_\beta = \lambda_\tau \tag{14}$$

3.3.2. Information processing network

For the information processing network (layer 2 to I+1 in Fig. 4), we use a multilayer ReLU neural network $\mathcal{G}(\cdot)$ which normalizes each weight matrix so that its singular value is equal to 1, specifically:

$$\mathcal{G}(X_h) = {}^{SN}_{1}\boldsymbol{\Phi}^{I+1}(\boldsymbol{\Phi}_R(\ldots {}^{SN}_{1}\boldsymbol{\Phi}^4(\Psi_R({}^{SN}_{1}\boldsymbol{\Phi}^3(\Psi_R(X_h))))))), \tag{15}$$

where the superscript of $\boldsymbol{\Phi}$ denotes the layer index. This design avoids gradient vanishing caused by the Sigmoid function introduced in the attention head.

3.4. Lipschitz continuity analysis

Lemma 3.2. *The proposed neural network $\Psi(\theta,\beta,\tau,T)$ is Lipschitz continuous and its Lipschitz constant satisfies $\|\Psi\|_L \le 0.25\lambda_\theta + 0.25\lambda_\tau + \lambda_\beta + \lambda_T$.*

Proof. From Eq. (9) and Eq. (10), we can get $\|\Psi_S(\cdot)\|_L = 0.25$, $\|\Psi_R(\cdot)\|_L = 1$. Applying Lemma 3.1 to each element of X_h, we can get following inequalities:

$$\begin{aligned}
&\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\theta\Psi_S\left({}^{SN}_{\lambda_\theta}\boldsymbol{\Phi}^1\theta\right)\right\|_L \le \lambda_\theta\|\Psi_S(\cdot)\|_L = 0.25\lambda_\theta\\
&\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\beta\Psi_R\left({}^{SN}_{\lambda_\beta}\boldsymbol{\Phi}^1\beta\right)\right\|_L \le \lambda_\beta\|\Psi_R(\cdot)\|_L = \lambda_\beta\\
&\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\tau\Psi_S\left({}^{SN}_{\lambda_\tau}\boldsymbol{\Phi}^1\tau\right)\right\|_L \le \lambda_\tau\|\Psi_S(\cdot)\|_L = 0.25\lambda_\tau\\
&\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_T\Psi_R\left({}^{SN}_{\lambda_T}\boldsymbol{\Phi}^1T\right)\right\|_L \le \lambda_T\|\Psi_R(\cdot)\|_L = \lambda_T
\end{aligned} \tag{16}$$

Denote $X_1 = [\theta_1,\beta_1,\tau_1,T_1]^{\mathrm{T}}, X_2 = [\theta_2,\beta_2,\tau_2,T_2]^{\mathrm{T}}$ in the feasible region. So the following inequality holds:

$$\begin{aligned}
&\|X_h(X_1) - X_h(X_2)\|_2\\
\le&\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\theta\Psi_S\left({}^{SN}_{\lambda_\theta}\boldsymbol{\Phi}^1\theta_1\right) - {}^{SN}_{1}\boldsymbol{\Phi}^2_\theta\Psi_S\left({}^{SN}_{\lambda_\theta}\boldsymbol{\Phi}^1\theta_2\right)\right\|_2\\
&+\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\beta\Psi_R\left({}^{SN}_{\lambda_\beta}\boldsymbol{\Phi}^1\beta_1\right) - {}^{SN}_{1}\boldsymbol{\Phi}^2_\beta\Psi_R\left({}^{SN}_{\lambda_\beta}\boldsymbol{\Phi}^1\beta_2\right)\right\|_2\\
&+\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_\tau\Psi_S\left({}^{SN}_{\lambda_\tau}\boldsymbol{\Phi}^1\tau_1\right) - {}^{SN}_{1}\boldsymbol{\Phi}^2_\tau\Psi_S\left({}^{SN}_{\lambda_\tau}\boldsymbol{\Phi}^1\tau_2\right)\right\|_2\\
&+\left\| {}^{SN}_{1}\boldsymbol{\Phi}^2_T\Psi_R\left({}^{SN}_{\lambda_T}\boldsymbol{\Phi}^1T_1\right) - {}^{SN}_{1}\boldsymbol{\Phi}^2_T\Psi_R\left({}^{SN}_{\lambda_T}\boldsymbol{\Phi}^1T_2\right)\right\|_2
\end{aligned} \tag{17}$$

Base on Definition 1 and inequality (17), $\|X_h(X_1) - X_h(X_2)\|_2 \le P$, where

$$\begin{aligned}
P =&0.25\lambda_\theta\|\theta_1-\theta_2\|_2 + \lambda_\beta\|\beta_1-\beta_2\|_2\\
&+ 0.25\lambda_\tau\|\tau_1-\tau_2\|_2 + \lambda_T\|T_1-T_2\|_2.
\end{aligned} \tag{18}$$

And the following inequalities always hold:

$$\begin{aligned}
&\|\theta_1-\theta_2\|_2 \le \|X_1-X_2\|_2; \|\tau_1-\tau_2\|_2 \le \|X_1-X_2\|_2\\
&\|\beta_1-\beta_2\|_2 \le \|X_1-X_2\|_2; \|T_1-T_2\|_2 \le \|X_1-X_2\|_2.
\end{aligned} \tag{19}$$

Since $\|X_h(X_1) - X_h(X_2)\|_2 \le P$, we can get the following inequality by applying inequality (19) to Eq. (18):

$$\begin{aligned}
&\|X_h(X_1) - X_h(X_2)\|_2 \le P\\
&\le (0.25\lambda_\theta + 0.25\lambda_\tau + \lambda_\beta + \lambda_T)\|X_1 - X_2\|_2.
\end{aligned} \tag{20}$$

Therefore, we can get the Lipschitz constant of the attention head as $\|X_h\|_L \le 0.25\lambda_\theta + 0.25\lambda_\tau + \lambda_\beta + \lambda_T$. Since $\|\Psi_R\|_L = 1$, we can get $\|\mathcal{G}\|_L = 1$ by applying Lemma 3.1 to Eq. (15). According to the

structure of the proposed neural network, the output of the network can be written as $\Psi(X) = \mathcal{G}(X_h(X))$. Applying Lemma 3.1, we can get $\|\Psi\|_L \le \|X_h\|_L \|\mathcal{G}\|_L \le 0.25\lambda_\theta + 0.25\lambda_\tau + \lambda_\beta + \lambda_T$. □

Lemma 3.3. *The inequality*

$$\|\Psi(X_1) - \Psi(X_2)\|_2 \le 0.25(\lambda_\theta\|\theta_1 - \theta_2\|_2 + \lambda_\tau\|\tau_1 - \tau_2\|_2) + \lambda_\beta\|\beta_1 - \beta_2\|_2 + \lambda_T\|T_1 - T_2\|_2$$

holds when $X_1 = [\theta_1, \beta_1, \tau_1, T_1]^{\mathrm{T}}$, $X_2 = [\theta_2, \beta_2, \tau_2, T_2]^{\mathrm{T}}$ *are in the feasible region.*

Proof. With $\|\mathcal{G}(\cdot)\|_L = 1$, based on the definition of Lipschitz constant, we can get:

$$\begin{aligned}\|\Psi(X_1) - \Psi(X_2)\|_2 &= \|\mathcal{G}(X_h(X_1)) - \mathcal{G}(X_h(X_2))\|_2 \\ &\le \|X_h(X_1) - X_h(X_2)\|_2\end{aligned} \tag{21}$$

Combined with Eq. (18), we complete the proof. □

3.5. Performance analysis

Compared to the ReLU deep learning network commonly used for similar tasks [12,13], the proposed network has the following advantages: fewer parameters, a faster training process, and more robust performance. In this subsection, we will analyze these advantages. The faster training process and more robust performance will be further verified through the experiment in Section 5.2.

3.5.1. Fewer parameters

This is mainly achieved by drawing on the idea of depthwise separable convolutions, replacing the fully connected layer with channel-wise processing of the input data. Suppose we have two structures as shown in Fig. 5, each with the same number of neurons, $4D$. The parameter count of our weight matrix is $4D$, while that of B is $16D$. Our parameter count is only a quarter of the ReLU deep neural network.

3.5.2. Faster training process and more robust performance

This advantage arises from our targeted processing and weighting of the signals in each channel as described in Section 3.3.1, meaning that the network structure has a certain physical basis. This allows the network to identify the relationship between inputs and outputs more effectively. The performance will be discussed later through experiments in Section 5.2.

4. Controller design and stability analysis

The controller is similar to neural-lander [12], a deep neural network based unmanned aerial vehicle landing controller, and the controller with nonlinear feedback linearization [6]. The proposed control system is shown in Fig. 3. A cascade square root controller (SQRT controller) is used to generate the desired angular acceleration command. The thrust calculator (T Calculator) computes the desired thrust for the propellers using an iterative fixed point algorithm, using the disturbance estimation from the proposed neural network. Leveraging Lipschitz continuity of the proposed neural network, a contraction mapping is constructed to guarantee the unique solution of the thrust calculator.

4.1. Square root controller (SQRT controller)

The square root controller is an adaptive gain controller [6]. It limits the second-order derivative of the error signal, setting a small gain for large errors to avoid overshooting and a large gain to reduce the steady-state error. Given an error signal $e \in \mathbb{R}$, the second order derivative limit $\ddot{e}_l \in \mathbb{R}^+$, and the largest gain K, the square root gain ${}^{sqrt}_{K}K^{\ddot{e}_l}_{e} \in \mathbb{R}$ is computed as following:

$${}^{sqrt}_{K}K^{\ddot{e}_l}_{e} = {}^{sqrt}_{K}K(e, \ddot{e}_l) = \begin{cases} \dfrac{\sqrt{2K^2\|e\|_2 - \ddot{e}_l}}{K\|e\|_2}, & \|e\|_2 > \dfrac{\ddot{e}_l}{K^2}, \\ K, & \text{otherwise.} \end{cases} \tag{22}$$

4.2. Controller design

The controller is similar to the sliding mode controller [23]. Given the desired body pitch angle θ_d, the tracking error is computed as $\tilde{\theta} = \theta_d - \theta$. Since we always desire the lift $\mathbf{T}$ to be perpendicular to the ground to minimize its impact on motion control [6], the tilting angle β is computed using a PID controller:

$$\begin{aligned}\beta(k) = -\theta(k) &+ K_p\tilde{\theta}(k) + K_I\sum\tilde{\theta}(k) \\ &+ K_D(\tilde{\theta}(k) - \tilde{\theta}(k-1)),\end{aligned} \tag{23}$$

where K_p, K_I, K_D are the parameters of a PID controller. The output of the PID controller is limited to $[-0.1, 0.1]$ rads.

In our previous works [4,6], we mainly focused on the ground locomotion of DoubleBee on a horizontal plane. The torque required

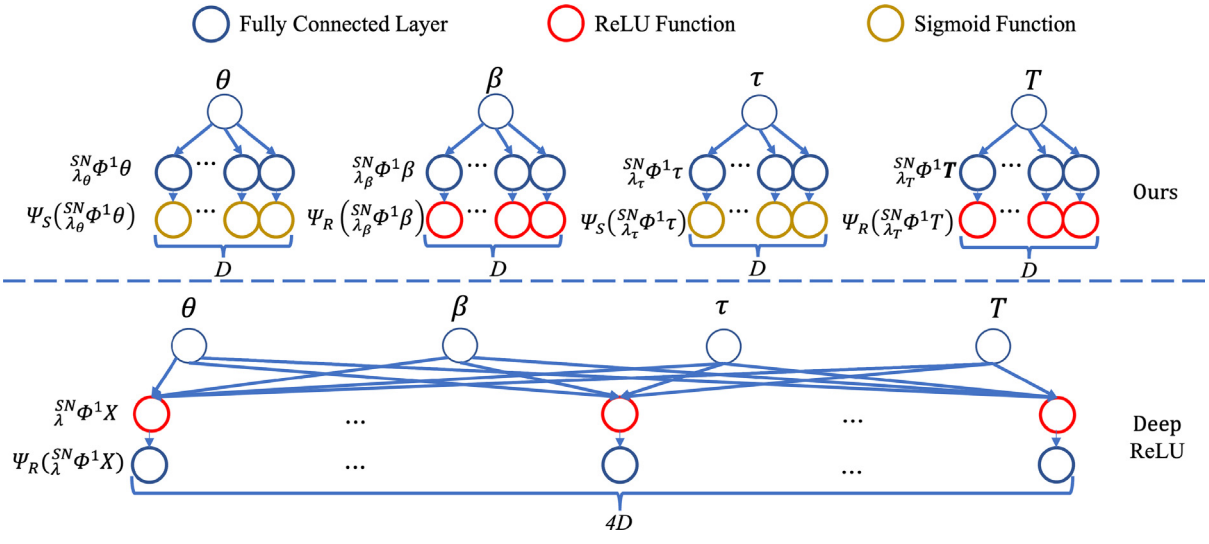

Fig. 5. The comparison between the proposed attention head and fully connected layer in Deep ReLU neural network.

for moving on a level ground is significantly lower than that for moving on a slope, as extra torque is needed to drive the robot against its weight. As a result, the previously ignored influence of τ on the angular acceleration of the robot (Eq. (8)) should be considered when the robot moves on a slope. Based on Eq. (7) and (23), it is clear that at larger θ, a larger T is required to realize the same angular acceleration. To ensure sufficient thrust capability to counteract the influence of α_d, we make the following assumption.

Assumption 2. When moving on a slope, the desired body pitch angle always satisfies $\theta_d \in [-1.57, -0.52] \cup [0.52, 1.57]$ radians.

To design the controller, we first design a composite variable $s = 0$ as a manifold on which the pitch tracking error $\tilde{\theta} \to 0$. Based on Eq. (5), s is denoted as:

$$s = \dot{\tilde{\theta}} + K_\theta \tilde{\theta} = \omega_\theta - \omega_r, \tag{24}$$

where $\omega_r = \omega_d - K_\theta \tilde{\theta}$ is a reference angular velocity obtained from the desired pitch rate $\omega_d = \dot{\theta}_d$, and $K_\theta = {}^{sqrt}_{K_1} K^{\alpha_l}_{\tilde{\theta}}$ is the gain obtained from the square-root controller. Using the neural network we introduced in Section 3.3, we can obtain the estimated disturbance $\hat{\alpha}_d = \Psi(\tau, \theta, \beta, T) = \Psi(\xi, T)$, where $\xi = \{\tau, \theta, \beta\}$ for notation simplification. We design the desired angular acceleration contributed by the thrust α_T as:

$$\alpha_T = \bar{\alpha}_T - \hat{\alpha}_d \quad \text{with} \quad \bar{\alpha}_T = \dot{\omega}_r - \alpha_{\mathbf{W}} - K_\omega s, \tag{25}$$

where $K_\omega = {}^{sqrt}_{K_2} K^{\alpha_\omega}_{s}$, $\bar{\alpha}_T$ represents the angular acceleration needed to track θ_d when $\hat{\alpha}_d = 0$. Combining Eq. (25) with Eq. (7), we can get the closed-loop system dynamics:

$$\dot{s} + K_\omega s = \alpha_d - \hat{\alpha}_d = \epsilon. \tag{26}$$

We will derive that $\tilde{\theta}$ is bounded as long as the estimation error $\epsilon = \alpha_d - \hat{\alpha}_d$ is bounded. The stability analysis will be detailed in Section 4.4 later.

4.3. Thrust calculator

This section details the approach to obtain the motor thrust input T. Using Eq. (7) and (25), we can get:

$$\begin{aligned} T = \mathcal{H}(T) &= \frac{J_B}{2l_2 \sin(\beta)}(\bar{\alpha}_T - \hat{\alpha}_d) \\ &= \frac{J_B}{2l_2 \sin(\beta)}(\bar{\alpha}_T - \Psi(\xi, T)). \end{aligned} \tag{27}$$

Eq. (27) cannot be solved to obtain an analytical solution. Below, we describe an iterative approach to find T^* such that $T^* = \mathcal{H}(T^*)$.

Lemma 4.1. *With $\xi, \bar{\alpha}_T$ fixed, $\mathcal{H}(T)$ is a contraction mapping if $\|\frac{J_B}{2l_2 \sin(\beta)}\|_2 \|\Psi(\xi, T)\|_L < 1$.*

Proof. Denote $T_1, T_2 \in \mathbb{T}$, where $\mathbb{T}$ is a set of feasible thrusts. If $\xi, \bar{\alpha}_T$ are fixed, the following inequality holds:

$$\begin{aligned} \|\mathcal{H}(T_1) - \mathcal{H}(T_2)\|_2 &\le \left\| \frac{J_B}{2l_2 \sin(\beta)} \right\|_2 \|\Psi(\xi, T_1) - \Psi(\xi, T_2)\|_2 \\ &\le \left\| \frac{J_B}{2l_2 \sin(\beta)} \right\|_2 \|\Psi(\xi, T)\|_L \|T_1 - T_2\|_2 \end{aligned} \tag{28}$$

With $\|\frac{J_B}{2l_2 \sin(\beta)}\|_2 \|\Psi(\xi, T)\|_L < 1$, we can get $\|\mathcal{H}(T)\|_L < 1$, so $\mathcal{F}(T)$ is a contraction mapping. □

To ensure the condition in the above lemma holds, we set the hyperparameter $\lambda_\theta, \lambda_\tau, \lambda_\beta, \lambda_T$ of the neural network (Section 3.3) such that

$$\left\| \frac{J_B}{2l_2 \sin(\beta)} \right\|_2 (0.25\lambda_\theta + 0.25\lambda_\tau + \lambda_\beta + \lambda_T) < 1. \tag{29}$$

As a result, $\mathcal{H}(T)$ is a contraction mapping, and there always exists a unique solution T^* for $T^* = \mathcal{H}(T^*)$. Hence, we can use the fix-point iteration method to solve T^*, as described in the following pseudocode.

After obtaining T^*, we can use the predetermined thrust-to-PWM curve (Fig. 6) to get the electrical signal for controlling the motors.

Algorithm 1 Find Fixed Point of Contraction Mapping

```
Input: H(·), T_0, tolerance, maxIterations
Output: Fixed point T*
T_current ← T_0
iteration ← 0
while iteration < maxIterations do
    T_next ← H(T_current)
    if |T_next − T_current| < tolerance then
        return T_next                          ▷ Fixed point found
    end if
    T_current ← T_next
    iteration ← iteration + 1
end while
return T_current    ▷ Return current point if max iterations reached
```

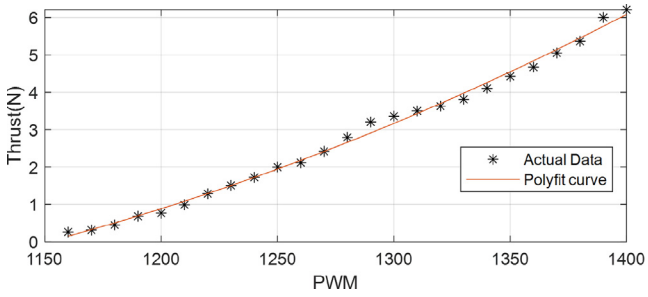

Fig. 6. The relationship between the thrust and PWM [6].

4.4. Stability analysis

In this subsection, we discuss the stability of the proposed controller. To prove the stability, we first give some assumptions with practical reasons.

Assumption 3. $\theta_d, \omega_d, \dot{\omega}_d$ are bounded.

$\theta_d, \omega_d, \dot{\omega}_d$ are desired states set by the pilot or a high-level planner whose bounds can be imposed and determined beforehand. In the proposed controller, we use the sliding model method to convert the angle tracking problem to an angular velocity tracking problem (Eq. (24)). If Assumption 3 is not satisfied, there will be an unlimited ω_r, in practice, this ω_r will be larger than the angular velocity limitation which is set for safety. So the controller will rewrite the ω_r to the limitation that satisfies Assumption 3. So for ease of analysis, we will assume that the desired state is bounded.

Assumption 4. $\tau_r, \tau_l, \dot{\tau}_r, \dot{\tau}_l$ are bounded.

τ_r, τ_l are generated by a position controller to control the wheel speed. Most wheel speed controllers produce continuous outputs, and the motor generates torque continuously. Therefore, it is unlikely that the motor torques and their change rates become unbounded.

Assumption 5. $\|\alpha_d(k) - \hat{\alpha}_d(k)\|_2 \le \epsilon_{max}$ where ϵ_{max} is a positive constant.

The proposed network shows good robustness on unseen data, as will be shown in Section 5.2. In practice, if the input during operation remain within the normal bounds of the training data, the estimation error is unlikely go out of the norm. Hence, we can assume that Assumption 5 holds.

Lemma 4.2. *If Assumptions 3, 4 and 5 hold, $\tilde{\theta}$ will exponentially converge to error ball:*

$$\lim_{t \to \infty} \|\tilde{\theta}(t)\|_2 \le \frac{\epsilon_{max}}{K_\theta K_\omega} \tag{30}$$

Proof. Denote the Lyapunov function as $\mathcal{V} = 0.5s^2$, by applying Eq. (26), we can get its time-derivative $\dot{\mathcal{V}}$:

$$\begin{aligned}\dot{\mathcal{V}} &= s\dot{s}\\ &= s(-K_\omega s + \epsilon)\\ &\le \|s\|_2\| - K_\omega s + \epsilon\|_2\\ &\le -K_\omega s^2 + \epsilon_{max}\|s\|_2\\ &\le -2K_\omega\mathcal{V} + \epsilon_{max}\sqrt{2\mathcal{V}}\end{aligned} \tag{31}$$

Denote $\mathcal{W} = \sqrt{\mathcal{V}} = \sqrt{0.5}\|s\|_2$, so $\dot{\mathcal{W}} = \frac{\dot{\mathcal{V}}}{2\sqrt{\mathcal{V}}}$. We can rewrite Inequality (31) as:

$$\dot{\mathcal{W}} \le -K_\omega\mathcal{W} + \epsilon_{max}\sqrt{0.5} \tag{32}$$

Solving the Inequality (32), we can obtain:

$$\begin{aligned}\mathcal{W}(t) \le& \mathcal{W}(t_0)e^{-K_\omega(t-t_0)}\\ &+ \int_{t_0}^{t} e^{-K_\omega(t-\gamma)}\sqrt{0.5}\epsilon_{max}\mathrm{d}\gamma.\end{aligned} \tag{33}$$

By applying $u = t - \gamma$ to Inequality (33):

$$\begin{aligned}\mathcal{W}(t) &\le \mathcal{W}(t_0)e^{-K_\omega(t-t_0)} + \sqrt{0.5}\epsilon_{\max}\int_0^{t-t_0} e^{-K_\omega u}\mathrm{d}u\\ &\le \mathcal{W}(t_0)e^{-K_\omega(t-t_0)} + \frac{\sqrt{0.5}\epsilon_{\max}}{K_\omega}\left(e^{-K_\omega(t-t_0)} - 1\right)\\ &\le \mathcal{W}(t_0)e^{(-K_\omega)(t-t_0)} + \frac{\sqrt{0.5}\epsilon_{\max}}{K_\omega}.\end{aligned} \tag{34}$$

Applying $\mathcal{W} = \sqrt{0.5}\|s\|_2$ to Inequality (34), we can get:

$$\|s(t)\|_2 \le \|s(t_0)\|_2 e^{-K_\omega(t-t_0)} + \frac{\epsilon_{\max}}{K_\omega}. \tag{35}$$

From Inequality (35), it is easy to find that $\|s(t)\|_2$ converges exponentially, and the convergence rate K_ω and the static error is $\lim_{t\to\infty}\|s(t)\|_2 \le \frac{\epsilon_{\max}}{K_\omega}$. So, the proposed controller is input-to-state stable. Based on Eq. (24), we have:

$$\lim_{t\to\infty}\|\tilde{\theta}(t)\|_2 = \lim_{t\to\infty}\frac{\|s(t)\|_2}{K_\theta} \le \frac{\epsilon_{\max}}{K_\theta K_\omega}. \quad \square \tag{36}$$

5. Experiment

This section first introduces the hardware used and the data collection and preprocessing methods for network training. Then, experimental evaluations of the proposed neural network and controller performance are presented.

5.1. Data collection and processing

As shown in Fig. 7, we set up a slope for the experiment, connecting DoubleBee to a ground computer with an AMD 5950XT CPU via a USB 3.0 cable. DoubleBee's onboard controller updates at 400 Hz and logs the PWM outputs to the rotors for providing lift T_{pwm}, the servo angles β, the torque control commands to the wheels motor τ, the angular velocity $\omega_\phi, \omega_\theta, \omega_\psi$, and the attitude ϕ, θ, ψ. The onboard controller sends the most recent data string to the ground computer via MAVROS at 100 Hz for recording. We use manual remote control and predetermined control command sequences to generate high-level commands and employ the low-level tracking controller from our previous work [4] to collect data. We collect a total of 12 sets of robot movement data at different target angles, ranging from 30 to 80 degrees in 5-degree increments. These datasets are used for the offline training and testing of the learning model. Within each dataset, DoubleBee executes a single ascent and descent along the slope under the control of a PID controller.

Fig. 7. Diagram of DoubleBee moving on a slope in an experiment in its side view.

After data collection, we obtain thrust T by the PWM-to-thrust mapping shown in Fig. 6. Based on Eq. (7), we can use angular acceleration α_θ to directly calculate $\hat{\alpha}_d$ as the ground truth for training. However, to the best of our knowledge, no existing sensors can directly measure angular acceleration α_θ with good accuracy. Therefore, we estimate α_θ using available measurements including angular velocity ω_θ and angle θ. Because the measurement of angular velocity ω_θ obtained from the onboard low-cost gyroscope sensor is very noisy, estimating the angular acceleration α_θ by taking the difference of two consecutive angular velocity samples would amplify the noise and result in a very low signal-to-noise ratio, causing the neural network to only learn the noise distribution instead of the relationship between input and output. Therefore, based on the system update Eq. (5), we used a Kalman filter (KF) with angle and angular velocity measurements to estimate the angular acceleration.

The comparison between KF-based and difference-based angular acceleration estimation is shown in Fig. 8. We observe that the KF-based estimate contains much less noise but inevitably introduces a lag in the estimation. In practice, we offset this delay by aligning the first thrust peak observed when DoubleBee transitions from an idle state to an operating state with the corresponding peak in the angular acceleration estimate.

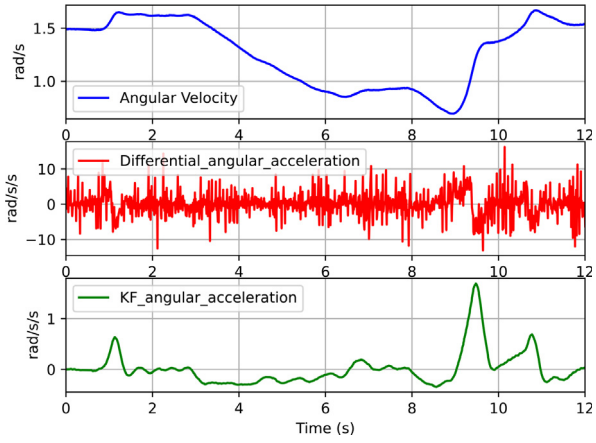

Fig. 8. The angular velocity and angular acceleration estimated by the differentiation method and Kalman filter.

5.2. Neural network performance

To validate the effectiveness of the proposed method, we conduct comparative studies on the training and inference performance, the robustness of the network, and the effectiveness of the attention head. Lipschitz continuous deep ReLU network [12,13], which is commonly used in similar disturbance estimation problems is selected as the baseline for comparison.

In the experiments, the attention head of the proposed network is set to 20 neurons, followed by an information processing network consisting of three layers, each with 64 neurons. The baseline network is set to a similar configuration, consisting of four layers in total. The

first layer comprises 80 neurons, while the subsequent layers are set to 64 neurons. We employ an Adam optimizer with 64 batch size for model training, with each model trained for 50 epochs.

5.2.1. Training and inference performance

In this experiment, we select the initial 70% data in each of the 12 data sequence as the training set, while the remaining data is used for model testing. To ensure that the performance of the baseline is not limited by the number of neurons, we also include a comparison with a deep ReLU neural network with four layers, each with 512 neurons. The experimental results are shown in Fig. 9.

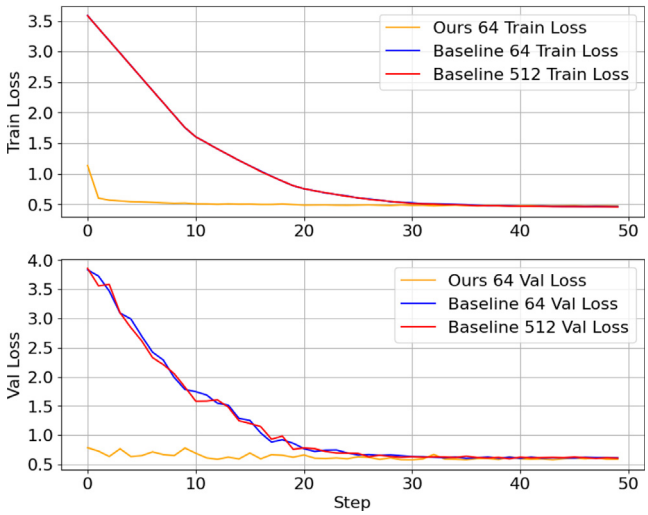

Fig. 9. The result of randomly selected data training and validation.

It is clearly shown that our proposed network converges faster, and the performance of the baseline with a single layer of 512 neurons is similar to that with a single layer of 64 neurons, indicating that the optimal performance has been reached. In terms of convergence speed, our proposed network shows a clear advantage. The experimental results demonstrate that incorporating prior knowledge into the network through the application of attention heads allows the network to achieve better initialization, consequently resulting in faster convergence.

Furthermore, in terms of the inference time, the inference speed for a single input sample of the proposed network is 0.15 ms on a CPU and 0.34 ms on a GPU, respectively. Our network has a sufficiently small and simple parameter size, resulting in similar time consumed to upload the input sample to a GPU as the time for direct inference on a CPU; thus, the performance on a GPU is worse than that on a CPU. The fast inference speed ensures the feasibility of the iterative method for solving T (Section 4.3), ensuring the speed and accuracy of our controller.

Our proposed neural network leverages prior knowledge by incorporating attention heads, leading to better initialization and achieving a good balance between model complexity and performance. While increasing the number of neurons can enhance model capacity to some extent, the experiments indicate that such improvements are limited and may increase computational costs. Thus, the proposed physics-inspired attention head is effective in achieving fast convergence without sacrificing performance.

5.2.2. Robustness

The robustness and generalization capability of the network are crucial for ensuring reliable control across diverse conditions. In this regard, we conducted further experiments to assess the proposed network's ability to adapt to varying settings. We redivided the datasets, randomly selecting seven whole data sequences for model training and the remaining sequences for data testing. Due to the data sequences were collected under different parameter settings, the disparity between the training and test sets is significant. This presents a challenge for the networks' generalization, as they may tend to overfit to the training domain.

Fig. 10 shows the comparison results on the robustness test. Even trained on a more challenging setting, the performance of the proposed method still performs well and converges fast. In contrast, the baseline method experiences more challenges in this experimental setting, exhibiting a relatively significant performance degradation. Specifically, at the final epoch, the baseline method achieves a MAE loss of 0.70, while the loss for the proposed method is only 0.49, indicating a notable improvement of 0.21 over the baseline. The experimental results prove that our method exhibits strong robustness. We believe this is mainly attributed to the contribution of the attention head, which enables the network to learn the physical model more effectively and quickly, thereby enhancing robustness.

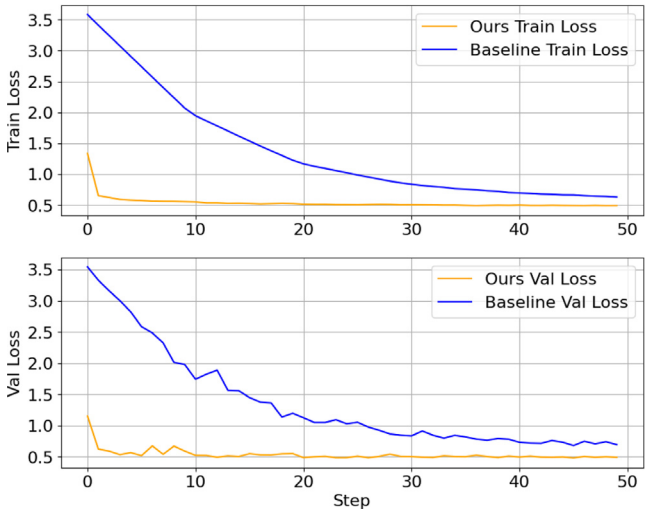

Fig. 10. The result of robustness test.

The ability to effectively generalize across diverse conditions, as demonstrated by the significant improvement in MAE loss, underscores the network's capacity to avoid overfitting and maintain performance even when faced with varying data distributions. This robustness is particularly valuable in real-world applications where the network must operate under unpredictable and dynamic conditions. The results provide strong evidence that our physics-inspired attention-based architecture contributes to its superior adaptability and resilience, making it a promising approach for learning-based control in complex environments.

5.2.3. Verification on attention head

The incorporation of the attention head aims to assist the network in discovering the underlying physics principles and focusing more on physical quantity T with the prior knowledge. In this experiment, the importance of each physical quantity used for inference in both the proposed network and the baseline is explored.

We utilized the absolute gradients obtained via backpropagation to represent the relative importance of each physical quantity towards the final results. The trained model was evaluated on 1000 randomly selected samples. Within each sample, the relative importance of each parameter was normalized to a range of 0 to 1, with a cumulative sum equal to 1. Fig. 11 presents the average importance of each physical quantity. It can be seen that the network using the attention head focuses more on the influence of T and τ. This is intuitive and

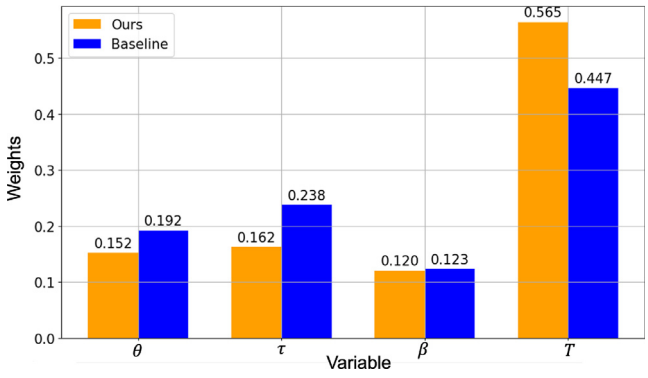

Fig. 11. The weights of focus of variables.

aligns with the dynamics discussed in Section 2, which proves the effectiveness of the attention head module. The findings underscore the value of the attention head in guiding the network's focus towards significant physical variables, thereby enhancing both interpretability and robustness. By prioritizing critical physical factors, the attention mechanism facilitates a more accurate and insightful understanding of the system's dynamics, leading to improved performance and reliability in prediction tasks. This demonstrates the practical benefits of incorporating attention mechanisms in neural networks for complex physical modeling.

5.3. Trajectory tracking performance

In this subsection, we tested the trajectory tracking performance of the proposed controller on a 30 deg slope. The controller used for comparison was the cascade PID controller from our previous work [4], which tracks the desired angular velocity ω_r. To ensure fairness, we used the same torque τ sequence and target states and started at the same location for the tests. The results are shown in Fig. 12. The tracking accuracy of our controller improved significantly compared to the baseline controller, as observed in the 40% reduction in RMSE. Notably, within the 0–6 s, when torque τ is present, our controller tracks the reference with a significantly smaller error than the baseline, showing the ability of the proposed neural network to accurately estimate the unknown disturbances and effectiveness of the proposed controller in compensating the estimated the disturbances.

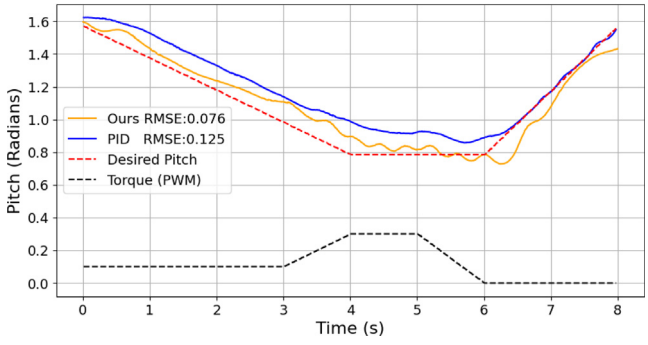

Fig. 12. Trajctory tracking performance of proposed controller and baseline controller.

However, after 6 s, when the torque was set to zero, our controller exhibited overshoot and noticeable tracking errors, while the baseline controller achieved accurate tracking. The main reason for this issue is the insufficient training data in low-torque scenarios. We focus on studying the effect of wheel movements on the robot's attitude, and more than 90% of the collected data contains wheel movement. Nonetheless, the angular error remains small throughout the operation (< 0.15 radian), consistent with our proven result of theoretically guaranteed stability.

While the proposed controller demonstrates strong performance in scenarios with significant disturbances, further refinement is necessary to handle low-torque situations effectively. This insight highlights the importance of comprehensive training data coverage for achieving robust performance across various operational conditions.

6. Conclusion and future works

In this work, we present a data-driven solution for attitude control of DoubleBee on slopes. Inspired by physics modeling, we add a channel-separated attention head to a multi-layer fully connected neural network using ReLU functions to predict the combined disturbance caused by ground effects, motor torque, axis shifts, and other factors during motion, thereby improving the neural network's training speed and robustness. Then, we design a sliding mode controller utilizing the predicted results and prove its stability and error bound. In practical experiments, the tracking error is reduced by nearly 40% compared to the baseline.

Also, in practical implementation, we face some challenges like limited control capability, data ground-truth acquisition, and real-time communication of high-frequency data. To address these, we assume specific body pitch angles in the decoupled mode to ensure sufficient control capability. For data ground-truth acquisition, we design trajectories that meet specific assumptions, use a Kalman filter, and apply peak alignment for accurate angular acceleration estimates. Lastly, we rely on USB cables and signal amplifier chips for stable data transmission.

In this work, we make a strong assumption on the moving direction of the robot (Assumption 1). Future work will consider arbitrary movement on slopes and focus on achieving full-state control of DoubleBee to expand its application prospects. Furthermore, considering various real-world constraints [24], the trajectory planning of DoubleBee will be considered in future work.

CRediT authorship contribution statement

Xinhang Xu: Validation, Formal analysis, Data curation, Conceptualization. **Yizhuo Yang:** Methodology, Investigation, Data curation, Conceptualization. **Muqing Cao:** Writing – review & editing, Formal analysis, Conceptualization. **Thien-Minh Nguyen:** Writing – review & editing. **Kun Cao:** Writing – review & editing, Conceptualization. **Lihua Xie:** Writing – review & editing.

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.

Data availability

The authors do not have permission to share data.

References

[1] E. Sihite, A. Kalantari, R. Nemovi, A. Ramezani, M. Gharib, Multi-Modal Mobility Morphobot (M4) with appendage repurposing for locomotion plasticity enhancement, Nat. Commun. 14 (1) (2023) 3323.

[2] A. Tagliabue, S. Schneider, M. Pavone, A.-a. Agha-mohammadi, Shapeshifter: A multi-agent, multi-modal robotic platform for exploration of titan, in: 2020 IEEE Aerospace Conference, IEEE, 2020, pp. 1–13.

[3] M. Cao, J. Zhao, X. Xu, L. Xie, AirCrab: A hybrid aerial-ground manipulator with an active wheel, 2024, arXiv preprint arXiv:2403.15805.

[4] M. Cao, X. Xu, S. Yuan, K. Cao, K. Liu, L. Xie, Doublebee: A hybrid aerial-ground robot with two active wheels, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 6962–6969.
[5] J. Lin, R. Zhang, N. Pan, C. Xu, F. Gao, Skater: A novel bi-modal bi-copter robot for adaptive locomotion in air and diverse terrain, IEEE Robot. Autom. Lett. (2024).
[6] M. Cao, X. Xu, K. Cao, L. Xie, System identification and control of the ground operation mode of a hybrid aerial–ground robot, Control Theory Technol. 21 (3) (2023) 458–468.
[7] L. Xie, Output feedback H∞ control of systems with parameter uncertainty, Int. J. Control 63 (4) (1996) 741–750.
[8] M. Ran, J. Li, L. Xie, A new extended state observer for uncertain nonlinear systems, Automatica 131 (2021) 109772.
[9] L. Xie, Y.C. Soh, Robust Kalman filtering for uncertain systems, Systems Control Lett. 22 (2) (1994) 123–129.
[10] H. Sira-Ramirez, M.A. Oliver-Salazar, On the robust control of buck-converter DC-motor combinations, IEEE Trans. Power Electron. 28 (8) (2012) 3912–3922.
[11] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, S. Levine, Model-based meta-reinforcement learning for flight with suspended payloads, IEEE Robot. Autom. Lett. 6 (2) (2021) 1471–1478.
[12] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, Neural lander: Stable drone landing control using learned dynamics, in: 2019 International Conference on Robotics and Automation, Icra, IEEE, 2019, pp. 9784–9790.
[13] B. Wang, Z. Ma, S. Lai, L. Zhao, Neural moving horizon estimation for robust flight control, IEEE Trans. Robot. (2023).
[14] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.
[15] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.
[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[17] C. Canudas-de Wit, P. Tsiotras, E. Velenis, M. Basset, G. Gissinger, Dynamic friction models for road/tire longitudinal interaction, Veh. Syst. Dyn. 39 (3) (2003) 189–226.
[18] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning, ICML-10, 2010, pp. 807–814.
[19] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: International Conference on Learning Representations, 2018.
[20] P.L. Bartlett, D.J. Foster, M.J. Telgarsky, Spectrally-normalized margin bounds for neural networks, Adv. Neural Inf. Process. Syst. 30 (2017).
[21] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, Bio-inspired Neurocomputing (2021) 203–224.
[22] A. Matus-Vargas, G. Rodriguez-Gomez, J. Martinez-Carranza, Ground effect on rotorcraft unmanned aerial vehicles: A review, Intell. Serv. Robotics 14 (1) (2021) 99–118.
[23] Y. Shtessel, C. Edwards, L. Fridman, A. Levant, et al., Sliding Mode Control and Observation, vol. 10, Springer, 2014.
[24] M. Cao, K. Cao, S. Yuan, T.-M. Nguyen, L. Xie, NEPTUNE: Nonentangling trajectory planning for multiple tethered unmanned vehicles, IEEE Trans. Robot. 39 (4) (2023) 2786–2804, http://dx.doi.org/10.1109/TRO.2023.3264950.

Xinhang Xu received his Bachelor of Engineering in Automation from South China University of Technology, Guangzhou, China and the M.Sc. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2023. His research interests include modeling and control of aerial and ground robots, motion planning, and multi-agent systems.

Yizhuo Yang received a B.S. degree in electrical and electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2021 and the M.Sc. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2022. He is currently pursuing a Ph.D. degree in electrical and electronic engineering at the Nanyang Technological University, Singapore. His research interests include multi-modal fusion and audio-based robotics perception.

Muqing Cao received his Bachelor of Engineering (Honors) in Aerospace Engineering from Nanyang Technological University, Singapore, in 2017 and his Ph.D. degree in Electrical and Electronic Engineering from Nanyang Technological University, Singapore, in 2023. He is a research fellow in Delta-NTU Corporate Laboratory, working on robot localization and navigation in environments with dense human traffic. His research interests include multi-robot systems, multi-robot coverage, tethered robots, motion planning, modeling and control of aerial and ground robots. He is the co-organizer of the CARIC challenge at CDC 2023.

Thien-Minh Nguyen is currently a Research Assistant Professor at the Centre for Advanced Robotics Technology Innovation (CARTIN), NTU, Singapore. He was the Wallenberg-NTU Presidential Postdoctoral Fellow at NTU and KTH Royal Institute of Technology from 2020 to 2023. He received the 2020 EEE Best Thesis Award (Innovation Track), and the 2nd Prize at the 2021 Hilti SLAM Challenge (Academic Track). His research interests include autonomous systems, robot perception, navigation, and learning. He is the co-organizer of the CARIC challenge at CDC 2023.

Kun Cao received the B.Eng. degree in mechanical engineering from Tianjin University, Tianjin, China, in 2016, and the Ph.D. degree in electrical and electronic engineering from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2021. He is currently the 2022 Wallenberg-NTU Presidential Postdoctoral Fellow with the School of Electrical and Electronic Engineering at Nanyang Technological University and the School of Electrical Engineering and Computer Science at KTH Royal Institute of Technology in Sweden. His research interests include localization, formation control, distributed optimization, and soft robotics.

Lihua Xie received the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, in 1992. Since 1992, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently a President's Chair Professor and Director of the Center for Advanced Robotics Technology Innovation. He served as the Head of the Division of Control and Instrumentation from July 2011 to June 2014. He held teaching appointments in the Department of Automatic Control, Nanjing University of Science and Technology from 1986 to 1989.

Dr Xie's research interests include robust control and estimation, networked control systems, multi-agent networks, localization and unmanned systems. He is an Editor-in-Chief for Unmanned Systems and has served as Editor of the IET Book Series in Control and Associate Editor of a number of journals, including IEEE Transactions on Automatic Control, Automatica, IEEE Transactions on Control Systems Technology, IEEE Transactions on Network Control Systems, and IEEE Transactions on Circuits and Systems-II. He was an IEEE Distinguished Lecturer (Jan 2012–Dec 2014). Dr Xie is a Fellow of the Academy of Engineering Singapore, IEEE, IFAC, and CAA, and Vice President of IEEE Control System Society.