A High-Payload Robotic Hopper Powered by Bidirectional Thrusters

Song Li, *Student Member, IEEE,* Songnan Bai, Ruihan Jia, Yixi Cai, Runze Ding, Yu Shi, Fu Zhang, *Member, IEEE,* and Pakpong Chirarattananon, *Member, IEEE*

***Abstract*—Mobile robots have revolutionized various fields, offering solutions for manipulation, environmental monitoring, and exploration. However, payload capacity remains a limitation. This paper presents a novel thrust-based robotic hopper capable of carrying payloads up to 9 times its own weight while maintaining agile mobility over less structured terrain. The 220 gram robot carries up to 2 kg while hopping—–a capability that bridges the gap between high-payload ground robots and agile aerial platforms. Key advancements that enable this high-payload capacity include the integration of bidirectional thrusters, allowing for both upward and downward thrust generation to enhance energy management while hopping. Additionally, we present a refined model of dynamics that accounts for heavy payload conditions, particularly for large jumps. To address the increased computational demands, we employ a neural network compression technique, ensuring real-time onboard control. The robot's capabilities are demonstrated through a series of experiments, including leaping over a high obstacle, executing sharp turns with large steps, as well as performing simple autonomous navigation while carrying a 730 g LiDAR payload. This showcases the robot's potential for applications such as mobile sensing and mapping in challenging environments.**

***Index Terms*—Hopping, legged robots, high payload, neural network, SLIP, autonomous navigation.**

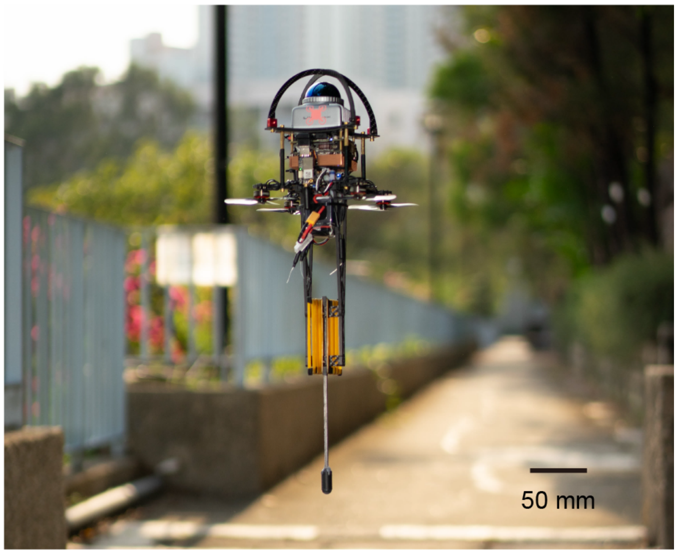


Fig. 1. A 220 g thrust-based hopping robot constructed from a quadcopter (160 g) and a passive leg (60 g) demonstrating an ability to carry a 730 g LiDAR payload for mapping applications. Despite a thrust limit of only 440 g, the hopper is able to carry as much as 2 kg.

I. Introduction

Mobile robots have emerged as versatile platforms for a wide range of applications, making impacts on various fields including manipulation [1], [2], environmental monitoring [3]–[5], subterranean [6] and space [7] exploration, and disaster response [8]. Their ability to navigate and operate in environments that are dangerous or inaccessible to humans makes them invaluable tools across various domains.

When considering complex and unstructured environments where wheeled robots struggle, among the most widely adopted mobile robot platforms for real-world applications are legged robots and aerial robots, each offering distinct advantages and limitations. Legged robots, such as quadrupeds [6], [7] and humanoids [9], excel in their ability to navigate uneven terrain [9]–[11] and carry substantial payloads [12], [13]. Their multi-jointed limbs allow for precise foot placement and force control, enabling them to climb stairs, step over obstacles, and maintain stability in challenging environments [14]. However, they typically exhibit slower speeds compared to aerial platforms and may struggle in extremely cluttered environments. Conversely, aerial robots offer rapid movement and the ability to bypass obstacles by flying over them [15]–[17]. Nevertheless, they are often constrained by limited payload capacity [18], endurance [19], and reduced manipulation capabilities [2], [20].

To harness the strengths of both legged and aerial platforms, researchers have proposed hybrid robotic platforms that augment humanoid robots with thrusters or flight components, enabling them to overcome large obstacles or traverse gaps through short-duration flights [21], [22]. An alternative strategy, which forms the basis of this work, involves equipping a quadrotor with a passive, springy leg [23]–[26]. This design leverages the quadrotor's existing actuators for both flight and terrestrial locomotion. It maintains a compact form factor and retains flight efficiency, without requiring additional actuators that would increase both weight and power consumption.

Incorporating a passive leg transforms a quadcopter into a

This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region of China (grant numbers CityU 11209024 and PolyU R5006-23). Song Li, Songnan Bai, Ruihan Jia, Runze Ding, Yu Shi, Pakpong Chirarattananon are with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong SAR, China. Yixi Cai, Fu Zhang are with the Mechatronics and Robotic Systems Laboratory, Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China. Pakpong Chirarattananon is also with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong SAR, China, Centre for Nature-Inspired Engineering, City University of Hong Kong, Hong Kong SAR, China, and Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada. Song Li and Songnan Bai contributed equally to this work. Corresponding author: Pakpong Chirarattananon, Email: p.chirarattananon@utoronto.ca

thrust-based hopper, facilitating continuous jumping akin to leg-actuated hopping robots [27]–[35]. The stance dynamics of this hybrid system can be approximated using a spring-mass model, drawing parallels to the well-studied Spring-Loaded Inverted Pendulum (SLIP) model [36]–[38], albeit with notable differences in actuation and control strategies. The elastic leg enables energy recovery during landing, significantly enhancing efficiency compared to conventional flying robots. PogoDrone demonstrated a 20% reduction in power consumption compared to hovering [23], though its efficiency was limited by position control methods requiring a thrust-to-weight ratio exceeding unity. More recent platforms such as Hopcopter [24], PogoX [25], and MultiMo-MHR [26] have further improved efficiency through model-based hopping controllers, enabling continuous locomotion with thrust-to-weight ratios below one. Despite these advances, the maximum payload capacity of thrust-based hopping robots remained unexplored and unvalidated.

This paper presents a 220 g legged quadcopter capable of carrying payloads up to 2 kg (Fig. 1 and Video S1), effectively overcoming the severe payload limitations of conventional flying robots. Compared to its 35-g predecessor [24] (see also Table S1 in the Supplementary Materials), the scaling up dramatically increases both absolute payload capacity (2 kg vs 8 g) and relative payload-to-mass ratio (9.1 vs. 0.23), improving autonomy by allowing integration of a 730-g LiDAR and onboard computer for autonomous navigation.

Two key technical advancements over previous thrust-based hopping robots [23]–[26] underpin this work. First, we introduce bidirectional thrusters [39]–[43], enabling the robot to accelerate towards the ground during the descending phase of its hop. This additional actuation mechanism allows the robot to compensate for the increased energy dissipation resulting from heavier payloads, significantly expanding its load-bearing capabilities. To fully leverage this modification, the hopping controller is also redesigned to regulate downward thrust, ensuring controlled landings and improved energy efficiency. Second, we present a more comprehensive derivation of the stance phase dynamics. Unlike our previous model [24], this approach incorporates the weight of both the robot and its payload during the stance phase. While this consideration notably increases the model's complexity [37], it is essential for maintaining satisfactory hopping performance as the robot's mass becomes substantial. To address the increased computational demands of this more sophisticated model, we employ a neural network (NN) compression technique [44], ensuring efficient onboard control. These enhancements enable the robot to perform aggressive and agile maneuvers—-such as jumping over obstacles and executing sharp turns with large steps, even with significant payloads-that were not demonstrated in not previously demonstrated in [23]–[26].

The remainder of this paper is organized as follows: Section II details the design and operating principles of the hopping platform with bidirectional thrusters. Section III provides an analysis of the enhanced payload capability achieved through the combination of elastic leg mechanisms and bidirectional thrust. Section IV presents the comprehensive dynamic model of the hopping robot, including the refined stance phase dynamics. Section V describes the neural network-based approach for efficient onboard control. Section VI outlines the control strategies for maintaining hopping height and trajectory following with bidirectional thrusters and varying payloads. Experimental results demonstrating the robot's performance in carrying heavy payloads and navigating autonomously are presented in Section VII. Finally, Section VIII concludes the paper with a discussion of the implications of this work and potential future directions.

II. Hopping Platform with Bidirectional Thrusters

A. Prototype with Bidirectional Thrusters

The design of the robot is based on our previous thrust-based hopper, the Hopcopter [24], combining a quadcopter with a passive leg (Fig. 2A). The quadrotor consists of a flight control board (Bitcraze, Crazyflie bolt 1.1) and four sets of brushless motors with 3.5 inch propellers. In addition, we employ bidirectional electronic speed controllers (ESCs, Flycolor X-Cross 45A). This allows each rotor to generate forward (upward) or reverse (downward) thrust depending on the spinning direction as shown in Fig. 2D.

To account for the difference in aerodynamic efficiencies of propellers in the forward and reverse configurations, we tested four types of propellers (diameters between 3 to 4 inches) for thrust and torque coefficients in both configurations following the protocols in [45] as detailed in Supplementary Materials. The results reveal that the ratio of thrust coefficients in the reverse direction to forward direction, defined as γ, varies from -0.36 to -0.56. The propeller type with the highest absolute γ (0.56) was selected.

As a hopper, the upper leg section is attached underneath the quadcopter with a telescopic lower section and a rubber foot. As illustrated in Fig. 2B, the lower leg is constrained to the upper leg via two sets of bearings, constraining the movement to a single degree of freedom while minimizing the sliding friction. The telescopic motion is spring-loaded with elastomer (rubber bands). In the default state when the leg length is l_0, this elastomer is pre-streched by a length l_p. The leg contraction results in further elongation of the elastomer beyond l_p (Fig. 2C). As used in [24], this strategy permits more elastic potential energy to be temporarily stored in the leg structure upon landing under the same leg contraction. The physical parameters of the hopping robot are listed in Table S2. The mass of the robot (without payload) is $m_r = 220$ g , with the component breakdown listed in Table S3.

B. Thrust-based Hopping Strategy

Unlike previous hopping robots that used a moving mass [35], [46] or leg motors [29], [31], this robot utilizes the thrust generated by propellers to regulate the hopping locomotion as the robot alternates between aerial and terrestrial phase. In addition, the energy injection happens in the aerial phase, instead of the stance as leg-actuated hopper [28], [32], [47].

During the aerial phase, energy loss due to air drag is negligible, as the robot's vertical speed remains relatively low (ranging from 3.3 m/s to 3.9 m/s, corresponding to a

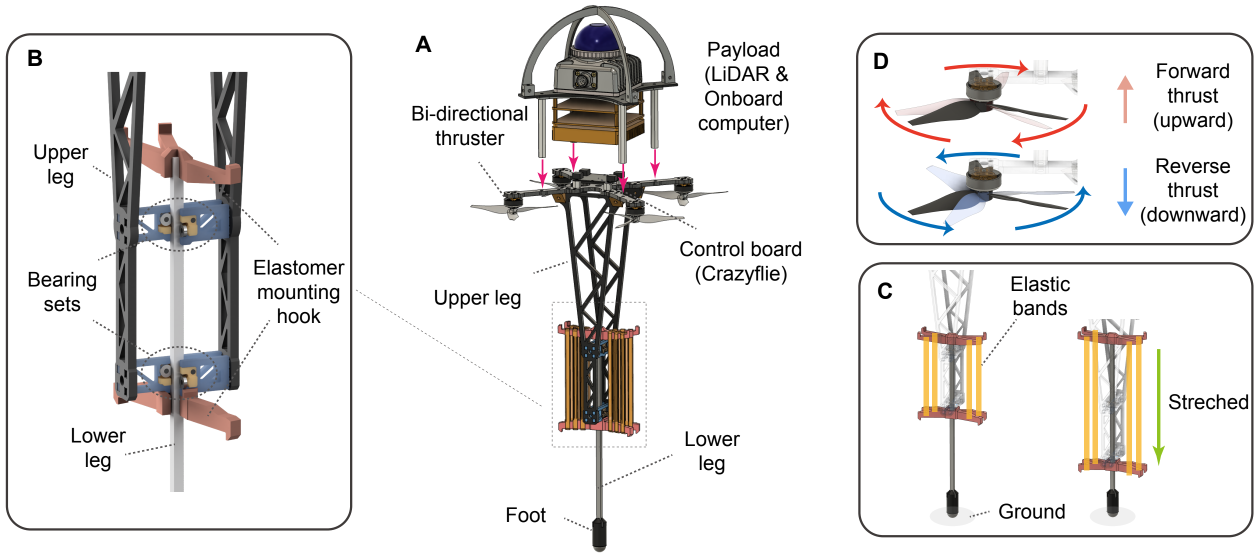


Fig. 2. Hopping robot design and major components. (A) Model of the prototype based on a multirotor vehicle with four bidirectional thrusters. Payload can be attached on top of the robot. (B) An unactuated telescopic leg is affixed underneath the quadcopter. (C) Upon landing (ground contact), the leg undergoes compression, stretching the rubber bands. This results in the hopping motion. (D) Bidirectional electronic speed controller permits the robot to generate both upward and downward thrust by reversing the propeller's spinning direction. This further enhances the payload carrying capacity for hopping.

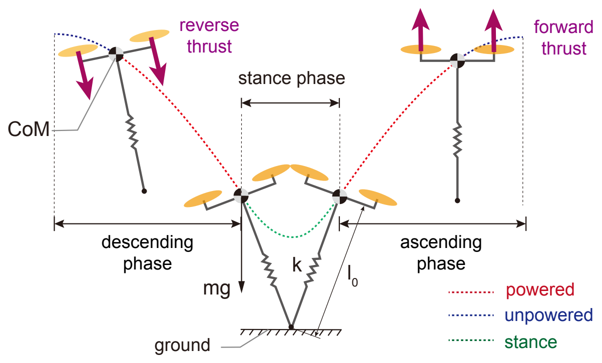


Fig. 3. Operating principles of a thrust-based hopping robot. As a hybrid dynamical system, the robot alternates between aerial and terrestrial phases, with landing and takeoff transitions. Thanks to the unactuated spring-loaded leg, the hopper swings passively in the stance phase. To retain the desired hopping height and control the hopping trajectory, propelling thrust is employed to compensate for viscous losses and attitude control in the aerial phase.

jumping height of 0.55 m to 0.75 m), and its small cross-sectional area minimizes aerodynamic resistance. As the robot transitions from the aerial phase to the passive stance phase, the ground reaction force induces the leg to retract, elongating the elastomer. The landing impact on the ground and the viscous dissipation of the leg mechanism during stretching and compressing lead to a proportion of energy loss in the stance phase [24]. Consequently, the takeoff speed will be smaller than the landing speed. Without compensation, the total energy of the system would gradually decrease with each hopping step.

To retain the hopping height, energy is injected into the system during the aerial phase via propelling thrust as illustrated in Fig. 3. Forward (upward) thrust is applied when the robot ascends, immediately after the robot takes off. This powered ascent strategy was similarly adopted in [24]. Thanks to the use of bidirectional thrusters introduced in this work, the robot gains the ability to add more energy into the system via powered descent. This is by applying reverse (downward) thrust prior to landing to accelerate the fall. The use of powered descent is beneficial when the robot carries a heavy payload, which incurs larger energy loss in the stance phase.

Apart from compensating for the energy lost, propelling torque is responsible for stabilizing the attitude of the robot in flight. This is used to manipulate the hopping trajectory. Unlike flying, the hopping trajectory is influenced by the attitude of the robot at the moment it lands on the ground. This hopping principle applicable to the robot with a single passive leg is detailed in Section IV, leading to the development of the hopping controller presented in Section VI

III. Enhanced Payload Capability Through Bidirectional Thurst

In this section, we analyze the enhanced payload capability of our hopping robot, which is achieved through the combination of an elastic leg mechanism and a bidirectional thrust strategy. We present a simple model to quantify this enhancement and compare our results with experimental data and other mobile robots.

Consider a thrust-based hopping robot of mass m_r carrying a payload of mass m_p, the amount of energy for the robot to reach the hopping height h is $(m_r + m_p)gh$. We define η as the proportion of energy lost per jump, accounting for energy dissipation at landing due to ground impact and leg mechanism.

For the robot to maintain a continuous jump at height h, it must be actuated to compensate for the energy loss. This is ideally achieved by applying upward (forward) propelling thrust T_f during ascent and downward (reverse) thrust T_r during descent when the robot is in the air as depicted in Fig. 3. The total energy injected per cycle is $(T_f + T_r)h$.

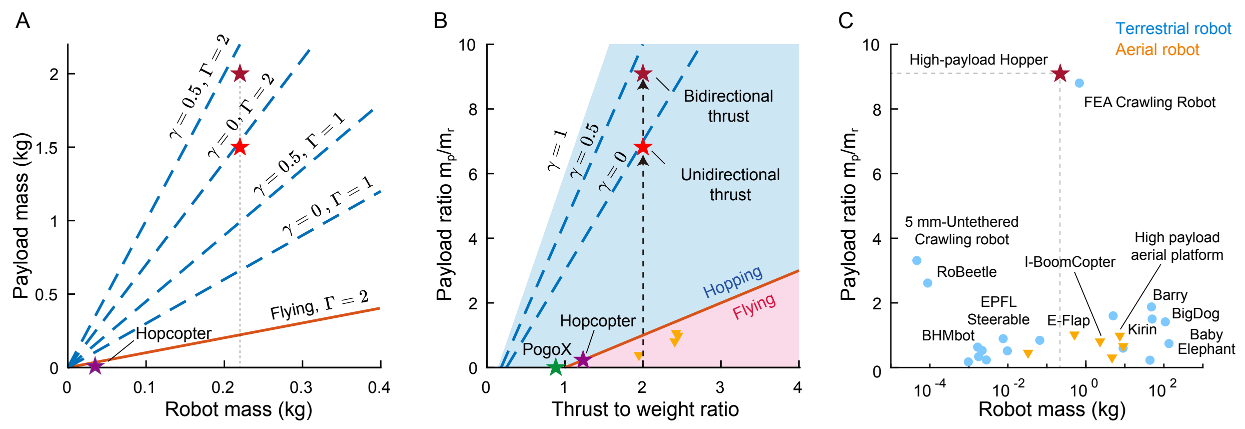


Fig. 4. Payload capability of various robots. (A) Absolute payload limit versus mass for thrust-based hopping robots according to their thrust-to-weight ratios Γ and bidirectional thrust efficiency γ, assuming $\eta = 0.25$. The plot shows that scaling up the 35-g robot from [24] to 220 g dramatically increases the payload from 8 g to 2 kg, allowing the robot to carry LiDAR for autonomous navigation. (B) Theoretical payload limit of thrust-based hopping robots. The plot shows the payload limit (m_p normalized by the original robot's mass m_r as payload ratio m_p/m_r) against the thrust-to-weight ($\Gamma = T/m_r$) ratio. The proposed robot with $\Gamma = 2$ experimentally demonstrates its ability to carry a payload 9.1 times as heavy as its own weight when hopping. This limit is increased with the use of directional thrusters ($\gamma > 0$). (C) Experimental payload ratios demonstrated by mobile robots across size scales (wheeled robots excluded). The plot highlights the superior payload capacity of the proposed thrust-based hopper, which also demonstrates an ability to overcome obstacles and traverse over less structured terrain. For complete data and references, refer to Table S4.

To determine the energy loss, we start from the apex. The total energy just before landing includes the potential energy at the apex and the contribution from the reverse thrust during descent $(m_r + m_p)gh + T_r h$. Thus, the energy lost at landing is $\eta[(m_r + m_p)gh + T_r h]$.

To maintain hopping at steady height h, the energy balance condition is:

$$(T_f + T_r)h = \eta\left[(m_r + m_p)gh + T_r h\right], \tag{1}$$

independent of h as it cancels out on both sides. Notice the difference in the impact of forward and reverse thrusts. The energy injected during the descent from the reverse thrust is partially dissipated upon landing ($\eta T_r h$). Energy from the forward thrust applied during ascent and is not subject to this loss. This means the use of reverse thrust is less efficient than forward thrust.

To determine the maximum payload mass m_p, we consider the maximum forward thrust using the notion of thrust-to-weight ratio $\Gamma = T_f/m_r g$. In addition, since the reverse thrust is usually lower than the forward thrust, we introduce $\gamma = T_r/T_f \leq 1$ to indicate the performance of the bidirectional thruster. Employing these definitions, (1) becomes:

$$\frac{m_p}{m_r} = \Gamma\frac{1 + (1-\eta)\gamma}{\eta} - 1. \tag{2}$$

To understand the significance of (2), we first consider a regular flying robot with an identical thrust-to-weight ratio Γ. In such a case, the maximum payload mass is given by $m_p/m_r = \Gamma - 1$. In comparison, for thrust-based hopping, the thrust-to-weight ratio is effectively amplified by a factor of $\left[1 + (1-\eta)\gamma\right]/\eta$, implying the payload mass is upper bounded by $m_p < (\Gamma\left[1 + (1-\eta)\gamma\right]/\eta - 1)m_r$. Fig. 4A and B illustrate the absolute and relative payload limits achievable by thrust-based hopping robots for selected values of Γ and γ, assuming an energy loss factor of $\eta = 0.25$. Notably, the payload capacity of hopping robots can be several times greater than that of flying robots with the same thrust-to-weight ratio.

This amplification is influenced by two key factors: γ and η. The energy loss factor η is largely determined by the leg efficiency and can be dependent on the hopping height. For instance, previous 35-gram hopping robot with unidirectional thrust ($\gamma = 0$) [24] has an estimated η of approximately 0.23 for jump heights from 0.68 m to 0.96 m. This implies an amplification factor of approximately 4.4. Theoretically, without considering control constraints, the 35 gram hopper with a thrust limit of 42 gf could carry a payload of up to 150 grams when control is not considered. Therein, the robot was able to hop while carrying a light payload with the total mass $m_r + m_p = 43$ g, slightly over the thrust limit. However, the true payload capability was not analyzed or examined.

In this work, we later experimentally demonstrate that the proposed robot in Fig. 2A, with a mass $m_r = 220$ g and a maximum thrust of 440 gf ($\Gamma = 2$), is able to operate reliably when carrying a 2.0 kg payload, thanks to the use of bidirectional thrusters. This equates to a payload ratio m_p/m_r of 9.1 and a total mass of 2220 g, 5.0 times as large as the thrust limit. According to (2), the result implies a loss factor η below 0.28. Moreover, without the use of reverse thrust ($\gamma = 0$), the robot can still carry a payload as heavy as 1.5 kg, implying a similar loss factor of $\eta = 0.26$. Compared to the 35-gram hopper in [24], this scaling up and the integration of bidirectional thrusters dramatically increases the absolute payload limit of the robot as shown in Fig. 4A.

Comparing with other aerial and terrestrial robots (excluding wheeled and tracked robots) with power autonomy [10], [12], [13], [18], [27], [48]–[66] across size scales (Fig. 4C, see Table S4 for details), the proposed robot shows a substantially higher payload ratio while maintaining an ability to traverse less structured terrain. Since absolute payload capacity is strongly influenced by robot size, payload ratio provides a more meaningful basis for comparison [12], [67], [68], particularly across different scales. While not all referenced robots are optimized for heavy payloads, payload capacity remains a critical performance metric for the applications involving transportation [12] and autonomy [58]. Among existing robots, a crawling robot powered by fluidic elastomer actuators [57] displayed an exceptional payload ratio of 8.8. However, the crawling motion of the robot in [57] is far less agile (14 mm/s) and incompatible with uneven ground. In addition, although there exist high-performance racing drones with exceptionally high thrust-to-weight ratios ($\sim 4-5$ [16], [69] or as large as 12 [70]), they have yet to demonstrate heavy payload carrying flight. More importantly, such systems would similarly and dramatically benefit from hopping when it comes to payload ratio as indicated by (2) and Fig. 4B.

IV. Hopping Dynamic Model

The dynamics of the hopping robot can be divided into two primary phases: the aerial phase and the stance phase. The aerial phase is further subdivided into ascending and descending states, as depicted in Fig. 3. Transitions between the aerial and stance phases are marked as landing and takeoff events. Within the aerial phase, the robot can be in either powered or unpowered flight, regardless of whether it is ascending or descending. The powered flight corresponds to the robot actively generating thrust.

A. Model of the Flight Dynamics

Neglecting aerodynamic drag, in the aerial phase, the robot is regarded as a rigid body of mass $m = m_r + m_p$, indistinguishable from a regular quadrotor. The translational dynamic equation can be written into

$$m\ddot{\mathbf{p}} = \mathbf{R}\mathbf{e_3}(-1)^k \sum_{i=1}^{4} f_i + mg\mathbf{e}_3, \tag{3}$$

where $\mathbf{p} = [x, y, z]^T$ denotes the position of the robot in the inertial frame, $\mathbf{R}$ is the rotation matrix associated with the body frame, $\mathbf{e_3} = [0, 0, 1]^T$ is a basis vector, $k \in \{0, 1\}$ indicates the spinning direction of the propellers ($k = 0$ for forward thrust and $k = 1$ for reverse thrust), $\sum_{i=1}^{4} f_i$ is the total thrust magnitude, and g is the free-fall acceleration.

The attitude dynamics in the aerial phase is governed by

$$\mathbf{I}\dot{\boldsymbol{\omega}}_{\boldsymbol{b}} + \boldsymbol{\omega}_{\boldsymbol{b}} \times \mathbf{I}\boldsymbol{\omega}_{\boldsymbol{b}} = \boldsymbol{\tau}_{\boldsymbol{p}}, \tag{4}$$

where $\mathbf{I}$ is the inertial tensor, $\boldsymbol{\omega}_{\boldsymbol{b}} = [\omega_x, \omega_y, \omega_z]^T$ is the body-centric angular velocity, $\boldsymbol{\tau}_{\boldsymbol{p}}$ is the torque generated by the propellers.

Given the configuration of the robot with the propeller arm length d, the torque in (4) can be derived as

$$\boldsymbol{\tau}_{\boldsymbol{p}} = \sum_{i=1}^{4} \mathbf{l}_i \times \mathbf{e_3} f_i + \mathbf{e_3} M_i, \tag{5}$$

where $\mathbf{l_i}$ indicates the location of the i^{th} rotor, for example, $\mathbf{l_1} = d/\sqrt{2}[1, -1, 0]$. M_i is the propeller's induced torque.

Due to the asymmetric profile of the propeller blades, individual rotor thrust f_i and induced torque M_i are dependent on the operating condition (forward or reverse thrust) as testified in Supplementary Materials. Given the same motor commands or spinning rate, the magnitudes of both f_i and M_i are reduced by a factor of γ and γ_m, respectively, for reverse operations. This characteristic must be taken into account when the low-level controller determines the motor commands from the desired thrust and torque values.

B. Air-Ground Transition

As a hybrid dynamical system, the robot exhibits a discrete dynamic behavior at the air-ground transitions.

Immediately after landing (timestamp: t_{LD}), we consider the non-slip pointy foot as a spherical joint. The stationary ground contact point (pivot) can be regarded as the origin of the stance phase model. Defining $\mathbf{p}_s = [x_s, y_s, z_s]^T$ as the position of the Center of Mass (CoM) from the pivot, the initial position and velocity of the stance phase can be computed from the aerial phase state according to

$$\mathbf{p}_s(t_{\text{LD}}) = \mathbf{R}(t_{\text{LD}})\mathbf{e_3}l_0 = \mathbf{z}_b(t_{\text{LD}})l_0, \tag{6}$$

$$\dot{\mathbf{p}}_s(t_{\text{LD}}) = \dot{\mathbf{p}}(t_{\text{LD}}), \tag{7}$$

where l_0 is the undeformed leg length. The landing timestamp t_{LD} can be solved for based on $\mathbf{p}(t)$ from (3) and $\mathbf{R}(t)$ from (4) using

$$\mathbf{e}_3^T\mathbf{p}(t_{\text{LD}}) = \mathbf{e}_3^T\mathbf{R}(t_{\text{LD}})\mathbf{e_3}l_0. \tag{8}$$

For the transition back from stance to aerial phase, the position and translational velocity at takeoff time are continuous and can be written as

$$\mathbf{p}(t_{\text{TO}}) = \mathbf{p}_s(t_{\text{TO}}) + \mathbf{p}(t_{\text{LD}}), \tag{9}$$

$$\dot{\mathbf{p}}(t_{\text{TO}}) = \dot{\mathbf{p}}_s(t_{\text{TO}}), \tag{10}$$

with t_{TO} being the takeoff time. This is when the elastic leg recoils to its original length or

$$\|\mathbf{p}_s(t_{\text{TO}})\| = l_0. \tag{11}$$

Meanwhile, the attitude and angular velocity of the robot at takeoff moment can be found as

$$\mathbf{R}(t_{\text{TO}})\mathbf{e}_3 = \mathbf{z}_b(t_{\text{TO}}) = \mathbf{p}_s(t_{\text{TO}})/l_0, \tag{12}$$

$$\boldsymbol{\omega}_{\boldsymbol{b}}(t_{\text{TO}}) \times \mathbf{p}_s(t_{\text{TO}}) = \dot{\mathbf{p}}_s(t_{\text{TO}}). \tag{13}$$

The dynamics of $\mathbf{p}_s$ depend on the elastic leg mechanism as described below.

C. Complete Stance Phase Model

Unlike previous SLIP-based models that neglect the weight of the robot in the stance phase [24], [71], [72], the consideration for carrying a heavy payload renders the influence of the term mg important in the short stance duration even when compared with the large elastic force.

In the following model, the robot is regarded as a point mass rotating around the non-slip ground contact point rather than as a rigid body [37], [73]–[76]. Thanks to the design with a relatively long leg (nominal length: l_0), during the stance phase, the moment of inertia around the pivot point is much larger than the inertia tensor of the body itself, allowing us to

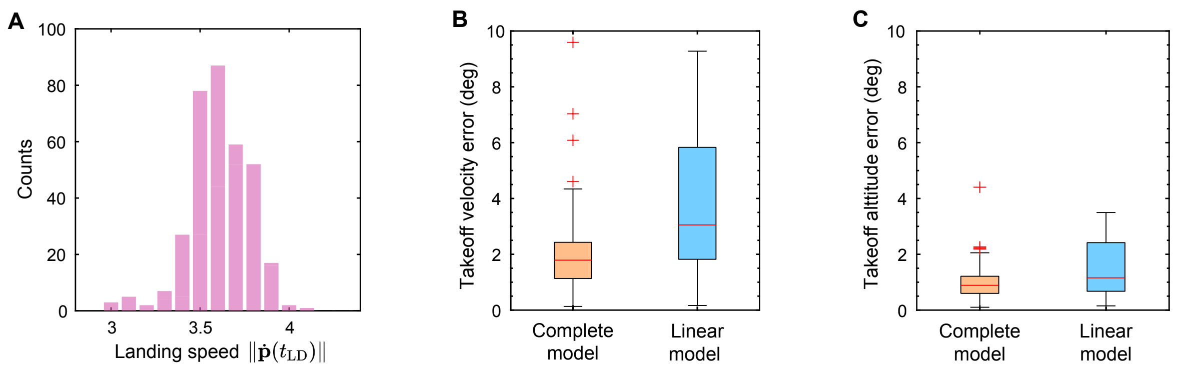


Fig. 5. Identification and validation of the stance phase model. (A) The distribution of landing speed from the drop test. (B) Angular errors between the predicted and measured takeoff velocity for the complete stance phase model and empirical linear model. (C) Angular errors between the measured and predicted takeoff body axis for the complete stance phase model and empirical linear model. The boxes represent the interquartile range (IQR), with the median line. Whiskers extend to the lowest and highest data points within 1.5 times the IQR from the box edges.

neglect the latter [24]. Consequently, the equation of motion is simplified to

$$m\ddot{\mathbf{p}}_s = \frac{\mathbf{p}_s}{\|\mathbf{p}_s\|}\left(-k_s\left(l - l_0 - l_p\right) - \epsilon\dot{l}\right) - mg\mathbf{e}_3, \qquad (14)$$

where k_s is the linear stiffness of the elastomer, $l = \|\mathbf{p}_s\|$ is the instantaneous leg length, and ϵ is a linear damping coefficient.

D. Validation of the Stance Phase Model

To validate the stance phase model in (14) and identify relevant physical parameters, we experimentally collected the landing and takeoff states of the robot by dropping the robot from different heights (between 0.42 m and 0.88 m) and orientations. We collected a total of 343 data points (average height: 0.68 m). In this test, the total weight of the robot was 824 g, including a 500 g dummy payload and an enlarged battery. The resultant landing speeds varied from 3 m/s to over 4 m/s as illustrated in Fig. 5A.

The trajectory data captured from the motion capture system were processed and encoded. The landing state consists of the landing velocity $\dot{\mathbf{p}}(t_{\mathrm{LD}})$ and the body axis $\mathbf{R}(t_{\mathrm{LD}})\mathbf{e}_3 = \mathbf{z}_b(t_{\mathrm{LD}})$ as present in (6) and (7). Similarly, the takeoff state includes the velocity $\dot{\mathbf{p}}_s(t_{\mathrm{TO}})$ from (10) and body attitude $\mathbf{z}_b(t_{\mathrm{TO}})$ or $\mathbf{p}_s(t_{\mathrm{TO}})/l_0$ from (12). Together, these vectors are visually illustrated in Fig. 6A.

Based on the complete stance phase in (14), we employed a Genetic Algorithm to identify unknown parameters: $\boldsymbol{\Theta} = \{l_0, l_p, k_s, \epsilon\}$. The objective function is

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}}\left[w_\Theta\epsilon_{zb}(\boldsymbol{\Theta}) + \epsilon_v(\boldsymbol{\Theta})\right], \qquad (15)$$

where $w_\Theta = 2$ is a positive weight, $\epsilon_{zb}(\boldsymbol{\Theta})$ is the Root Mean Square (RMS) angular error of the takeoff body axis (the difference between the experimental measurements and the model predictions based on the landing states), and $\epsilon_v(\boldsymbol{\Theta})$ is the RMS error of the takeoff velocity axis. The identified parameters are listed in Table S2.

To evaluate the accuracy of the model, we inspect the prediction errors. The best-fitted parameters result in the average prediction error of $0.9°$ for the takeoff body axis $\mathbf{z}_b(t_{\mathrm{TO}})$ and $2.0°$ for the direction of the takeoff velocity $\dot{\mathbf{p}}_s(t_{\mathrm{TO}})$, plotted as ‘Complete model’ in Fig. 5B and C. This verifies that the dynamics described by (14) accurately captures the actual behavior of the robot.

V. Compression of Inverse Stance Phase Model for Onboard Control

With the initial state from (6) and (7), and identified parameters, the 3-DoF stance phase dynamics can be simulated forward to obtain the takeoff state. The inverse map, providing the landing state corresponding to the desired takeoff state, is required for stabilizing and controlling the robot’s hopping trajectory.

One approach to obtain the inverse map is to search for the landing state that results in the desired takeoff state (e.g., shooting methods), requiring the stance phase dynamics to be simulated several times to yield the solution. Unfortunately, simulating the stance phase dynamics forward using (14) on the flight control board is computationally demanding and cannot be achieved at a sufficiently high frequency. In our previous work [24], we circumvented this issue by neglecting the mg term, which effectively reduces the dimension of the system and allows for an analytical solution to be obtained. However, this simplification is not ideal for a robot carrying a heavy payload, as the weight term becomes significant.

In this section, we examine two alternative methods to address this challenge. The first approach is to empirically fit the model to the inverse map linearly, substantially decreasing computational complexity. However, this linear model is still based on an assumption that is valid when the influence of the term mg is negligible. This strategy, or a look-up table, becomes unattainable when the weight term is considered as the map becomes nonlinear and the dimension of the map increases from 1 to 5. To overcome this issue, we then introduce a neural network-based compression model to encode the solution of (14). This approach provides high accuracy over a wide range of landing states and allows the map to be used onboard under limited computational power.

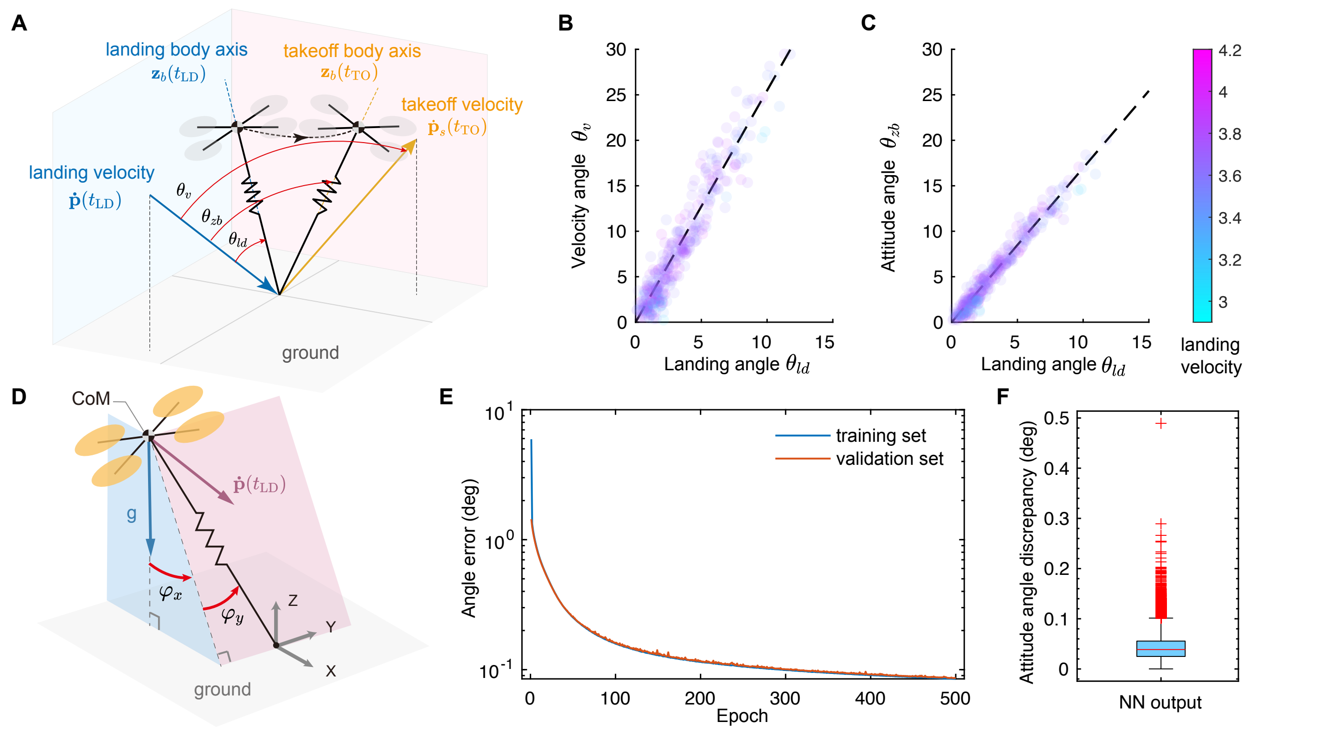


Fig. 6. Compression of the stance phase model. (A) Vectors and angles used to identify the state, including velocity vectors (not to scale), body axes of the robot at landing and takeoff. (B) Landing angle versus takeoff velocity angle. (C) Landing angle versus takeoff attitude angle. (D) Definitions of the angles used as the input and output for the neural network. (E) Average landing angle error produced by the training set and validation set while learning for 500 epochs. (F) Angle discrepancy between the predicted landing attitude of the trained neural network and the test data set.

A. *Linear Model*

In [24], it was established that when the weight of the robot in (14) is small compared with the elastic force, all four vectors associated with the landing and takeoff states, namely $\dot{\mathbf{p}}(t_{\mathrm{LD}})$ $\mathbf{z}_b(t_{\mathrm{LD}})$, $\dot{\mathbf{p}}_s(t_{\mathrm{TO}})$, and $\mathbf{z}_b(t_{\mathrm{TO}})$, are coplanar. We define (i) the landing angle θ_{ld} as the angle between the landing velocity $\dot{\mathbf{p}}(t_{\mathrm{LD}})$ and attitude $\mathbf{z}_b(t_{\mathrm{LD}})$, (ii) θ_v as the angle between the landing and takeoff velocities, and (iii) θ_{zb} as the angle between the landing velocity and takeoff body axis as illustrated in Fig 6A.

Based on previous findings [24], [77], [78], the takeoff state could be predicted by the landing state (velocity and landing angle) for a small range of θ_{ld}. As a result, all four vectors are approximately coplanar and the stance phase map could be further simplified. As plotted in Fig. 6B and C, both θ_v and θ_{zb} empirically exhibit an approximately linear relationship against θ_{ld} as follows.

$$\theta_v = \kappa_v \theta_{ld}, \tag{16}$$

$$\theta_{zb} = \kappa_{zb} \theta_{ld}, \tag{17}$$

where $\kappa_v = 2.54$ and $\kappa_{zb} = 1.69$ are empirically determined constants.

We can employ the linear relations in (16) and (17) to predict the takeoff state ($\dot{\mathbf{p}}_s(t_{\mathrm{TO}})$, $\mathbf{z}_b(t_{\mathrm{TO}})$) from the landing state ($\dot{\mathbf{p}}(t_{LD})$, $\mathbf{z}_b(t_{LD})$) without directly applying the full dynamics or (14). The linear model predicts the takeoff velocity direction and body axis with the average angular errors of 3.8° and 1.5°, with the error distributions illustrated by boxplots in Fig. 5B and C. As anticipated, the errors are somewhat larger than the predictions made by the complete model in (14), partially attributed to the negligence of the weight time.

B. *Inverse Stance Phase Model with Neural Network Compression*

Here, we use a neural network to construct an inverse map of the complete stance phase model. This is required by the hopping controller as described later in Section VI-B. To capture the impact of the term $mg\mathbf{e}_3$ in (14) in the model, we incorporate information on the direction of the takeoff velocity with respect to gravity. The network then predicts the required landing attitude. In addition, we take into consideration different payload masses. This allows the model to be readily implemented for the robot carrying different payloads without retraining.

As the NN is designed to function with the hopping controller, it takes a 6×1 vector as input and produces a 2×1 output vector. The input contains information on the landing velocity and the desired direction of the takeoff velocity. The network then outputs the body attitude of the robot that would result in the desired direction of the takeoff velocity (this is provided by the hopping controller) for that particular landing velocity.

More specifically, the input is

$$\mathbf{x}_N = \begin{bmatrix} \|\dot{\mathbf{p}}(t_{\mathrm{LD}})\| \\ \arccos\left(\dfrac{\mathbf{e}_3^T \dot{\mathbf{p}}(t_{\mathrm{LD}})}{\|\dot{\mathbf{p}}(t_{\mathrm{LD}})\|}\right) \\ \dot{\mathbf{p}}(t_{\mathrm{TO}})/\|\dot{\mathbf{p}}(t_{\mathrm{TO}})\| \\ m_p \end{bmatrix}, \tag{18}$$

where the first element is the landing speed and the second is the angle between the velocity and gravity. Together, they emphasize the relative direction of gravity. The third entry is the takeoff velocity direction and the last is the payload mass.

The output is defined as

$$\mathbf{y}_N = [\varphi_x, \varphi_y]^T, \tag{19}$$

in which the elements are Euler angles representing the landing attitude ($\mathbf{z}_b(t_{LD}) = [\cos(\varphi_y)\sin(\varphi_x), \sin(\varphi_y), \cos(\varphi_x)\cos(\varphi_y)]^T$) that results in the landing velocity in $\mathbf{x}_\mathrm{N}$. The angle definitions are visualized in Fig. 6D.

1) Training Data Generation: To train the neural network model, we first generate training data from the complete stance phase model instead of relying on the limited experimental data. The numerical simulation (4th-order Runge-Kutta method, time step: 1 ms) provides a large range of data that uniformly spans the input space, reducing bias in the process.

To cover a wide range of possible scenarios, we first consider the landing state, consisting of three dimensions of data (landing speed, angle, and payload mass) from the input vector and two from the output (φ_x, φ_y), amounting to five independent variables. We assign a specific range for these five independent elements, within which we uniformly generated 15 discrete points, yielding 15^5 sets of training data. For the landing speed, the range is from 1 to 5 m/s, corresponding to a hopping height from 0.05 m to 1.28 m. For the angle with respect to the vertical, the range is from $0°$ to $30°$. For payload mass, the input varies from 0.5 kg to 1.5 kg. For landing attitude specified by $\mathbf{y}_\mathrm{N}$, both Euler angles span a range of $\pm 15°$. These ranges encompass conditions of most low-speed hopping. After the simulation with the complete stance phase model, we obtain the remaining elements in $\mathbf{x}_\mathrm{N}$, corresponding to the takeoff velocity. The resultant takeoff speeds vary from 1 m/s to 5 m/s. Remark that the takeoff speed could be as high as the landing speed despite viscous losses when the center of mass of the robot when taking off is lower than at landing. In the end, we obtain all eight elements for $N = 15^5$ pairs of input/output vectors.

2) Training of the neural network: We train a neural network to represent the inverse map of the stance phase based on the generated simulation data, mapping the 6×1 input $\mathbf{x}_\mathrm{N}$ to the 2×1 output $\mathbf{y}_\mathrm{N}$. The architecture of the network consists of six fully connected layers structured as $6 \rightarrow 64 \rightarrow 256 \rightarrow 256 \rightarrow 32 \rightarrow 2$, employing rectified linear unit (ReLU) activations. This design incorporates a total of $91,170$ parameters, significantly fewer than the $15^5 = 759,375$ parameters required for a direct lookup table of the inverted dynamics.

To optimize the neural network's output, we defined a loss function that minimizes the angular discrepancy between the predicted landing attitude vector and the training data, represented by the dot product of the two unit vectors. This loss function is expressed as:

$$\min \frac{1}{N} \sum_{i}^{N} \left(1 - \mathbf{z}_b^T\left(\boldsymbol{y}_N(i)\right) \hat{\mathbf{z}}_b\left(\mathbf{x}_\mathrm{N}(i)\right)\right)^2 \tag{20}$$

where the index i refers to the i^{th} set of data. Here $\mathbf{z}_b^T\left(\boldsymbol{y}_N(i)\right)$ is the landing attitude corresponding to $\mathbf{y}_\mathrm{N}$ taken from the training data, and $\hat{\mathbf{z}}_b\left(\mathbf{x}_\mathrm{N}(i)\right)$ is the predicted landing attitude based on the output of the network, computed using the current set of hyperparameters. The minimization is over a set of hyperparameters of the entire network.

The neural network was implemented and trained using PyTorch, employing the Adam optimizer with a learning rate of 0.01. To balance the contributions of different vector ranges to the total loss, we applied batch-wise weighting. The dataset, comprising 15^5 samples, was randomly shuffled and partitioned into training (80%), validation (10%), and test (10%) sets. To mitigate overfitting, we monitored performance on the validation set during training. Training progressed for 2000 epochs, with convergence monitored using an angle discrepancy metric derived from the loss function. As illustrated in Fig. 6E, this metric converged to $0.09°$ for both training and validation sets after 500 epochs, ultimately reaching $0.06°$ at the conclusion of training.

To assess the trained network, we evaluated its performance on the test set ($75,937$ samples). We compared the predicted attitude $\hat{\mathbf{z}}_b(t_{LD})$ constructed from the output vector with the ground truth and inspected the angular errors. The results demonstrate that, excluding a few outliers, the errors are consistently below $0.1°$ (Fig. 6F). This is negligible when compared against the takeoff angular errors computed from the predictions from the complete model and the experimentally collected data (velocity: $4.2°$, attitude: $2.0°$, Fig. 5B and C).

VI. Hopping Controller For Variable Payload

In this section, we present the control architecture for stabilizing the attitude and trajectory of the thrust-based hopping robot. The controller leverages the fact that the stance phase is extremely short and passive, meaning no active control is exerted during this phase. This controller is hinged on the reverse map of the stance phase, compressed in to the trained neural network. The key method employed is controlling the landing attitude, which serves as the initial condition for the stance phase. This effectively manipulates the direction of the takeoff attitude, takeoff velocity, and subsequent aerial phase trajectory.

The controller is divided into four primary components: (i) a height controller, (ii) a high-level position controller, (iii) thrust and attitude management, and (iv) a low-level attitude controller, as illustrated in Fig. 7A. The hopping height controller ensures the robot reaches the desired hopping height by determining the appropriate duration for applying forward and reverse thrust during the ascending and descending aerial phase. This calculation determines the amount of energy to be injected into the system to compensate for any losses. The high-level position controller, making use of the trained neural network, executes once per hopping cycle, calculating the landing attitude based on the discrepancy between the predicted and desired landing positions. The outputs from both the height and position controllers are fed into the thrust and attitude management module, which determines the thrust direction, magnitude, and robot orientation throughout

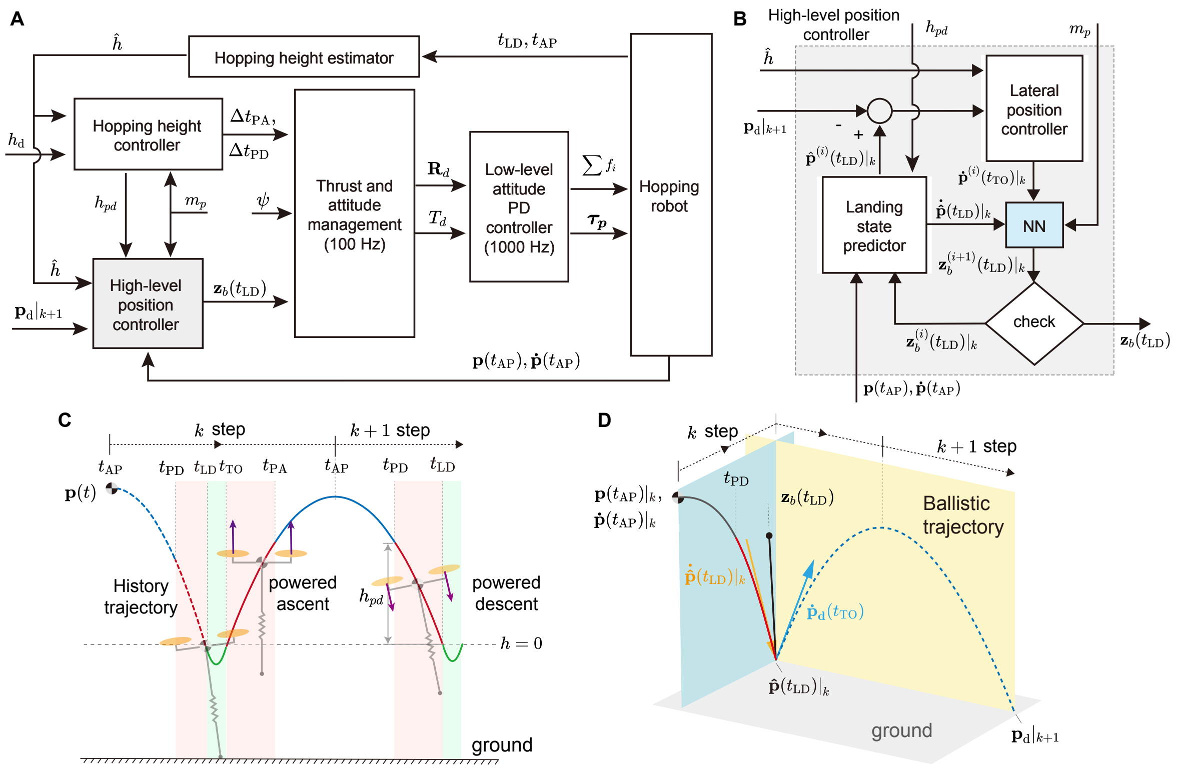

A
Hopping height estimator
$t_{\mathrm{LD}}, t_{\mathrm{AP}}$
$\hat{h}$
h_{d}
Hopping height controller
Δt_{PA}, Δt_{PD}
Thrust and attitude management (100 Hz)
$\mathbf{R}_d$
T_d
Low-level attitude PD controller (1000 Hz)
$\sum f_i$
$\boldsymbol{\tau}_p$
Hopping robot
h_{pd}
m_p
ψ
High-level position controller
$\mathbf{p}_{\mathrm{d}}|_{k+1}$
$\mathbf{z}_b(t_{\mathrm{LD}})$
$\mathbf{p}(t_{\mathrm{AP}}), \dot{\mathbf{p}}(t_{\mathrm{AP}})$
B
High-level position controller
Lateral position controller
Landing state predictor
NN
check
$\dot{\mathbf{p}}^{(i)}(t_{\mathrm{LD}})|_k$
$\dot{\mathbf{p}}^{(i)}(t_{\mathrm{TO}})|_k$
$\dot{\mathbf{p}}(t_{\mathrm{LD}})|_k$
$\mathbf{z}_b^{(i+1)}(t_{\mathrm{LD}})|_k$
$\mathbf{z}_b^{(i)}(t_{\mathrm{LD}})|_k$
C
k step
$k+1$ step
t_{AP} t_{PD} t_{LD} t_{TO} t_{PA} t_{AP} t_{PD} t_{LD}
$\mathbf{p}(t)$
History trajectory
powered ascent
powered descent
h_{pd}
$h=0$
ground
D
k step
$k+1$ step
t_{PD}
$\mathbf{p}(t_{\mathrm{AP}})|_k$, $\dot{\mathbf{p}}(t_{\mathrm{AP}})|_k$
$\mathbf{z}_b(t_{\mathrm{LD}})$
Ballistic trajectory
$\dot{\mathbf{p}}(t_{\mathrm{LD}})|_k$
$\dot{\mathbf{p}}_{\mathrm{d}}(t_{\mathrm{TO}})$
$\hat{\mathbf{p}}(t_{\mathrm{LD}})|_k$
ground
$\mathbf{p}_{\mathrm{d}}|_{k+1}$

Fig. 7. Architecture of the hopping controller and related diagrams. (A) The overall control architecture involves the hopping height controller, high-level position controller, thrust and attitude management module, and low-level attitude controller. (B) Implementation of the high-level position controller. (C) Diagram illustrating the timing for powered ascent and powered descent used for the hopping height controller. (D) Diagram showing the process of position control, involving the landing state of the current step, takeoff velocity, and the subsequent landing position.

the entire aerial phase. Subsequently, the thrust and attitude commands are relayed to the onboard low-level controller for motor actuation.

In addition to the use of the NN-compressed reverse stance phase model, the primary distinction from our previous work [24] lies in the use of reverse thrust, which leads to two changes: (i) the coupling of landing attitude and reverse thrust in the landing state prediction within the high-level position controller, and (ii) the incorporation of an additional component in thrust management to account for the powered descent phase. These modifications are necessary because the use of reverse thrust during the powered falling phase induces lateral acceleration when the robot aligns its body axis with the desired landing attitude. This lateral acceleration causes the actual landing point to deviate from the expected point (based on a free-falling trajectory), consequently influencing the selection of the optimal landing attitude.

A. Hopping Height Controller

As indicated by (1), the hopping height is closely related to the amount of energy in the system, which can be regulated through powered ascent and descent during the flight phase. Therefore, the height control strategy involves two main steps: nominal height prediction and height control. First, we present a method for predicting the nominal hopping height of the next step when the robot is landing. Then, we detail our control strategy, which adjusts the thrust duration of the robot in the aerial phase. The amount of adjustment is based on the nominal hopping height and the desired hopping height, taking into consideration the payload weight. This ensures the desired hopping height is approximately reached.

1) Nominal Hopping height: The nominal hopping height $\hat{h}$ is estimated based on the current amount of energy of the robot. This serves as a reference value for the hopping height controller to evaluate whether additional energy is required for the robot to reach the desired height h_{d}. The estimation is carried out cyclically at the moment (t_{LD}) of the landing of each hop step, based on the recorded apex velocity $\dot{\mathbf{p}}(t_{\mathrm{AP}})$, the velocity at highest point of the aerial phase, and the descent duration $t_{\mathrm{LD}} - t_{\mathrm{AP}}$ (refer to Fig. 7C). We determine t_{AP} by monitoring the vertical speed, and the landing timestamp t_{LD} is detected through a spike in the accelerometer reading caused by ground impact.

In this step, the nominal height is computed assuming the kinetic energy of the robot at landing is subsequently converted to the potential energy, after taking into consideration the loss factor η that captures the energy loss in the leg mechanism in the stance phase. This yields

$$mg\hat{h} = \frac{1-\eta}{2} m \left(\dot{\mathbf{p}}(t_{\mathrm{AP}})^T \dot{\mathbf{p}}(t_{\mathrm{AP}}) + g^2 (t_{\mathrm{LD}} - t_{\mathrm{AP}})^2 \right), \quad (21)$$

$$\hat{h} = \frac{1-\eta}{2g} \left(\dot{\mathbf{p}}(t_{\mathrm{AP}})^T \dot{\mathbf{p}}(t_{\mathrm{AP}}) + g^2 (t_{\mathrm{LD}} - t_{\mathrm{AP}})^2 \right). \quad (22)$$

In the absence of control, $\hat{h}$ will be the maximally feasible hopping height of the robot after it takes off. Furthermore, the conservation of energy also informs us about the speed of the robot when it takes off from

$$\frac{1}{2}m\|\dot{\mathbf{p}}_{\mathrm{d}}(t_{\mathrm{TO}})\|^2 = mg\hat{h}, \tag{23}$$

with the nominal height $\hat{h}$ obtained when $\dot{\mathbf{p}}_{\mathrm{d}}(t_{\mathrm{TO}})$ is upright.

2) Hopping height control: Depending on the nominal hopping height in (22) and the desired height h_{d}, our height control strategy regulates two key parameters: the duration of powered ascent ($t_{\mathrm{PA}} - t_{\mathrm{TO}}$) and powered descent ($t_{\mathrm{LD}} - t_{\mathrm{PD}}$). The approach varies depending on the payload weight.

First, we employ (2) to determine the maximum payload mass m_p that is compatible with the unidirectional thrust limit (taking $\gamma = 0$ and $\eta = 0.25$), resulting in $m_p = (\Gamma/\eta + 1)\, m_r g = 1.54$ kg.

For lighter payload within this limit, we employ only powered ascent, setting t_{PD} in Fig. 7C (and later on in (28)-(31)) to t_{LD}. The powered ascent time $t_{\mathrm{PA}} - t_{\mathrm{TO}}$ is chosen based on the nominal height $\hat{h}$ from (22) and the setpoint h_{d}. Following the strategy in [24], the decision is made as follows.

If the nominal hopping height is higher than the desired height ($\hat{h} \geq h_{\mathrm{d}}$), we forego the powered ascent as no additional energy needs to be injected into the system. Otherwise, we apply a short burst of powered ascent immediately after takeoff to maximize the power input. The required energy input to achieve the desired height is $mg(h_{\mathrm{d}} - \hat{h})$. Given that the vertical speed of the robot immediately after takeoff is approximately $\sqrt{2g\hat{h}}$, the required powered ascent time $t_{\mathrm{PA}} - t_{\mathrm{TO}}$ can be approximated from

$$T_f\sqrt{2g\hat{h}}(t_{\mathrm{PA}} - t_{\mathrm{TO}}) = mg(h_{\mathrm{d}} - \hat{h}), \tag{24}$$

where T_f is the magnitude of the forward thrust force. Its value depends on the payload mass. To meet the energy condition in (1), T_f should be greater than ηmg. Concurrently, T_f cannot exceed its maximum limit ($\Gamma m_r g$) while leaving some room for torque generation and attitude control.

For payload exceeding the unidirectional thrust limit of 1.54 kg, we employ both powered ascent and descent. The powered descent was designed to inject more energy into the system. The additional energy needed during the powered descent phase could be evaluated by subtracting the maximum energy that could be replenished during the ascent phase $\Gamma m_r g\hat{h}$ from the expected energy lost $\eta mg\hat{h}$ in each hop. It follows that when the energy is injected with the maximum reverse thrust $T_r = \gamma\Gamma m_r g$, the required powered descent distance h_{pd}, defined as the height at which downward thrust is activated (see Fig. 7C), can be estimated from:

$$(\gamma\Gamma m_r g)h_{pd} = \eta mg\hat{h} - \Gamma m_r g\hat{h}, \tag{25}$$

$$h_{pd} = \left(\frac{\eta m}{\Gamma m_r} - 1\right)\frac{\hat{h}}{\gamma}. \tag{26}$$

Based on the maximum payload mass condition in (2), the total mass satisfies $m < \Gamma(1+\gamma)m_r/\eta$. Hence, in (26), we have the term $\eta m/\Gamma m_r - 1 < \gamma$ and $h_{pd} < \hat{h}$. This means the powered descent distance is shorter than $\hat{h}$ as expected. In practice, T_r is set to be lower than the maximum limit of $\gamma\Gamma m_r g$ in order to maintain sufficient attitude control capability. Therefore, (25) is only approximately valid. Taking a cue from (25), we introduce a tuning parameter κ_h to yield h_{pd} as a function of $\hat{h}$ according to

$$h_{pd} = \kappa_h\hat{h}. \tag{27}$$

Expressing h_{pd} as $\mathbf{e}_3^T(\hat{\mathbf{p}}(t_{\mathrm{PD}}) - \hat{\mathbf{p}}(t_{\mathrm{LD}}))$, the duration for powered descent $t_{\mathrm{LD}} - t_{\mathrm{PD}}$ can also be later computed using (28)-(31). This use of powered descent increases the vertical speed of the robot at landing, pushing the robot closer to the desired height h_{d} in the subsequent jump.

B. High-level Position Control

Similar to other hopping robots capable of continuous jumps [24], [28] and unlike aerial robots, the aerial locomotion of the robot in this work is without direct position control. Instead, the hopping trajectory is regulated through the jumping direction or the direction of the takeoff velocity following the short stance phase. As previously mentioned, the takeoff velocity is highly dependent on the attitude of the robot at landing.

The high-level position control is performed in three steps: (i) a landing state predictor estimates the current (cycle k) landing state based on the apex position and velocity, (ii) the errors between the desired landing position for cycle $k+1$ and the current landing position is used to calculate a desired aerial trajectory which is represented by a takeoff velocity, (iii) the landing attitude needed to achieve the takeoff velocity is iteratively estimated.

The coupling between the landing attitude and the takeoff velocity, which influences the subsequent landing location, complicates the control strategy. The controller must solve for the desired landing attitude iteratively as illustrated by Fig. 7B. The iterative approach, indexed by a bracketed superscript $(\cdot)^{(i)}$, initializes with the landing attitude $\mathbf{z}_b^{(0)}(t_{\mathrm{LD}})|_k$. It then computes the upcoming landing position $\hat{\mathbf{p}}^{(0)}(t_{\mathrm{LD}})|_k$ and corresponding takeoff velocity $\dot{\mathbf{p}}_{\mathrm{d}}^{(0)}(t_{\mathrm{TO}})|_k$. This leads to the expected next landing position $\hat{\mathbf{p}}^{(0)}(t_{\mathrm{LD}})|_{k+1}$. Based on the difference between this and the setpoint position, the controller updates the landing attitude, yielding $\mathbf{z}_b^{(1)}(t_{\mathrm{LD}})|_k$. The process is repeated until the position error of $\hat{\mathbf{p}}(t_{\mathrm{LD}})^{(i)}|_{k+1}$ is marginalized.

Below, we first elaborate on how the landing position of the current cycle k is predicted. This is computed once per hop. Then, the position control is detailed. These steps are iteratively executed once per hop. For brevity, we omit the iteration index $(\cdot)^{(i)}$ from the following equations.

1) Landing state prediction: The upcoming landing state, comprising the landing position and velocity is predicted ($\hat{\mathbf{p}}(t_{\mathrm{LD}})|_k, \dot{\hat{\mathbf{p}}}(t_{\mathrm{LD}})|_k$) once per hopping cycle after the robot reaches the apex during the aerial phase.

As shown in Fig. 7D, the predictor assumes a free-fall descent followed by a powered descent (at time t_{PD}), neglecting air drag. Given the apex position $\mathbf{p}(t_{\mathrm{AP}})$ and velocity $\dot{\mathbf{p}}(t_{\mathrm{AP}})$, the position $\hat{\mathbf{p}}(t_{\mathrm{PD}})|_k$ and velocity $\dot{\hat{\mathbf{p}}}(t_{\mathrm{PD}})|_k$ of the robot at the beginning of the powered descent phase are estimated as

$$\hat{\mathbf{p}}(t_{\text{PD}})|_k = \mathbf{p}(t_{\text{AP}})|_k + \dot{\mathbf{p}}(t_{\text{AP}})|_k (t_{\text{PD}} - t_{\text{AP}}) - \frac{1}{2}\mathbf{e}_3 g (t_{\text{PD}} - t_{\text{AP}})^2, \tag{28}$$

$$\dot{\hat{\mathbf{p}}}(t_{\text{PD}})|_k = \dot{\mathbf{p}}(t_{\text{AP}})|_k - g\mathbf{e}_3 (t_{\text{PD}} - t_{\text{AP}}), \tag{29}$$

where t_{AP} is the apex timestamp and t_{PD} marks the start of the powered descent phase. The beginning of the powered descent phase t_{PD} is empirically obtained when the robot falls to a height h_{pd} (designated by the height controller in (26)) and is computed from $\mathbf{e}_3^T \hat{\mathbf{p}}(t_{\text{PD}}) = h_{pd}$. Hence, the time the robot spends in free fall is $t_{\text{PD}} - t_{\text{AP}}$.

During the powered descent with a reverse thrust T_r, the robot is assumed to be aligned with the desired landing attitude $\mathbf{z}_b(t_{\text{LD}})|_k$. It experiences an additional acceleration of $T_r/m\mathbf{z}_b$. Thus, the estimated landing position $\hat{\mathbf{p}}(t_{\text{LD}})$ and velocity $\dot{\hat{\mathbf{p}}}(t_{\text{LD}})$ are

$$\hat{\mathbf{p}}(t_{\text{LD}})|_k = \hat{\mathbf{p}}(t_{\text{PD}})|_k + \dot{\hat{\mathbf{p}}}(t_{\text{PD}})|_k (t_{\text{LD}} - t_{\text{PD}}) - \frac{1}{2}\left(\mathbf{e}_3 g + \mathbf{z}_b(t_{\text{LD}})|_k T_r/m\right)(t_{\text{LD}} - t_{\text{PD}})^2, \tag{30}$$

$$\dot{\hat{\mathbf{p}}}(t_{\text{LD}})|_k = \dot{\hat{\mathbf{p}}}(t_{\text{PD}})|_k - \left(\mathbf{e}_3 g + \mathbf{z}_b(t_{\text{LD}})|_k T_r/m\right)(t_{\text{LD}} - t_{\text{PD}}), \tag{31}$$

where the landing time t_{LD} is calculated by imposing the flat ground condition $\mathbf{e}_3^T \hat{\mathbf{p}}(t_{\text{LD}})|_k = 0$.

2) Desired takeoff velocity: To ensure the robot follows a prescribed hopping trajectory, we implement an iterative lateral position control strategy based on the predicted landing position of the next hop $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$. This is then used to compute the required takeoff velocity $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ that would approximately and subsequently navigate the robot to the desired position $\dot{\mathbf{p}}_{\text{d}}(t_{\text{TO}})|_k$.

Similar to [24], our strategy completely relies on regulating the takeoff velocity of the robot via manipulating the landing attitude, without directly controlling the position of the robot in the aerial phase. This means that starting from the apex, the landing position of the robot in the current cycle (indexed as k) is largely determined and predicted as $\hat{\mathbf{p}}(t_{\text{LD}})|_k$ by (30). Nevertheless, the next landing point $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$ can be altered by strategically regulating the takeoff velocity $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ in the current step. This target takeoff velocity is realized by controlling the robot's landing attitude $\mathbf{z}_b(t_{\text{LD}})|_k$ as it serves as the initial conditions for the passive stance phase.

Starting from the predicted landing position $\hat{\mathbf{p}}(t_{\text{LD}})|_k$ and a hypothetical takeoff velocity $\dot{\mathbf{p}}(t_{\text{TO}})|_k$, the landing position $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$ in the next hopping step is estimated by assuming an entirely ballistic trajectory. As the flight phase time is equal to $2\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k/g$, the next landing location is expected to be

$$\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} = \hat{\mathbf{p}}(t_{\text{LD}})|_k + \dot{\mathbf{p}}(t_{\text{TO}})|_k \left(\frac{2\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k}{g}\right) - \frac{1}{2} g\mathbf{e}_3 \left(\frac{2\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k}{g}\right)^2, \tag{32}$$

which can be re-arranged into

$$\dot{\mathbf{p}}(t_{\text{TO}})|_k = \frac{g\left(\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} - \hat{\mathbf{p}}(t_{\text{LD}})|_k\right)}{2\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k} + \mathbf{e}_3\left(\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k\right), \tag{33}$$

In this form, we attempt to obtain an explicit expression of $\dot{\mathbf{p}}(t_{\text{TO}})|_k$. To do so, we express $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ in terms of horizontal and vertical components. With the energy equation in (23), we have

$$\left\| \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}^T \dot{\mathbf{p}}(t_{\text{TO}})|_k \right\|^2 + (\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k)^2 = 2g\hat{h}. \tag{34}$$

Substituting $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ from (33) into (34) yields

$$\frac{\|\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} - \hat{\mathbf{p}}(t_{\text{LD}})|_k\|^2 g^2}{4(\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k)^2} + (\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k)^2 = 2g\hat{h}, \tag{35}$$

where we have applied the fact that $\mathbf{e}_3^T\left(\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} - \hat{\mathbf{p}}(t_{\text{LD}})|_k\right) = 0$ (flat ground condition). The result permits us to solve for the vertical component of $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ from

$$\frac{\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k}{\sqrt{g\hat{h}}} = \sqrt{1 - \sqrt{1 - \left\|\frac{\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} - \hat{\mathbf{p}}(t_{\text{LD}})|_k}{2\hat{h}}\right\|^2}}. \tag{36}$$

Then, the horizontal components can be evaluated by projecting (33) along the directions of $\mathbf{e}_1$ and $\mathbf{e}_1$, making use of (36) as

$$\mathbf{e}_{1,2}^T\dot{\mathbf{p}}(t_{\text{TO}})|_k = \frac{g\mathbf{e}_{1,2}^T\left(\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1} - \hat{\mathbf{p}}(t_{\text{LD}})|_k\right)}{2\mathbf{e}_3^T\dot{\mathbf{p}}(t_{\text{TO}})|_k}. \tag{37}$$

Together, (36) and (37) provide a full expression of the takeoff velocity as a function of $\hat{\mathbf{p}}(t_{\text{LD}})|_k$, which is dependent on $\mathbf{z}_b(t_{\text{LD}})|_k$ through (30).

3) Iterative implementation of the position controller: In this final step, we iteratively determine the landing attitude $\mathbf{z}^{(i)}(t_{\text{LD}})|_k$ and the takeoff velocity $\dot{\mathbf{p}}^{(i)}(t_{\text{TO}})|_k$ that together minimizes the landing position error. This procedure is also detailed in Algorithm 1 in the Supplementary Materials.

First, after the robot has reached the apex, we initialize the landing attitude to be the vertical: $\mathbf{z}_b^{(0)}(t_{\text{LD}})|_k = \mathbf{e}_3$. This is used to by (30) and (33) to predict the current $\hat{\mathbf{p}}(t_{\text{LD}})|_k$ and the next landing positions $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$.

The landing position at cycle $k+1$ is highly dependent on the takeoff velocity $\dot{\mathbf{p}}(t_{\text{TO}})|_k$. Hence, we regard $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$ as the setpoint position and use (36) and (37) to estimate the the required takeoff velocity $\dot{\mathbf{p}}(t_{\text{TO}})|_k$. This is by substituting $\hat{\mathbf{p}}(t_{\text{LD}})|_{k+1}$ in those equations with the reference $\mathbf{p}_{\text{d}}|_{k+1}$ (if the robot is unable to reach the setpoint in a single hop, an intermediate setpoint is introduced as described in Supplementary Materials). Once the corresponding value of $\dot{\mathbf{p}}(t_{\text{TO}})|_k$ is evaluated, it is used as an input vector $\mathbf{x}_{\text{N}}$ for the NN. The output $\mathbf{y}_{\text{N}}$ of the network is translated to an auxiliary attitude vector $\hat{\mathbf{z}}_b^{(i)}(t_{\text{LD}})$. We then use the outcome to incrementally update the desired landing attitude at step k for the next iteration according to

$$\mathbf{z}_b^{(i+1)}(t_{\text{LD}}) = \frac{\mathbf{z}_b^{(i)}(t_{\text{LD}}) + w_z\hat{\mathbf{z}}_b^{(i)}(t_{\text{LD}})}{\|\mathbf{z}_b^{(i)}(t_{\text{LD}}) + w_z\hat{\mathbf{z}}_b^{(i)}(t_{\text{LD}})\|}, \tag{38}$$

where $w_z = 0.5$ is an update weight.

Next, we repeat steps 2 and 3 to recompute the desired takeoff velocity and update the desired landing attitude in an iterative fashion. The process is terminated when $\|\mathbf{z}_b^{(i)}(t_{\text{LD}}) - \mathbf{z}_b^{(i-1)}(t_{\text{LD}})\| < 10^{-4}$. In practice, this usually takes $3-4$

iterations. After the convergence, the final value of $\mathbf{z}_b^{(i)}(t_{\mathrm{LD}})$ is passed on to the attitude controller as elaborated below.

C. Thrust and Attitude Management Module

In this step, thrust T and attitude $\mathbf{R}$ of the robot in the aerial phase are controlled according to the operational states: unpowered flight, powered descent, and powered ascent, as depicted in Fig. 3. Depending on the commanded thrust and attitude, the low-level control is then responsible for computing the control torque and corresponding motor commands.

While hopping, the thrust direction switches twice each cycle when power descent is adopted. In the meantime, the thrust magnitude in the powered ascent and descent is provided by the height controller. Whereas during the unpowered flight phase, the desired thrust is nominally zero, complying with the assumption of ballistic trajectories used for the next landing position estimation in (32).

To compute the attitude setpoint $\mathbf{R}_{\mathrm{d}}$ from the desired body axis $\mathbf{z}_b$, we make use of the fact that $\mathbf{z}_b = \mathbf{R}\mathbf{e}_3$ is independent of the yaw angle ψ when the rotation matrix is parameterized by roll ϕ, pitch θ, and yaw ψ angles as $\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\phi)\mathbf{R}_x(\theta)$. Treating ψ as an arbitrary setpoint, the desired roll and pitch rotations can be determined from $\mathbf{z}_b$ and ψ according to

$$\mathbf{R}_y(\phi)\mathbf{R}_x(\theta)\mathbf{e}_3 = \mathbf{R}_z(\psi)^T\mathbf{z}_b. \tag{39}$$

As previously described, the desired attitude while the robot is ascending is $\mathbf{z}_b = \mathbf{e}_3$. In contrast, while descending (powered or unpowered), the desired attitude is determined by the landing position controller as the desired landing attitude. The target thrust and attitude are then taken by the low-level attitude controller to produce the motor comments.

D. Low-level Attitude Controller

Given the desired thrust T_d and attitude setpoint $\mathbf{R_d}$ from the thrust and attitude management module, we employ the following controller to regulate the robot's attitude and generate appropriate motor commands.

The attitude controller aims to correct both attitude and angular rate errors, taking into consideration a possibly large attitude error. Hence, we compute the attitude error as $\mathbf{R}^T\mathbf{R}_{\mathrm{d}}$, which represents the rotation between the desired attitude $\mathbf{R_d}$ and the current attitude $\mathbf{R}$. This error is then expressed using the angle-axis representation as a vector $\mathbf{e}_\theta$ to avoid the singularity issue. The control torque $\boldsymbol{\tau}$ is then calculated as [79]:

$$\boldsymbol{\tau} = k_p\mathbf{e}_\theta + k_d\left(\boldsymbol{\omega}_{\mathrm{d}} - \boldsymbol{\omega}_b\right) \tag{40}$$

where k_p and k_d are control gains and $\boldsymbol{\omega_d}$ is the desired angular rate, calculated by taking the numerical derivative of the desired attitude.

To obtain the motor commands $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ from the control torque τ and desired thrust T_d, we deploy the following optimization strategy.

$$\begin{aligned}\mathbf{u} =& \underset{\mathbf{u}}{\mathrm{argmin}}\,|T_d - T| \\ &\text{subject to } u_i \geq 0 \text{ and } \boldsymbol{\tau} = \mathbf{A}_{f,r}\mathbf{u}.\end{aligned} \tag{41}$$

Here, the matrix $\mathbf{A}_{f,r} = \{\mathbf{A}_f, \mathbf{A}_r\}$ (for forward and reverse thrust) maps the motor commands to torque.

Unlike the use of linear mapping and matrix inverse as commonly deployed in quadrotors, the method in (41) is required in this work as the desired thrust value in the aerial phase, particularly in the unpowered phase, is oftentimes small or zero. Computing the motor commands directly from deterministically solving the linear map would result in negative values of u_i's. In such cases, the proposed method would allow the robot to generate thrust slightly higher than the target in order to ensure the motor commands remain positive and the desired control torque is produced.

VII. Experimental Results

To evaluate the performance of the robot, the hopping strategy, and validate the payload carrying ability, we conducted several indoor and outdoor experiments. First, in order to compare the effect of payload weight on the robot's ability to track the trajectory, we evaluate the trajectory tracking performance of the robot with variable payload up to 2 kg. Second, we investigate the hopping maneuver with various payload weights. Third, to demonstrate the agility of the hopping robot while carrying a heavy load, we conducted a 90 degree sharp turn experiment, with a 500 gram payload. Finally, we equipped the robot with an onboard computer and LiDAR sensor as functional payload to perform outdoor hopping experiments, testing the robot both with and without autonomous navigation to showcase its potential for operating in and exploring unknown environments.

A. Trajectory Tracking

To validate the hopping performance of the robot and its payload carrying capacity, we conducted the trajectory tracking experiments with payload mass: no payload, 0.5 kg, 1.0 kg, 1.5 kg, and 2.0 kg (Video S2).

The reference trajectory was designed as a figure-8 path, confined within a 2.2×2.2 m area, as illustrated in Fig. 8A. The desired hopping height h_{d} was set to a constant 0.55 m. For the high-level position controller, the desired landing points for each hopping step (indicated by index k) were set to

$$\mathbf{p}_{\mathrm{d}}|_k = \left[1.1\cos\left(2\pi\frac{k}{80}\right), 1.1\sin\left(4\pi\frac{k}{80}\right), 0\right]^T, \tag{42}$$

for $k = 0, 1, 2, \cdots, 80$.

The position and velocity feedback at the apex ($\mathbf{p}(t_{\mathrm{AP}})$, $\dot{\mathbf{p}}(t_{\mathrm{AP}})$) was acquired through the motion capture system (Optitrack Prime 13w) at a frequency of 100 Hz. Additionally, to prevent IMU estimation drift during the robot's free fall (unpowered aerial phase) due to near-zero accelerometer readings [24], attitude measurements were transmitted to the onboard attitude estimator at a frequency of 10 Hz. The complete stance phase model described in Section V was employed to predict the landing attitude $\mathbf{z}_b\left(t_{\mathrm{LD}}\right)$ for each hopping step.

Without any payload, the robot only used powered ascent with a thrust command of $T_f = 0.53m_r g = 116$ gf, with an averaged powered ascending duration of $\Delta t_{\mathrm{PA}} = 0.15$ s

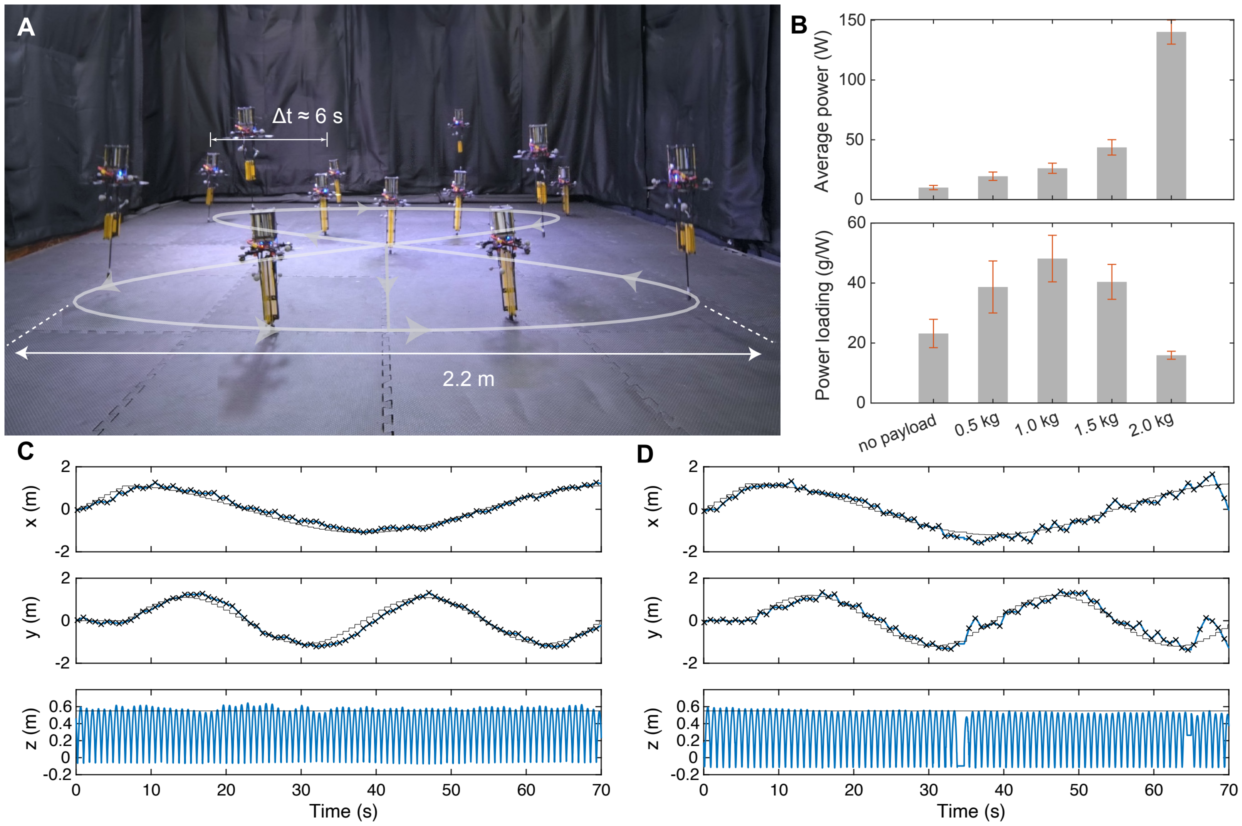


Fig. 8. Trajectory tracking experiments with different payload masses. (A) Composite photo capturing the robot's trajectory while hopping with a 2 kg payload. (B) Average power consumption and power loading when carrying different payloads. Error bars indicate one standard deviation. (C) and (D) Trajectory tracking results, without payload (C) and with 2 kg payload (D). The cross markers are the landing positions. The black lines indicate the landing setpoints (next hop). The blue lines indicate the realized trajectories.

as given by the height controller. The tracking results in Fig. 8C show the robot closely following the prescribed trajectory, with the root mean square errors of 0.17 m for the landing position and 0.036 m for the hopping height.

Furthermore, we estimate the power consumption during the maneuvers based on the thrust commands, exploiting the empirical model of propulsion power (Supplementary Materials). Including the power expended by the flight avionics of approximately 0.5 W, we yield an average power consumption of 10.6 W or 7.1 J per hop. The corresponding power loading is 23.1 g/W. Over the 69 s trajectory, the robot traveled 11.47 m, with an average lateral speed of 0.17 m/s.

With the increased payload weight (0.5 kg, 1.0 kg, and 1.5 kg) the powered ascent thrust command was set to $0.99m_rg$, $1.39m_rg$, and $2.09m_rg$, respectively. Based on the average duration of powered ascent, the robot consumed, on average, 19.5, 26.1, 43.6 W, or 13.0 to 17.5 and 29.2 J of energy per step. The average power loading of 38.6, 48.1, and 40.3 are higher than the no payload trajectory.

When carrying a 2 kg payload, nearer to the theoretical payload limit of 2.22 as computed by (2), the robot took advantage of the bidirectional thrusters to realize powered descent. The tuning parameter for the powered descent in (27) was set to $k_h = 0.8$. The average powered ascent and descent durations are 0.27 and 0.19 s. Thanks to the use of power descent, the power loading of 15.9 is approximately half of those of previous experiments accomplished with only powered ascent (see Fig. 8B) As observed in Fig. 8D, the tracking performance slightly deteriorated when compared with the no payload case. This is reflected by the slightly increased RMSEs of 0.36 m for the landing position and similar RMSEs of 0.019 m for hopping height.

B. Leaping with Payload

As indicated by the complete stance phase model in (14), the influence of the term $mg\mathbf{e}_3$, corresponding to the total weight of the platform, on the hopping dynamics becomes increasingly important when the directions of the leg vector (denoted by $\mathbf{p}_s/\|\mathbf{p}_s\|$) and the gravity ($\mathbf{e}_3$) are misaligned. This occurs in more aggressive jumps with larger steps when the robot's landing attitude $\mathbf{z}_b$ deviates significantly from the vertical. Therefore, the inverted complete stance phase model, compressed into the NN controller, should allow the robot to perform large jumps more accurately than the simplified linear model in (16) or an incomplete model used in [24] as those models neglect the impact of the term $mg\mathbf{e}_3$ for simplification.

Thus, to showcase the advantage of the NN-based controller, exploiting the complete stance phase model, we designed a trajectory with a large step and compared hopping performance with that obtained using a controller with the linear model (Fig. 9A and Video S3).

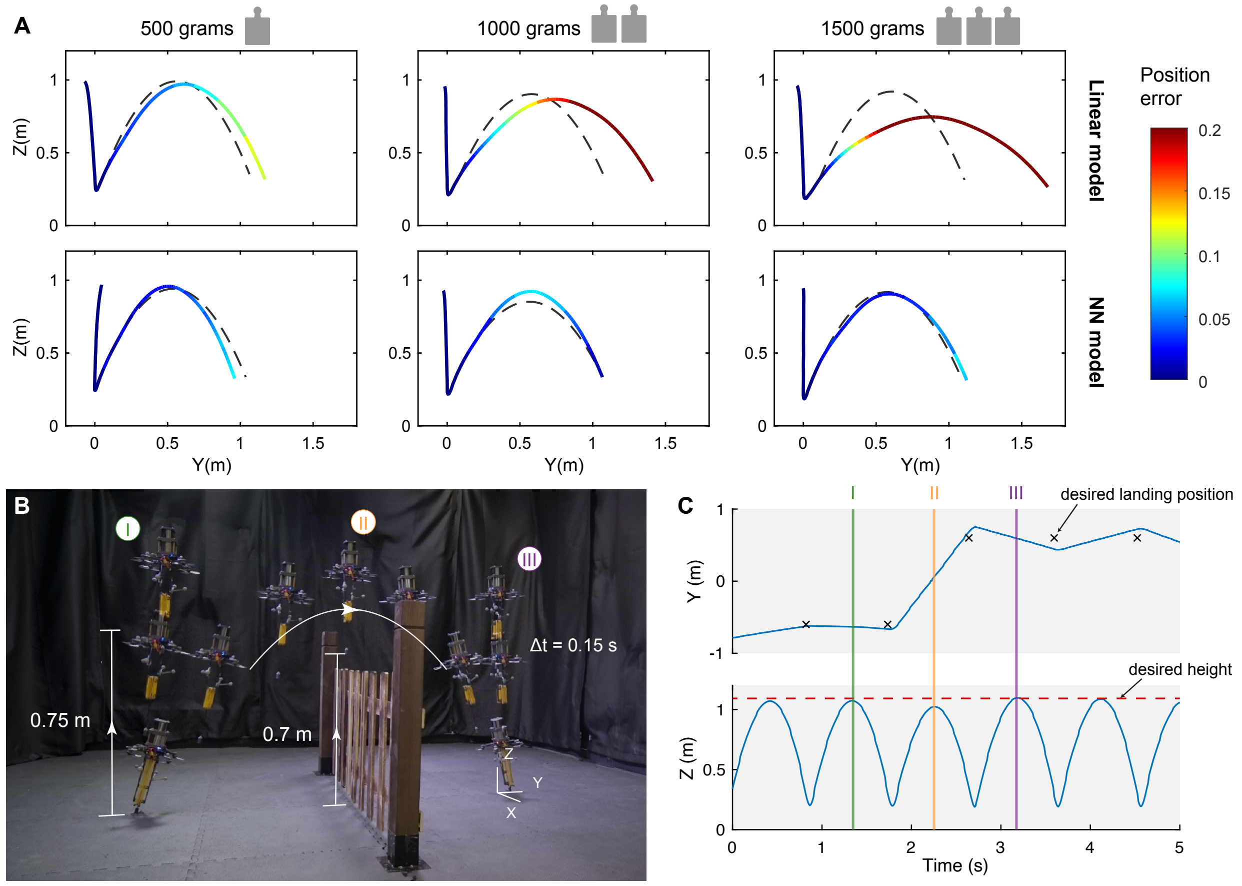


Fig. 9. Leaping with heavy payloads. (A) Leaping performance with three payload masses. The realized trajectories contrast the results obtained from the controllers based on the linear model (first row) and the NN-based model (second row). The black dash lines indicate the desired ballistic trajectory. The colored lines represent the actual trajectory. The color shows the position error compared with the reference. (B) Composite photo showing the trajectory of the robot leaping forward to overcome a 0.6-m fence with 1.5-kg payload. (C) Trajectory data of the leaping-over-the-fence experiment.

In this experiment, the hopper was commanded to perform a single hop with a step length of 1.0 m at a 0.65 m hopping height. To accomplish this, the lateral speed at the takeoff must reach 1.4 m/s. This was completed when the robot carried a 0.5 kg payload. The experiment was repeated with 1.0 kg and 1.5 kg payloads. Except for the stance phase model, other implementations and parameters for the two control methods were identical, including the high-level position controller, height controller, and low-level attitude controller.

The resultant trajectories are plotted in Fig. 9. For the lighter payload (0.5 kg), the difference between the two control methods is less clear. The position errors of the landing point are 0.07 m and 0.11 m for the NN-based model and the linear model. However, as the payload mass increases, the weight plays a more important role in the passive stance phase. Neglecting the robot's weight caused the hopper to overshoot the landing target. For the 1.5 kg payload, the robot landed 0.58 m away from the setpoint for the linear model-based controller whereas the error is only 0.07 m for the NN-based method. In addition, for the NN-based controller, the performance was not visibly affected by the variation in payload mass. The results corroborate the superior performance of the NN-based controller for payload carrying.

Leaping forward gives the hopping robot the ability to overcome obstacles. To showcase this advantage, we conducted an additional experiment where a 0.6-m-high fence was placed in the robot's path (Fig. 9B). The hopping robot, carrying a 1.5-kg payload, was controlled by a controller with a complete stance phase model. With a hopping height setpoint of 0.75 m, the robot successfully cleared the fence, leaping forward 1.2 m with an apex height of 0.7 m as plotted in Fig. 9C. Unlike wheeled or quadrupedal robots, which would require complex climbing mechanisms or ramps to navigate such obstacles, our hopping robot achieves this in a single, dynamic maneuver, demonstrating a clear advantage in traversing challenging terrain with heavy payloads.

C. Sharp Turn with Large Steps and Payload

To further demonstrate the ability to hop with large steps while carrying a heavy payload, we examined a turning maneuver with a 1.1-kg payload (Fig. 10 and Video S4). Using the ground reaction force for hopping, the robot is able to impulsively generate large horizontal acceleration for turning, exceeding what it could achieve when flying. This permits the robot to perform a 90 degree turn between two large jumps as demonstrated in Fig. 10D.

Here, the robot started the maneuver by jumping forward with a step size of 1.0 m and a height of 0.65 m. The

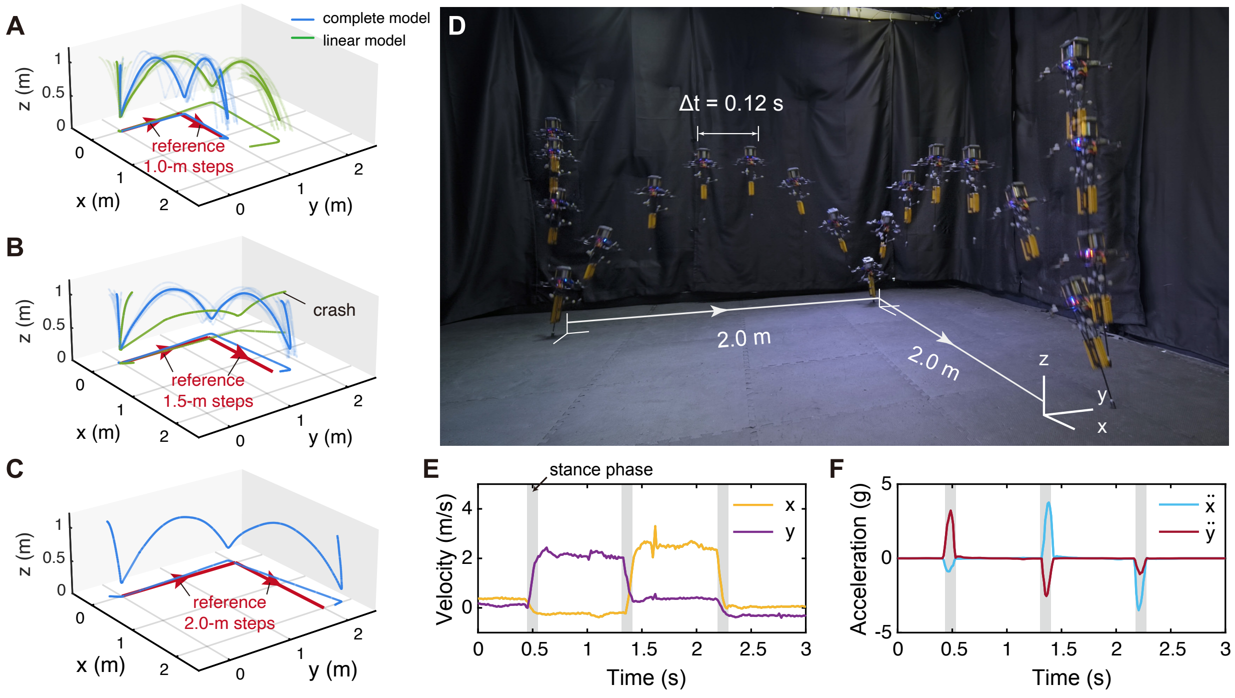


Fig. 10. Sharp turn with large steps and 1.1 kg payload. (A) The sharp turn trajectories with 1.0 m steps. The dark lines represent the trajectory averaged over 10 repeated experiments (light lines show individual trajectories). (B) The sharp turn trajectories with 1.5 m steps. The test with the NN-based controller was repeated 10 times. With the linear model-based controller, the robot crashed and was unable to complete the trajectory. (C) The trajectory of the sharp turn test with 2.0 m steps using the NN-based controller. (D) Composite photo of the experiment in (C). The horizontal speeds and accelerations from the experiment in (C) and (D) are plotted in (E) and (F).

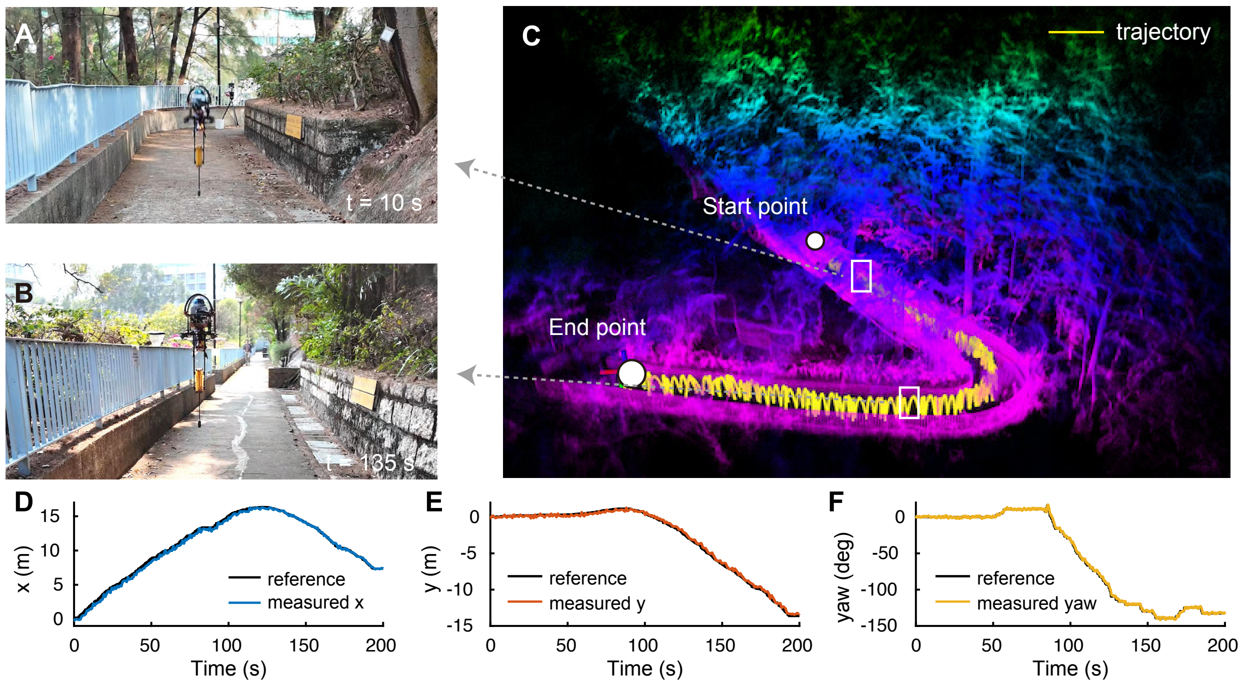


Fig. 11. Hill-side hopping with LiDAR feedback and a human pilot. (A) Photo of the robot at $t = 10$ s. (B) Photo of the robot after hopping for 135 s. (C) The point cloud map constructed by the hopping robot. The yellow line indicates the hopping trajectory. (D)-(E) Reference (from the human operator) and the actual position of the robot during the 200 s field experiment. (F) Reference and actual heading angle of the robot taken from the experiment.

robot's landing attitude setpoint was computed such that the next landing position is 1.0 m away with a 90 degree turn. After that, the previous setpoint was repeated, requiring the robot to rapidly decelerate and suddenly come to a stop by hopping in place. The test was repeated 10 times, with the two control methods (based on NN model and linear model). The realized trajectories are shown in Fig 10A. For the NN-based controller, the robots landed accurately at the setpoints,

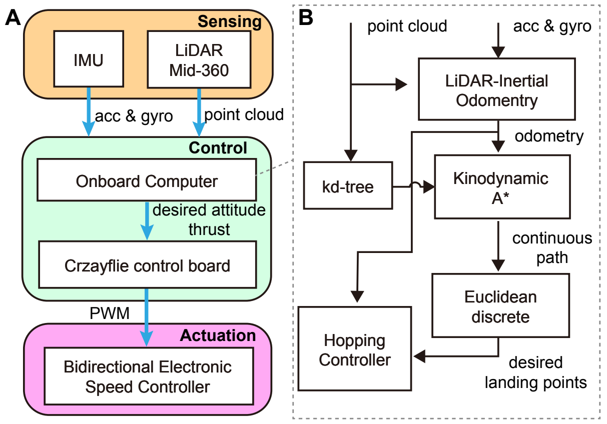


Fig. 12. Architecture of the autonomous navigation strategy. (A) Control architecture devised to realize autonomous hopping. (B) A block diagram illustrating the measurements and signals between the onboard sensors and computers.

with the RMS position errors of 0.13 m, 0.11 m, and 0.12 m for the three landing setpoints. While for the linear model-based controller, the RMSEs were larger for all setpoints: 0.20 m, 0.56 m, and 0.76 m. When the step size was increased to 1.5 m, the NN-based controller successfully completed the turning maneuver (Fig. 10B), achieving RMSEs of 0.21 m, 0.10 m, and 0.15 m. In contrast, the robot controlled by the linear model crashed after a jump. This outcome is consistent with the previous observations from Fig. 9A, in which the simplification of the stance phase model undesirably brought about landing position errors when the step size is large and the payload mass is substantial. This improvement in accuracy came with a modest increase in computational cost: the NN-based controller required 1.5 ms per control loop, compared to 1.3 ms for the linear model (implemented on a laptop with an Apple M2 chip).

To further assess the performance, we conducted an additional single-trial test with the step size increased to 2.0 m. Using the NN-based controller, the robot successfully completed the maneuver while carrying a 1.1 kg payload, achieving position errors of 0.02 m, 0.04 m, and 0.13 m for three landing setpoints. In the experiment, the entire trajectory took 1.82 s, counting from the first to the third jump, resulting in an average horizontal speed of 2.20 m/s as illustrated by Fig. 10E. The sharp turn was realized by the acceleration created by the second hop, leveraging the interaction between the elastic leg and the ground. The acceleration of $4.4g$ during this 0.11 s stance duration (Fig. 10F) dramatically exceeds what the robot could produce during flight, given its limited thrust-to-weight ratio of only 0.33 (considering the total weight of 1.32 kg), markedly lower than unity.

D. 3D Mapping for Wild Environment

With the ability to carry a heavy payload, the robot is able to carry a sensing and computing module to autonomously operate outside the laboratory, eliminating the need for feedback from the motion capture system. This was previously infeasible for the 35 g hopper in [24] due to its limited payload. Here, the robot was equipped with a 360-degree LiDAR (Livox Mid-360, 256 grams), an onboard computer (NUC with CPU i3-N305, weighs 220 grams), and an auxiliary power module (a 4S 850 mAh Li-ion battery and a voltage regulator, 115 g) as payload as shown in Fig. 2A. This allows the robot to perform onboard LiDAR-inertial odometry for state estimation, eliminating the need for feedback from the motion capture system for stabilization. In addition, we integrated an IMU (Bosch Sensortec, BMI088) with an extended acceleration range (up to ± 24 g) to substitute the built-in IMU of Livox Mid-360. This resolves the IMU saturation issue caused by the large ground impact (acceleration over 15 g). The extrinsic parameters between LiDAR and IMU were calibrated [80], and the homogeneous transformation matrix was derived for the LiDAR-inertial odometry [81], [82]. The total weight became 0.95 kg, implying a payload weight of 730 g (payload ratio of $m_p/m_r = 3.3$), including supporting structures and cables.

With the estimates of apex position $\mathbf{p}(t_{\text{AP}})$ and velocity $\dot{\mathbf{p}}(t_{\text{AP}})$ required for the landing state prediction in (30)-(31), the robot is able to hop stably outside the laboratory. To validate the effectiveness of the onboard LiDAR odometry, we conducted the outdoor hopping experiments along a hillside trail (Fig. 11A-B and Video S5). During the experiment, the desired position and heading of the hopping robot were manually given by a human pilot through a joystick as plotted in Fig. 11D-F. The reference hopping height was set to 0.7 m. While generating the point cloud map in Fig. 11C, the robot traveled 30 m, completing over 248 steps in 200 s. This averages to a speed of 0.15 m/s at a frequency of 1.25 Hz. Throughout, the robot closely followed the reference trajectory with the landing position RMSE of 0.40 m, demonstrating similar performance to the indoor trajectory tracking with a payload. The integration of the LiDAR system to enable outdoor operations here substitutes the use of stabilizing aerodynamic dampers proposed in [24]. This offers a solution for heavier hopping quadcopters as the stabilizing effect of air dampers becomes insufficient for larger robots due to the scaling issue nature of surface area-to-mass ratio.

In addition to providing the position and velocity for stabilization, the LiDAR also generated a 3D point cloud map in Fig. 11C. The map details the surroundings and the trajectory of the robot. This shows the potential use of the hopping robot for mapping applications in less structured environments.

E. Autonomous Operation in Outdoor Environment

Thanks to the point cloud map generated by the LiDAR system, the robot could leverage this sensing capability to hop and navigate autonomously, avoiding collisions, without directly relying on a human operator.

To enable autonomous navigation, we adopt the sensing and control structure in Fig. 13A. The implementation involves three subsystems: sensing, control, and actuation.

As part of the control subsystem, the autonomous navigation is enabled by the planning algorithm, implemented on the onboard computer as outlined in Fig. 13B. The point cloud data from the 3D LiDAR is split into two streams for processing. One stream is input to the LiDAR-inertial odometry

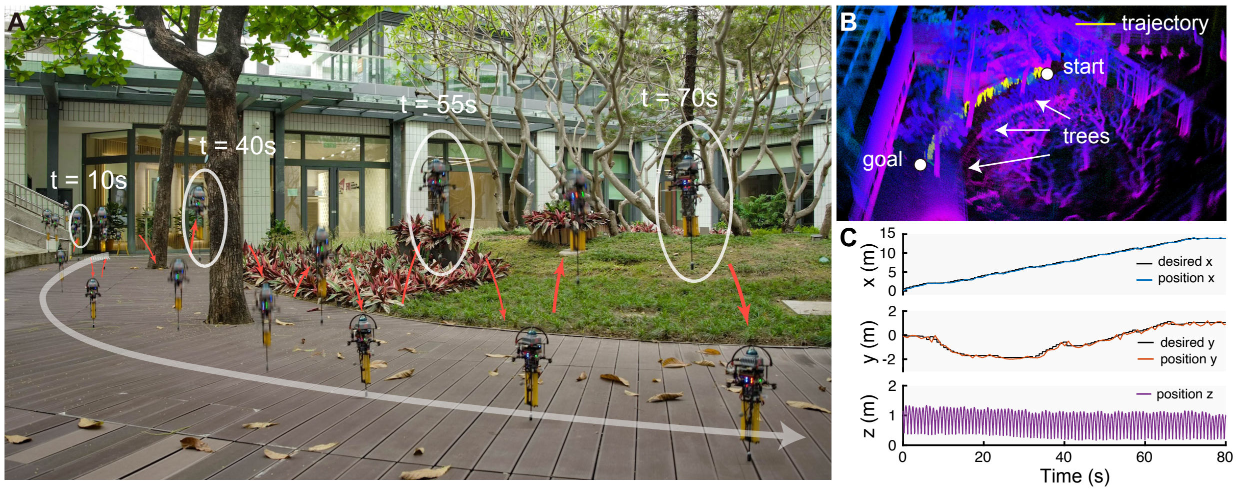


Fig. 13. Outdoor autonomous operation of the robotic hopper with real-time feedback from the LiDAR system. (A) A composite image of the robot navigating to the target location while avoiding collisions with trees. (B) A point cloud map constructed in real time, with the starting point, goal, and the trajectory shown. (C) The planned and actual trajectories obtained from the autonomous hopping experiment.

for estimating the robot's attitude and position, incorporating measurements of acceleration and angular velocity data from the IMU as carried out previously. The second stream provides input for trajectory planning and collision avoidance.

For trajectory planning, we employ a modified version of the kinodynamic A* algorithm, adapted from methods used in aerial robotics [15], [83]. This modification plans a safe, dynamically feasible path in 3D space while accommodating the planar characteristics of our high-level position controller. Specifically, the start and goal positions used for path searching are at the desired hopping height h_d. In between, the algorithm searches for a safe path at an altitude near h_d. This is by limiting the vertical acceleration of the generated path to a narrow range (± 0.1 m/s^2). However, only the horizontal component of the generated trajectory is used by the hopper as $\mathbf{p}_d|_{k+1}$, whereas h_d is kept constant. This ensures the planner considers obstacles in three dimensions along the trajectory.

To enhance the efficiency of safety checks, the point cloud data are organized into a k-d tree structure [15], [84]. The planned path is then projected onto the ground and discretized into several landing points, maintaining a constant Euclidean distance between each point. These discrete landing points serve as the desired values $\mathbf{p}_\mathrm{d}(t_\mathrm{LD})|_k$ for each hopping cycle, which are input into the onboard hopping position controller. The hopping altitude setpoint is separately set to h_d, regardless of the generated trajectory. To account for dynamic obstacles and changes in the environment during exploration, the trajectory is re-planned every five hopping steps by rerunning the planning algorithm. This adaptive approach ensures the robot can navigate safely through changing environments while maintaining its efficient hopping gait.

To validate the autonomous navigation ability and the effectiveness of the method, we conducted the experiment along a pedestrian path surrounded by trees as seen in Fig. 13C. The robot had no prior knowledge of the environment, except for the goal, located 15 m ahead and 1 m to the left of the starting point. Between them, there were several trees between the direct line of sight. The hopping height was set to a constant value of 0.7 m. The trajectory was computed in real-time by the onboard planner and updated every 5 hopping steps. The reference landing position was continuously updated such that the step size was 0.2 m.

The robot traveled over 15 m in 73 s (average speed: 0.2 m/s) without colliding with any obstacles (Video S6). During this period, the robot localized itself with the LiDAR-generated map shown in Fig. 13D. The generated trajectory was closely followed thanks to the high-level position controller. The landing position RMSE was only 0.25 m, with the tracking results plotted in Fig. 13E. The outcome verifies the feasibility of LiDAR-based autonomous navigation for the hopping robot, thanks to its substantial payload capacity.

VIII. Conclusion and Discussion

In this work, we significantly enhance payload capacity of thrust-based hoppers. This improvement is achieved through the integration of bidirectional thrusters and a refined stance phase model that accounts for gravitational effects during the stance phase. The bidirectional thrusters enable the robot to handle increased energy losses from ground impacts and the leg mechanism, accommodating heavier payloads. By considering gravitational forces, the robot achieves greater accuracy in hopping, especially during larger jumps with substantial payloads. Our analysis demonstrates a marked increase in payload capacity. The robot can carry loads up to 9.1 times (2.0 kg) its own mass (220 g) while maintaining stable hopping behavior. This load-carrying capability surpasses other mobile robots across size scales.

The enhanced dynamic model, coupled with a neural network-based control strategy, enables robust hopping performance across various tasks. Exploiting the increased payload capacity, we integrated a LiDAR and computer, demonstrating the robot's potential for autonomous navigation. This capability to hop with a thrust-to-weight ratio below one suggests promising applications for insect-scale aerial robots [85]–[87], which often face challenges in carrying sensors and batteries for untethered operations [17], [88].

Presently, the high-payload hopper displays impressive power loading compared to aerial robots at similar scales [19], indicating that hopping in place is an efficient strategy for loitering at low altitudes. The cost of transport of our robot remains high (over 30) as the demonstrated maneuvers are relatively slow. To enhance transport efficiency, realizing high-speed hopping remains a promising avenue for future research.

Another prospective direction involves developing a footpad to enable the robot to navigate or hold on to rough terrain [89], [90], which could be invaluable in search and rescue missions. Compared to a pointy foot, footpads reduce impact pressure, enabling hopping on soft surfaces or granular media [91]. While footpads may introduce additional energy losses, they offer a strategic trade-off between efficiency and versatility.

REFERENCES

[1] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 699–706, 2023.

[2] H. Jia, S. Bai, and P. Chirarattananon, "Aerial manipulation via modular quadrotors with passively foldable airframes," *IEEE/ASME Transactions on Mechatronics*, 2023.

[3] E. Aucone, C. Geckeler, D. Morra, L. Pallottino, and S. Mintchev, "Synergistic morphology and feedback control for traversal of unknown compliant obstacles with aerial robots," *Nature Communications*, vol. 15, no. 1, p. 2646, 2024.

[4] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, p. eaaw9710, 2019.

[5] N. Chen, F. Kong, W. Xu, Y. Cai, H. Li, D. He, Y. Qin, and F. Zhang, "A self-rotating, single-actuated uav with extended sensor field of view for autonomous navigation," *Science Robotics*, vol. 8, no. 76, p. eade4538, 2023.

[6] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[7] P. Arm, G. Waibel, J. Preisig, T. Tuna, R. Zhou, V. Bickel, G. Ligeza, T. Miki, F. Kehl, H. Kolvenbach *et al.*, "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Science robotics*, vol. 8, no. 80, p. eade9548, 2023.

[8] M. Chiou, G.-T. Epsimos, G. Nikolaou, P. Pappas, G. Petousakis, S. Mühl, and R. Stolkin, "Robot-assisted nuclear disaster response: Report and insights from a field exercise," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4545–4552.

[9] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[10] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[11] M. Sombolestan and Q. Nguyen, "Adaptive-force-based control of dynamic legged locomotion over uneven terrain," *IEEE Transactions on Robotics*, vol. 40, pp. 2462–2477, 2024.

[12] G. Valsecchi, N. Rudin, L. Nachtigall, K. Mayer, F. Tischhauser, and M. Hutter, "Barry: a high-payload and agile quadruped robot," *IEEE Robotics and Automation Letters*, 2023.

[13] B. Jin, S. Ye, J. Su, and J. Luo, "Unknown payload adaptive control for quadruped locomotion with proprioceptive linear legs," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 4, pp. 1891–1899, 2022.

[14] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, p. eadg5014, 2023.

[15] F. Kong, W. Xu, Y. Cai, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.

[16] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[17] Z. Tu, F. Fei, and X. Deng, "Untethered flight of an at-scale dual-motor hummingbird robot with bio-inspired decoupled wings," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4194–4201, 2020.

[18] R. Zufferey, J. Tormo-Barbero, M. M. Guzmán, F. J. Maldonado, E. Sanchez-Laulhe, P. Grau, M. Pérez, J. Á. Acosta, and A. Ollero, "Design of the high-payload flapping wing robot e-flap," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3097–3104, 2021.

[19] S. Bai, Q. He, and P. Chirarattananon, "A bioinspired revolving-wing drone with passive attitude stability and efficient hovering flight," *Science Robotics*, vol. 7, no. 66, p. eabg5913, 2022.

[20] T. Ching, J. Z. W. Lee, S. K. H. Win, L. S. T. Win, D. Sufiyan, C. P. X. Lim, N. Nagaraju, Y.-C. Toh, S. Foong, and M. Hashimoto, "Crawling, climbing, perching, and flying by fiba soft robots," *Science Robotics*, vol. 9, no. 92, p. eadk4533, 2024.

[21] K. Kim, P. Spieler, E.-S. Lupu, A. Ramezani, and S.-J. Chung, "A bipedal walking robot that can fly, slackline, and skateboard," *Science Robotics*, vol. 6, no. 59, p. eabf8136, 2021.

[22] Y. Li, Y. Zhou, J. Huang, Z. Wang, S. Zhu, K. Wu, L. Zheng, J. Luo, R. Cao, Y. Zhang *et al.*, "Jet-hr2: A flying bipedal robot based on thrust vector control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4590–4597, 2022.

[23] B. Zhu, J. Xu, A. Charway, and D. Saldaña, "Pogodrone: Design, model, and control of a jumping quadrotor," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2031–2037.

[24] S. Bai, Q. Pan, R. Ding, H. Jia, Z. Yang, and P. Chirarattananon, "An agile monopedal hopping quadcopter with synergistic hybrid locomotion," *Science Robotics*, vol. 9, no. 89, p. eadi8912, 2024.

[25] Y. Wang, J. Kang, Z. Chen, and X. Xiong, "Terrestrial locomotion of pogox: From hardware design to energy shaping and step-to-step dynamics based control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 3419–3425.

[26] S. Burns and M. Woodward, "Design and control of a high-performance hopping robot," 2024. [Online]. Available: https://arxiv.org/abs/2312.08301

[27] M. Kovač, M. Schlegel, J.-C. Zufferey, and D. Floreano, "Steerable miniature jumping robot," *Autonomous Robots*, vol. 28, pp. 295–306, 2010.

[28] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation," *Science Robotics*, vol. 1, no. 1, p. eaag2048, 2016.

[29] M. Bolignari, A. Mo, M. Fontana, and A. Badri-Spröwitz, "Diaphragm ankle actuation for efficient series elastic legged robot hopping," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4279–4286.

[30] J. An, T. Y. Chung, C. H. D. Lo, C. Ma, X. Chu, and K. W. Samuel Au, "Development of a bipedal hopping robot with morphable inertial tail for agile locomotion," in *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, 2020, pp. 132–139.

[31] C. Kim, D. J. Lee, S. P. Jung, and G. P. Jung, "Dipo: a miniaturized hopping robot via lightweight and compact actuator design for power amplification," *Bioinspiration & Biomimetics*, vol. 18, no. 4, 2023.

[32] J. K. Yim and R. S. Fearing, "Precision jumping limits from flight-phase control in salto-1p," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 2229–2236.

[33] J. K. Yim, E. K. Wang, and R. S. Fearing, "Drift-free roll and pitch estimation for high-acceleration hopping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8986–8992.

[34] W. D. Shin, W. Stewart, M. A. Estrada, A. J. Ijspeert, and D. Floreano, "Elastic-actuation mechanism for repetitive hopping based on power modulation and cyclic trajectory generation," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 558–571, 2023.

[35] E. Ambrose and A. D. Ames, "Improved performance on moving-mass hopping robots with parallel elasticity," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2457–2463.

[36] G. Wenger, A. De, and D. E. Koditschek, "Frontal plane stabilization and hopping with a 2dof tail," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 567–573.

[37] U. Saranlı, Ö. Arslan, M. M. Ankaralı, and Ö. Morgül, "Approximate analytic solutions to non-symmetric stance trajectories of the passive spring-loaded inverted pendulum with damping," *Nonlinear Dynamics*, vol. 62, pp. 729–742, 2010.

[38] A. De and D. E. Koditschek, "Parallel composition of templates for tail-energized planar hopping," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4562–4569.

[39] P. Yu, G. Chamitoff, and K. Wong, "Perching upside down with bi-directional thrust quadrotor," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 1697–1703.

[40] W. Jothiraj, C. Miles, E. Bulka, I. Sharf, and M. Nahon, "Enabling bidirectional thrust for aggressive and inverted quadrotor flight," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 534–541.

[41] K. Mao, J. Welde, M. A. Hsieh, and V. Kumar, "Trajectory planning for the bidirectional quadrotor as a differentially flat hybrid system," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1242–1248.

[42] M. Maier, "Bidirectional thrust for multirotor mavs with fixed-pitch propellers," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[43] J. Wehbeh and I. Sharf, "An mpc formulation on $so(3)$ for a quadrotor with bidirectional thrust and nonlinear thrust constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4945–4952, 2022.

[44] A. Tagliabue, Y.-H. Hsiao, U. Fasel, J. N. Kutz, S. L. Brunton, Y. Chen, and J. P. How, "Robust, high-rate trajectory tracking on insect-scale soft-actuated aerial robots with deep-learned tube mpc," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3383–3389.

[45] Y. H. Hsiao and P. Chirarattananon, "Ceiling effects for hybrid aerial–surface locomotion of small rotorcraft," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2316–2327, 2019.

[46] N. Csomay-Shanklin, V. D. Dorobantu, and A. D. Ames, "Nonlinear model predictive control of a 3d hopping robot: Leveraging lie group integrators for dynamically stable behaviors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 106–12 112.

[47] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extreme-acceleration (14-g) spatial jumping with salto-1p," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3345–3351.

[48] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[49] J. Zhang, F. Gao, X. Han, X. Chen, and X. Han, "Trot gait design and cpg method for a quadruped robot," *Journal of Bionic Engineering*, vol. 11, no. 1, pp. 18–25, 2014.

[50] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[51] B. Goldberg, R. Zufferey, N. Doshi, E. F. Helbling, G. Whittredge, M. Kovac, and R. J. Wood, "Power and control autonomy for high-speed locomotion with an insect-scale legged robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 987–993, 2018.

[52] Z. Liu, W. Zhan, X. Liu, Y. Zhu, M. Qi, J. Leng, L. Wei, S. Han, X. Wu, and X. Yan, "A wireless controlled robotic insect with ultrafast untethered running speeds," *Nature Communications*, vol. 15, no. 1, p. 3815, 2024.

[53] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft robotics*, vol. 1, no. 3, pp. 213–223, 2014.

[54] J. Liang, Y. Wu, J. K. Yim, H. Chen, Z. Miao, H. Liu, Y. Liu, Y. Liu, D. Wang, W. Qiu *et al.*, "Electrostatic footpads enable agile insect-scale soft robots with trajectory control," *Science Robotics*, vol. 6, no. 55, p. eabe7906, 2021.

[55] Z. Miao, J. Liang, H. Chen, J. Lu, X. Sun, Y. Liu, F. Tang, and M. Zhang, "Power autonomy and agility control of an untethered insect-scale soft robot," *Soft Robotics*, vol. 10, no. 4, pp. 749–759, 2023.

[56] X. Ji, X. Liu, V. Cacucciolo, M. Imboden, Y. Civet, A. El Haitami, S. Cantin, Y. Perriard, and H. Shea, "An autonomous untethered fast soft robotic insect driven by low-voltage dielectric elastomer actuators," *Science Robotics*, vol. 4, no. 37, p. eaaz6451, 2019.

[57] X. Dong, C. Tang, S. Jiang, Q. Shao, and H. Zhao, "Increasing the payload and terrain adaptivity of an untethered crawling robot via soft-rigid coupled linear actuators," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2405–2412, 2021.

[58] X. Yang, L. Chang, and N. O. Pérez-Arancibia, "An 88-milligram insect-scale autonomous crawling robot driven by a catalytic artificial muscle," *Science Robotics*, vol. 5, no. 45, p. eaba0015, 2020.

[59] H. H. Hariri, G. S. Soh, S. Foong, and K. L. Wood, "A highly manoeuvrable and untethered under-actuated legged piezoelectric miniature robot," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 59247. American Society of Mechanical Engineers, 2019, p. V05BT07A004.

[60] Y. Zhu, M. Qi, Z. Liu, J. Huang, D. Huang, X. Yan, and L. Lin, "A 5-mm untethered crawling robot via self-excited electrostatic vibration," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 719–730, 2022.

[61] L. Orr, B. Stephens, B. B. Kocer, and M. Kovac, "A high payload aerial platform for infrastructure repair and manufacturing," in *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*. IEEE, 2021, pp. 1–6.

[62] D. R. McArthur, A. B. Chowdhury, and D. J. Cappelleri, "Design of the i-boomcopter uav for environmental interaction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5209–5214.

[63] Dji matrice 600. [Online]. Available: https://www.dji.com/hk/support/product/matrice600

[64] Y. Qin, W. Xu, A. Lee, and F. Zhang, "Gemini: A compact yet efficient bi-copter uav for indoor applications," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3213–3220, 2020.

[65] Datasheet crazyflie 2.1. [Online]. Available: https://www.bitcraze.io/documentation/hardware/crazyflie_2_1/crazyflie_2_1-datasheet.pdf

[66] Z. Zhakypov, K. Mori, K. Hosoda, and J. Paik, "Designing minimal and scalable insect-inspired multi-locomotion millirobots," *Nature*, vol. 571, no. 7765, pp. 381–386, 2019.

[67] Y. Zhong, R. Wang, H. Feng, and Y. Chen, "Analysis and research of quadruped robot's legs: A comprehensive review," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, p. 1729881419844148, 2019.

[68] M. Arnold, L. Hildebrandt, K. Janssen, E. Ongan, P. Bürge, Á. G. Gábriel, J. Kennedy, R. Lolla, Q. Oppliger, M. Schaaf *et al.*, "Leva: A high-mobility logistic vehicle with legged suspension," *arXiv preprint arXiv:2503.10028*, 2025.

[69] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[70] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[71] J. E. Seipel and P. Holmes, "Running in three dimensions: Analysis of a point-mass sprung-leg model," *The International Journal of Robotics Research*, vol. 24, no. 8, pp. 657–674, 2005.

[72] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, "A simply stabilized running model," *SIAM review*, vol. 47, no. 3, pp. 519–549, 2005.

[73] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[74] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[75] R. Blickhan and R. Full, "Similarity in multilegged locomotion: bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, pp. 509–517, 1993.

[76] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[77] H. Geyer, A. Seyfarth, and R. Blickhan, "Spring-mass running: simple approximate solution and application to gait stability," *Journal of theoretical biology*, vol. 232, no. 3, pp. 315–328, 2005.

[78] H. Yu, H. Gao, and Z. Deng, "Toward a unified approximate analytical representation for spatially running spring-loaded inverted pendulum model," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 691–698, 2020.

[79] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[80] F. Zhu, Y. Ren, and F. Zhang, "Robust real-time lidar-inertial initialization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3948–3955.

[81] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[82] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[83] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, “Robust and efficient quadrotor trajectory generation for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[84] B. T. Lopez and J. P. How, “Aggressive 3-d collision avoidance for high-speed navigation.” in *ICRA*, 2017, pp. 5759–5765.

[85] S. B. Fuller, “Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 570–577, 2019.

[86] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, “Controlled flight of a biologically inspired, insect-scale robot,” *Science*, vol. 340, no. 6132, pp. 603–607, 2013.

[87] Y. Chen, H. Zhao, J. Mao, P. Chirarattananon, E. F. Helbling, N.-s. P. Hyun, D. R. Clarke, and R. J. Wood, “Controlled flight of a microrobot powered by soft artificial muscles,” *Nature*, vol. 575, no. 7782, pp. 324–329, 2019.

[88] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, “Untethered flight of an insect-sized flapping-wing microscale aerial vehicle,” *Nature*, vol. 570, no. 7762, pp. 491–495, 2019.

[89] M. G. Catalano, M. J. Pollayil, G. Grioli, G. Valsecchi, H. Kolvenbach, M. Hutter, A. Bicchi, and M. Garabini, “Adaptive feet for quadrupedal walkers,” *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 302–316, 2021.

[90] S. J. Wang, D. Kuang, S. D. Lee, R. J. Full, and H. S. Stuart, “Squirrel-inspired tendon-driven passive gripper for agile landing,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4184–4190.

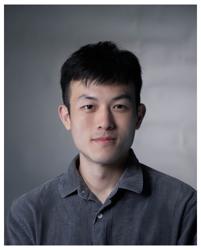
[91] H. Kolvenbach, P. Arm, E. Hampp, A. Dietsche, V. Bickel, B. Sun, C. Meyer, and M. Hutter, “Traversing steep and granular martian analog slopes with a dynamic quadrupedal robot,” *arXiv preprint arXiv:2106.01974*, 2021.

Song Li received the B.S. degree in Flight Vehicle Design from Xi’an Jiaotong University, Shaanxi, China, in 2020, and the M.Eng. degree in Flight Vehicle Design from Beihang University, Beijing, China, in 2023. He is currently pursuing the Ph.D. degree in Biomedical Engineering at City University of Hong Kong, Hong Kong SAR, China.

His research interests include bio-inspired robotics, micro air vehicles, flapping-wing robots, and flapping-wing aerodynamics.

Songnan Bai received a B.S. degree in mechanical engineering from Xi’an Jiaotong University, Xi’an, China, in 2017, an M.Sc. in Mechanical Engineering from City University of Hong Kong, Hong Kong SAR, and a Ph.D. degree in Biomedical Engineering from City University of Hong Kong, Hong Kong SAR.

He is currently a postdoc fellow in the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong SAR.



Ruihan Jia received the B.E. degree in mechatronic engineering from the University of Sydney, NSW, Australia, in 2023. He is currently working toward an M.Phil. degree in Biomedical Engineering at City University of Hong Kong, Hong Kong SAR, China.

His research interests include micro aerial vehicles, dynamics, and robotic perception.

Yixi Cai received a B.E. degree in Automation, in 2020, from the School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing, China.

He is currently a Ph.D. candidate in the Department of Mechanical Engineering, University of Hong Kong (HKU), Hong Kong, China.

Runze Ding received his B.S. degree in aircraft power engineering in 2015 and his M.Phil. degree in aerospace propulsion theory in 2019 from Nanjing University of Aeronautics and Astronautics, China, and a Ph.D. degree in biomedical engineering from City University of Hong Kong, SAR, China, in 2024.

He is currently a postdoctoral researcher with the Department of Biomedical Engineering, City University of Hong Kong, Kowloon, HKSAR, China. His research interests focus on micro aerial robotics.

Yu Shi received the B.E. degree in aerospace engineering, and the Ph.D. degree in navigation, guidance, and control from Beihang University, Beijing, China, in 2018 and 2023, respectively.

His research interests include robust control, networked control, and reinforcement learning.

Fu Zhang received a B.E. degree in automation from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2011, and a Ph.D. degree in controls from the University of California at Berkeley, Berkeley, CA, USA, in 2015.

Dr. Zhang joined the Department of Mechanical Engineering at the University of Hong Kong (HKU), Hong Kong, as an Assistant Professor in August 2018. Dr. Zhang’s current research interests are on robotics and controls, with a focus on unmanned aerial vehicle (UAV) design, navigation, control, and light detection and ranging (LiDAR)-based simultaneous localization and mapping (SLAM).

Pakpong Chirarattananon (S’12-M’15) received a B.A. degree in Natural Sciences from the University of Cambridge, U.K., and a Ph.D. degree in Engineering Sciences from Harvard University, Cambridge, MA, USA.

He is currently an Associate Professor at the Department of Mechanical & Industrial Engineering, Univeristy of Toronto, Canada. Prior to this, he was a faculty member at City University of Hong Kong, Kowloon, Hong Kong SAR, China. His research interests include bio-inspired robots, micro air vehicles, and the applications of control and dynamics in robotic systems.