

Real-Time Trajectory Generation and Hybrid Lyapunov-Based Control for Hopping Robots

Matthew Woodward¹

Abstract—The advent of rotor-based hopping robots has created very capable hopping platforms with high agility and efficiency, and similar controllability, as compared to their purely flying quadrotor counterparts. Advances in robot performance have increased the hopping height to greater than 4 meters and opened up the possibility for more complex aerial trajectories (i.e., behaviors). However, currently hopping robots do not directly control their aerial trajectory or transition to flight, eliminating the efficiency benefits of a hopping system. Here we show a real-time, computationally efficiency, non-linear drag compensated, trajectory generation methodology and accompanying Lyapunov-based controller. The combined system can create and follow complex aerial trajectories from liftoff to touchdown on horizontal and vertical surfaces, while maintaining strict control over the orientation at touchdown. The computational efficiency provides broad applicability across all size scales of hopping robots while maintaining applicability to quadrotors in general.

Index Terms—Hopping, Jumping, Robot, Control, Trajectory Generation

I. INTRODUCTION

HOPPING robots have shown remarkable efficiency as compared to their flying counterparts [1]–[4], however both the newer rotor-based and traditional hopping systems [5], [6] operate in the range of 0.6 to 1.6 meters without significant deviation from a predominantly ballistic trajectory. However, as our previous work on the MultiMo-MHR showed significant increases in hopping performance (> 4 m), the aerial phase has sufficient time and energy to begin, as with aerial systems, controlling the overall trajectory between liftoff (LO) and touchdown (TD), allowing for greater agility and adaptability in unstructured terrain. However, unlike aerial systems, trajectory generation for hopping robots must strictly control the TD states to ensure proper positioning, orientation, and foot-surface contact to avoid damage.

To date there exists five untethered continuous hopping robots including: MultiMo-MHR (our robot) [1], PogoDrone [7], Hopcopter [3], Salto/Salto-1P [8]–[15], and PogoX [4], [16] and one tethered continuous insect-scale hopping robot [17]; where, the hopping controllers focus on foot placement and orientation at TD. This allows for the subsequent LO state to be controlled facilitating control over the horizontal locomotion path and stability of the hopping cycle. Hopcopter and Salto have both explored hopping from vertical surfaces (i.e., walls). However, the vertical surface hopping controllers typically control orientation only. The Hopcopter transitions from horizontal flight control to an orientation hold controller

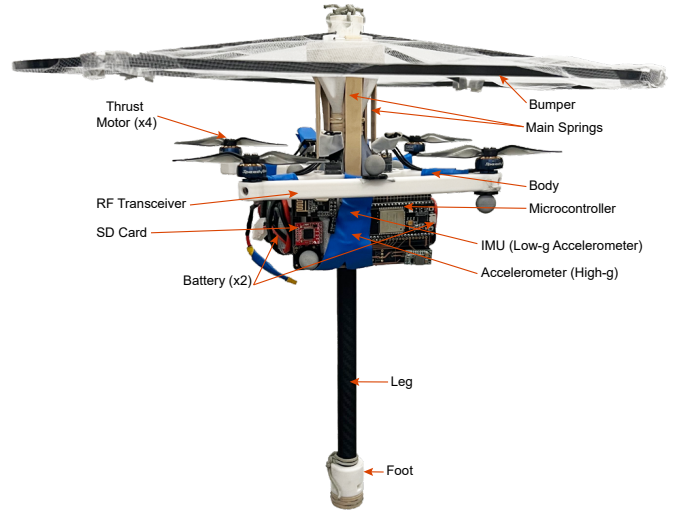


Fig. 1. Photo of the MultiMo-MHR with components labeled.

at 1.8 m from the wall, and the flight controller is reactivated after the wall-hop. Whereas, Salto initiates a wall-hop from a prior ground-hop oriented towards the wall. At LO an orientation hold controller activates to maintain a prescribed wall contact angle, where presumably the foot placement and orientation hold controller would be reactivated; however this is not discussed. In all cases the trajectories are predominantly ballistic however, to accommodate uncertain LO states and desired TD states, interact with both horizontal and vertical surfaces, and avoid obstacles, control over the entire trajectory from LO to TD is necessary to continue advancing the capabilities of rotor-based hopping robots.

The paper is organized as follows, with Section 2 presenting the dynamic model and the differential flatness derivation. Section 3 develops the real-time hopping trajectory generation methodology, and Section 4 derives the Lyapunov-based controller. Section 5 discusses the trajectory tracking results and Section 6 summarizes the work.

II. MODEL

The current generation of rotor-based hopping robots can be model as a quadrotor during the aerial phase of hopping locomotion with world frame basis $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$ and body frame basis $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ both measured in the world frame, and body frame angular velocity ω measured in the body frame (Fig. 1). As with [18], [19], neglecting the rotor dynamics, the

¹The author is with the Robot Locomotion and Biomechanics Laboratory. matthew.woodward@tufts.edu

Newton-Euler equations of motion about the center-of-mass $\mathbf{r}_{cm}$ are as follows,

$$m_r \ddot{\mathbf{r}}_{cm} = -m_r g \mathbf{z}_W + U_1 \mathbf{z}_B - \mathbf{D}_T \quad (1)$$

$$\mathbf{I}_r \dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \mathbf{I}_r \boldsymbol{\omega} + [U_2, U_3, U_4]^T - \mathbf{D}_R \quad (2)$$

where m_r is the robot mass, $\mathbf{I}_r = \text{diag}[I_x, I_y, I_z]$ is the rotational inertia matrix, g is gravity, $\mathbf{D}_T$ is the translational drag force vector, and $\mathbf{D}_R$ is the rotational drag torque vector. The rotors produce both a force $F_{mi} = \zeta_t \Omega_{mi}$ and torque $\tau_{mi} = \zeta_d \Omega_{mi}$ as a function of their angular velocity Ω_{mi} , thrust factor ζ_t , and drag factor ζ_d . Given the rotor configuration in relation the the body frame of the MultiMo-MHR, the inputs are as follows,

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} \zeta_t & \zeta_t & \zeta_t & \zeta_t \\ -\zeta_t L_m & -\zeta_t L_m & \zeta_t L_m & \zeta_t L_m \\ -\zeta_t L_m & \zeta_t L_m & \zeta_t L_m & -\zeta_t L_m \\ \zeta_d & -\zeta_d & \zeta_d & -\zeta_d \end{bmatrix} \begin{bmatrix} \Omega_{m1}^2 \\ \Omega_{m2}^2 \\ \Omega_{m3}^2 \\ \Omega_{m4}^2 \end{bmatrix}$$

where, L_m is the distance from the center of the rotors to the roll and pitch axes. Equations 1 and 2 can be expanded assuming an orientation parameterized by ZYX Euler angles resulting in the acceleration of the robot as,

$$\ddot{x} = m_r^{-1}((\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi)U_1 + D_x) \quad (3)$$

$$\ddot{y} = m_r^{-1}((\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi)U_1 + D_y) \quad (4)$$

$$\ddot{z} = -g + m_r^{-1}(\cos \phi \cos \theta U_1 + D_z) \quad (5)$$

$$\dot{p} = I_x^{-1}(qr(I_y - I_z) + U_2 + D_\phi) \quad (6)$$

$$\dot{q} = I_y^{-1}(pr(I_z - I_x) + U_3 + D_\theta) \quad (7)$$

$$\dot{r} = I_z^{-1}(pq(I_x - I_y) + U_4 + D_\psi) \quad (8)$$

where the non-linear drag forces $\mathbf{D}_T$ and torques $\mathbf{D}_R$ are represented as,

$$\mathbf{D}_T = \text{sign}(\dot{\mathbf{r}}_{cm}) \circ (R_{BW} \mathbf{C}_T) \dot{\mathbf{r}}_{cm}^2 = [D_x, D_y, D_z]^T \quad (9)$$

$$\mathbf{D}_R = \text{sign}(\boldsymbol{\omega}) \circ \mathbf{C}_R \boldsymbol{\omega}^2 = [D_\phi, D_\theta, D_\psi]^T, \quad (10)$$

and $\circ$ is the Hadamard product. This differs from previous works that have linearized about a nominal operating velocity [20], [21], as hopping robots inherently must undergo significant changes in velocity for locomotion; e.g., MultiMo-MHR operates across ± 7 m/s and increases in performance will only increase the linearization error [1]. Therefore, the general non-linear forms are instead used, where, $\mathbf{R}_{BW}$ is the rotation matrix from body to world frame, and $\mathbf{C}_T$ and $\mathbf{C}_R$ are the overall translational and rotational drag coefficient matrices in the body frame, respectively. The drag coefficient matrices include the air density ρ and effective area A , as $\mathbf{C}_{T,R} = 0.5 C_{T,R,i,j} \rho A$. The robot states are therefore defined as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]$.

A. Differential Flatness

From [18], [21], the quadrotor model is seen to be differentially flat for four inputs. Therefore, all states and inputs can be calculated from four specifically selected flat outputs and their derivatives. The selected flat outputs include the center-of-mass position $\mathbf{r}_{cm} = [x, y, z]^T$ and the yaw angle ψ . Therefore, given a desired trajectory in $\nu = [x, y, z, \psi]$, the

desired position, velocity, and acceleration of the 6 degrees-of-freedom including $\mathbf{x}_d = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]$ and $\mathbf{x}_{\ddot{d}} = [\ddot{x}, \ddot{y}, \ddot{z}, \ddot{p}, \ddot{q}, \ddot{r}]$ can be determined.

From [18], the center-of-mass position, velocity, acceleration, and jerk are determined directly from the flat outputs as $[\mathbf{r}_{cm}, \dot{\mathbf{r}}_{cm}, \ddot{\mathbf{r}}_{cm}, \ddot{\mathbf{r}}_{cm}]$ in the world frame, respectively. The acceleration then determines the orientation $\mathbf{R}_{BW} = [\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B]$ by,

$$\begin{aligned} a_{U_1} &= \ddot{\mathbf{r}}_{cm} + g \mathbf{z}_W + \frac{\mathbf{D}_T}{m_r} \\ \mathbf{z}_B &= a_{U_1} / \|a_{U_1}\|, \quad \mathbf{x}_\psi = [\cos \psi, \sin \psi, 0]^T \\ \mathbf{y}_B &= \frac{\mathbf{z}_B \times \mathbf{x}_\psi}{\|\mathbf{z}_B \times \mathbf{x}_\psi\|}, \quad \mathbf{x}_B = \mathbf{z}_B \times \mathbf{y}_B \end{aligned} \quad (11)$$

where $\mathbf{x}_\psi$ is the x-axis of the intermediate frame created by a yaw rotation. Taking the derivative of equation 1 results in,

$$\begin{aligned} m_r \ddot{\mathbf{r}}_{cm} &= \dot{U}_1 \mathbf{z}_B + \mathbf{R}_{BW} \boldsymbol{\omega} \times U_1 \mathbf{z}_B \\ &\quad - \text{sign}(\dot{\mathbf{r}}_{cm}) \circ ((\dot{\mathbf{R}}_{BW} \mathbf{C}_T) \dot{\mathbf{r}}_{cm}^2 + 2(\mathbf{R}_{BW} \mathbf{C}_T) \dot{\mathbf{r}}_{cm} \circ \ddot{\mathbf{r}}_{cm}) \\ &\quad - \delta(\dot{\mathbf{r}}_{cm}) \circ (2(\mathbf{R}_{BW} \mathbf{C}_T) \dot{\mathbf{r}}_{cm}^2 \circ \ddot{\mathbf{r}}_{cm}) \end{aligned} \quad (12)$$

where, the Dirac delta (δ) component will always be zero, and $\dot{\mathbf{R}}_{BW} = \mathbf{R}_{BW} \hat{\boldsymbol{\omega}}$; where, $\hat{\boldsymbol{\omega}}$ is the skew symmetric matrix of $\boldsymbol{\omega}$. Adjusting the center-of-mass jerk ($\ddot{\mathbf{r}}_{cm}$) by the drag and eliminating the Dirac delta components results in,

$$\begin{aligned} m_r \ddot{\mathbf{r}}_{cm}^* &= \dot{U}_1 \mathbf{z}_B + \mathbf{R}_{BW} \boldsymbol{\omega} \times U_1 \mathbf{z}_B \quad \text{where,} \\ \ddot{\mathbf{r}}_{cm}^* &= \ddot{\mathbf{r}}_{cm} + \frac{1}{m_r} \text{sign}(\dot{\mathbf{r}}_{cm}) \circ (((\mathbf{R}_{BW} \hat{\boldsymbol{\omega}}) \mathbf{C}_T) \dot{\mathbf{r}}_{cm}^2 \\ &\quad + 2(\mathbf{R}_{BW} \mathbf{C}_T) \dot{\mathbf{r}}_{cm} \circ \ddot{\mathbf{r}}_{cm}). \end{aligned} \quad (13)$$

Assuming the mass normalized thrust rate of change ($\dot{U}_1/m_r$) is approximately equal to the body z-axis drag adjusted jerk ($\mathbf{z}_B \ddot{\mathbf{r}}_{cm}^*$), $\dot{U}_1 \sim m_r \mathbf{z}_B \ddot{\mathbf{r}}_{cm}^*$, the drag adjusted jerk ($\ddot{\mathbf{r}}_{cm}^*$) then determines the body frame angular velocity as,

$$\mathbf{h}_\omega = \mathbf{R}_{BW} \boldsymbol{\omega} \times \mathbf{z}_B = m_r / U_1 (\ddot{\mathbf{r}}_{cm}^* - (\mathbf{z}_B \ddot{\mathbf{r}}_{cm}^*) \mathbf{z}_B) \quad (14)$$

$$p = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad q = \mathbf{h}_\omega \cdot \mathbf{x}_B, \quad r = \dot{\psi} \mathbf{z}_W \cdot \mathbf{z}_B \quad (15)$$

where equation 14 is the first derivative of equation 1. Equation 15, given the $\mathbf{R}_{BW} \hat{\boldsymbol{\omega}}$, results in three equations and three unknowns that can be solved for desired body rotational velocities $[p, q, r]$. In practice this can be simplified by, assuming a constant symmetric drag coefficient C_T resulting in $\ddot{\mathbf{r}}_{cm}^* = \ddot{\mathbf{r}}_{cm} + \frac{1}{m_r} \text{sign}(\dot{\mathbf{r}}_{cm}) \circ (2C_T \dot{\mathbf{r}}_{cm} \circ \ddot{\mathbf{r}}_{cm})$, or by using the angular velocities from the previous step ω_{k-1} . The angular acceleration and jerk, given the second and third derivatives of equation 1, can then be found in the same manner as angular velocity.

III. HOPPING TRAJECTORY GENERATION

To generate a trajectory, a set of keyframes $\alpha_i(t_i)$ at specific times t_i , is necessary to control the entry conditions, progression through, and exit conditions of the generated trajectory. Previous work in quadrotors has defined the initial and final keyframes $[\alpha_0, \alpha_m]$ as the desired flat outputs ν^T ; where the sequence of keyframes is connected together using piecewise polynomials, with smooth transitions, that represent the trajectory in each of the four flat outputs [18].

The hopping locomotion cycle is naturally segmented into the stance phase (TD to LO) and the aerial phase (LO to TD). However, due to the high impact forces and torques, and short duration of the stance phase, the aerial phase is the predominant phase for control, and will be the focus of the trajectory generation here.

Dividing the aerial phase into keyframes, results in an initial at LO $\alpha_0(t_0)$ and a final at TD $\alpha_m(t_m)$, and the potential for intermediate frames added between. The flat outputs of the keyframes must also be expanded to account for the highly dynamic stance phase and necessity to fully control the TD state to avoid damage and ensure proper subsequent LO state. This is achieved by expanding the keyframes to include the desired flat outputs and the first three derivatives, $\alpha_i = [\nu, \dot{\nu}, \ddot{\nu}, \dddot{\nu}]$ with $i = [0, \dots, m]$. The full state $\mathbf{x}$ can now be controlled by using the linear acceleration to control the roll and pitch $[\phi, \theta]$ (equation 11) and jerk to control the angular velocity $[p, q]$ (equations 14).

Given the independence of the flat outputs, the trajectory generation problem can be divided into four separate problems, where, $\nu_{i,j,k}(t)$ represents the value of the i^{th} keyframe, j^{th} flat output, and k^{th} derivative of the flat output at time t . Then the trajectory polynomial $\nu_{j,k}(t)$ that connects the keyframes of the k^{th} derivative of the j^{th} flat output can be modeled as follows,

$$\begin{aligned} \nu_{j,k}(t) &= \frac{d^k}{dt^k}(f_n(t)) \mathbf{c}_j \\ &= \frac{d^k}{dt^k}([1, t, t^2, \dots, t^n])[a_0, a_1, a_2, \dots, a_n]^T \end{aligned} \quad (16)$$

where, $t_0 \leq t \leq t_m$, and n is the order of the trajectory polynomial. As with [22], to solve for the coefficients $\mathbf{c}_j = [a_0, a_1, a_2, \dots, a_n]^T$, the problem can be setup as a linear algebraic solution $\mathbf{P}_l \mathbf{c}_j = \nu_j$, with full keyframes (includes all three derivatives), results in,

$$\mathbf{P}_l = \begin{bmatrix} f_n(t_0) \\ df_n(t_0)/dt \\ d^2 f_n(t_0)/dt^2 \\ d^3 f_n(t_0)/dt^3 \\ \vdots \\ f_n(t_m) \\ df_n(t_m)/dt \\ d^2 f_n(t_m)/dt^2 \\ d^3 f_n(t_m)/dt^3 \end{bmatrix}, \mathbf{c}_j = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \nu_j = \begin{bmatrix} \nu_{1,j,0}(0) \\ \nu_{1,j,1}(0) \\ \nu_{1,j,2}(0) \\ \nu_{1,j,3}(0) \\ \vdots \\ \nu_{m,j,0}(t_m) \\ \nu_{m,j,1}(t_m) \\ \nu_{m,j,2}(t_m) \\ \nu_{m,j,3}(t_m) \end{bmatrix}$$

(4(m+1) × n+1) (n+1 × 1) (4(m+1) × 1)

where, $\mathbf{P}_l$ represents the polynomial matrix of f_n and its derivatives $d^k/dt^k(f_n)$, and ν_j is the desired flat output vector. The general solution is therefore $\mathbf{c}_j = \mathbf{P}_l^{-1} \nu_j$ where $\mathbf{c}_j = \mathbf{c}_j^* + \mathbf{N}_l(\mathbf{P}_l) \mathbf{c}_{j_N}$, $\mathbf{c}_j^*$ is the least squares fit coefficients, $\mathbf{N}_l(\mathbf{P}_l)$ is the null space matrix of $\mathbf{P}_l$, and $\mathbf{c}_{j_N}$ is the null space coefficient vector. While $\mathbf{c}_j^*$ can be efficiently calculated, previous work has shown the null space coefficients $\mathbf{c}_{j_N}$ must be calculated through general optimization [22] or the overall coefficients $\mathbf{c}_j$ can be optimized through quadratic programming to minimized the square of the forth derivative of position (snap) [18]. Optimization, however, requires time

TABLE I
HOPPING TRAJECTORIES

Traj.	Keyframe (α_0) LO				(α_1)	Keyframe (α_2) TD			
	ν	$\dot{\nu}$	$\ddot{\nu}$	$\ddot{\nu}'$	$\ddot{\nu}$	ν	$\dot{\nu}$	$\ddot{\nu}$	$\ddot{\nu}'$
T1 (LO-TD): Ground-Ground, Wall-Ground									
x, y	x	x	x	x	x	o	x	x	x
z	x	x	x	x	x	x	x	x	x
ψ	x	x	o	o	o	x	x	o	o
T2 (LO-TD): Ground-Wall, Wall-Wall									
x, y	x	x	x	x	x	x	x	x	x
z	x	x	x	x	x	o	x	x	x
ψ	x	x	o	o	o	x	x	o	o
T3 (LO-TD): Ground-State, Wall-State									
x, y	x	x	x	x	x	x	x	x	x
z	x	x	x	x	x	x	x	x	x
ψ	x	x	o	o	o	x	x	o	o

The (x) indicates desired values, and the (o) indicates free values.

TABLE II
HOPPING KEYFRAMES

Traj.	Keyframe (α_0) LO		Keyframe (α_2) TD	
	x, y, z	ψ	x, y, z	ψ
U_{1d}	$U_{1LO} = 0.9m_r g$	NA	$U_{1TD} = 0.2m_r g$	NA
v_{TDd}	NA	NA	DV	NA
$\mathbf{z}_{Bd}$	NA	NA	DV	NA
ν	SE	SE	DV	DV
$\dot{\nu}$	SE	SE	$-v_{TD} \mathbf{z}_{Bd}$	0
$\ddot{\nu}$	$U_{1LO} \mathbf{z}_B - g \mathbf{z}_W$	0	$U_{1TD} \mathbf{z}_{Bd} - g \mathbf{z}_W - \frac{\mathbf{D}_T}{m_r}$	0
$\dddot{\nu}$	$[0, 0, 0]^T$	0	$[0, 0, 0]^T$	0

The (SE) indicates state estimation values, (DV) indicates desired values, and (NA) indicates not applicable. Note: the acceleration of the intermediate keyframe α_1 is equal to the acceleration of α_2 .

and computational power which limits the potential for real time trajectory generation. To allow real-time generation, normalized trajectories can be precalculated and then, in real-time, temporally and spatially scaled [18]. However, as scaling fundamentally changes the derivatives of the flat outputs, the final keyframe will not maintain the desired derivative values. Moreover, because of the desired derivative values and operation about zero thrust, instead of hover in quadrotors, scaling time or space will fundamentally and potentially significantly alter the required trajectory. Therefore, whereas quadrotors may be less concerned with the derivatives of the flat outputs at the final keyframe, hopping robots must maintain the desired values, as they dictate the TD orientation, TD energy, and LO state. Therefore, a computationally efficient real-time trajectory generation methodology for hopping robots will be developed.

To develop a real-time hopping trajectory generation methodology, three keyframes will be used including, an initial at LO $\alpha_0(t_0 = 0)$, an intermediate near TD $\alpha_1(t_1)$, and a final at TD $\alpha_2(t_2)$; where $t_2 = t_m$ is the total time from LO to TD, and the intermediate keyframe α_1 is used to force the robot to the desired TD orientation prior to the TD keyframe α_2 . Therefore, keyframe α_1 will only contain the accelerations and they will be set equal to the desired accelerations in keyframe α_2 . The order n of the trajectory polynomial $f_n(t)$ will be selected as one less than the total number of desired flat outputs across the the three keyframes ν_j ; where removing specific desired flat outputs, allows the value to vary and can reduce

TABLE III
CONTROL EQUATIONS

$$\begin{aligned}
U_1 &= m_r((\ddot{z}_d + \sigma_1) + \dot{e}_z k_{pz}) + \frac{k_{U1} m_r (\dot{e}_z k_{dz} + e_z k_{pz})}{2k_{dz}} \\
U_2 &= \frac{I_x(k_{dy}(\ddot{y}_d + \sigma_2) + \dot{e}_y k_{py} + e_y k_{pz})}{L_t k_{d\phi}} - \frac{I_x(\frac{D\phi}{I_x} - \dot{p}_d + \frac{r q(I_y - I_z)}{I_x})}{L_t} + \frac{I_x k_{U2} (\dot{e}_y k_{dy} + e_y k_{d\phi} + e_\phi k_{py} + e_y k_{pz})}{2L_t k_{d\phi}} \\
U_3 &= \frac{I_y(\dot{e}_x k_{px} + e_x k_{p\theta} - k_{dx}(-\ddot{x}_d + \sigma_3))}{L_t k_{d\theta}} + \frac{I_y(\dot{q}_d - \frac{D\theta}{I_y} + \frac{pr(I_x - I_z)}{I_y})}{L_t} + \frac{I_y k_{U3} (\dot{e}_x k_{dx} + e_x k_{d\theta} + e_\theta k_{px} + e_x k_{pz})}{2L_t k_{d\theta}} \\
U_4 &= I_z \dot{r}_d - D_\psi + \frac{I_z e_r k_{p\psi} + \frac{I_z e_\psi k_{U4} k_{p\psi}}{2}}{k_{d\psi}}; \text{ where, } \sigma_1 = -\frac{D_x(s_\phi s_\psi + c_\phi c_\psi s_\theta)}{m_r} + \frac{D_y(c_\psi s_\phi - c_\phi s_\psi s_\theta)}{m_r} - \frac{c_\phi c_\theta(D_z - g m_r)}{m_r} \\
\sigma_2 &= \frac{D_x(c_\phi s_\psi - c_\psi s_\phi s_\theta)}{m_r} - \frac{D_y(c_\phi c_\psi + s_\phi s_\psi s_\theta)}{m_r} - \frac{c_\theta s_\phi(D_z - g m_r)}{m_r}, \sigma_3 = \frac{D_x c_\psi c_\theta + D_y c_\theta s_\psi - (D_z - g m_r) s_\theta}{m_r}
\end{aligned}$$

Implementation: Set $[-e_y, -\dot{e}_y]$ for proper control direction. The MultiMo-MHR uses the following: $[k_{px}, k_{dx}, k_{py}, k_{dy}, k_{pz}, k_{dz}] = [10, 1, 10, 1, 10, 1]$, $[k_{p\phi}, k_{d\phi}, k_{p\theta}, k_{d\theta}, k_{p\psi}, k_{d\psi}] = [30, 1, 30, 1, 30, 1]$, and $[k_{U1}, k_{U2}, k_{U3}, k_{U4}] = [10, 80, 80, 80]$, with the remaining parameters from previous work [1], [2]. Notation: $s_\gamma = \sin(\gamma)$ and $c_\gamma = \cos(\gamma)$

the aggressiveness of the generated trajectory $\mathbf{c}_j^*$. Therefore, $\mathbf{P}_l$ is square, and the symbolic inverse $\mathbf{P}_l^{-1}(t_m)$ is easily precomputed as a function of both t_1 and t_2 ; which allows for an inherently parallel and therefore efficient calculation of the the least squares coefficients $\mathbf{c}_j^*$.

To add trajectory flexibility, the order n of the trajectory polynomial $f_n(t)$ in $\mathbf{P}_l$ is increased by n^* , adding n^* null space basis vectors to the null space matrix $\mathbf{N}_l(\mathbf{P}_l(f_{n+n^*}))_{(n+n^* \times n^*)}$ which can also be symbolically precomputed as functions of t_1 and t_2 . Therefore, the solution is modified to $\mathbf{c}_j = [(\mathbf{c}_j^*)^T, 0_{(1 \times n^*)}]^T + \mathbf{N}_l(\mathbf{P}_l(f_{n+n^*})) \mathbf{c}_{jN}$, where, $0_{(1 \times n^*)}$ is a zero vector to account for the added polynomial coefficients not included in $\mathbf{c}_j^*$. To analytically solve for the null space coefficients $\mathbf{c}_{jN}$, we will use the superposition principle of the null space. Therefore, substituting the modified solution into equation 16 and solving for the null space coefficients $\mathbf{c}_{jN}$ results in,

$$\mathbf{c}_{jN} = (\mathbf{M}_1^T \mathbf{M}_1)_{(n^* \times n^*)}^{-1} (\mathbf{M}_1^T \mathbf{M}_2)_{(n^* \times n^*)}; \text{ where, } (17)$$

$$\mathbf{M}_1 = \mathbf{P}_N \mathbf{N}_l(\mathbf{P}_l(f_{n+n^*})),$$

$$\mathbf{M}_2 = \nu_{j,k}(t) - \mathbf{P}_N [(\mathbf{c}_j^*)^T, 0_{(1 \times n^*)}]^T,$$

$$\mathbf{P}_{N(s \times n+n^*)} = \frac{d^k}{dt^k}(f_{n+n^*}(t)),$$

$\nu_{j,k}(t)$ is now the vector of the j^{th} desired flat outputs and its k^{th} derivatives at s time points, and $\mathbf{P}_N$ is the polynomial matrix of f_{n+n^*} at s time points. The desired flat outputs can specify any derivative $k \leq n + n^*$ and any number s of time points t such that $t_0 \leq t \leq t_m$ with the exception of the desired points specified in $\mathbf{P}_l$ as the null space will always be zero at those points. As can be seen in equation 17, both $\mathbf{M}_1^T \mathbf{M}_1$ and $\mathbf{M}_1^T \mathbf{M}_2$ are square with dimension $n^* \times n^*$. Therefore, regardless of the number of desired flat outputs $\nu_{j,k}(t)$, calculation of the null space coefficients will always be related to the number of null space basis vectors. Given the number of additional desired flat outputs s along the trajectory is less than or equal to n^* , the solution is guaranteed to satisfy them, whereas, if $s > n^*$ the result will be the least squares solution. However, as the fit is only using the null space, the desired LO, intermediate, and TD keyframe values

will remain unchanged. It is important to note that increasing the polynomial order $n + n^*$ increases trajectory adaptability but also increases the potential state derivative values, and therefore the difficulty in following the trajectory; where, robot characteristics such as thrust-to-weight and torque-to-rotational inertia will determine the upper limit.

A. Hopping Trajectories

Given the structured nature of the hop cycle, it is possible to identify general hopping trajectory types based on the TD surface and characteristics. Table I shows the six general hopping trajectories combined into three unique sets of desired and free flat outputs in the three hopping keyframes. These include those that TD on horizontal surfaces (T1: Ground-Ground, Wall-Ground) with position $[x, y]$ free, those that TD on vertical surfaces (T2: Ground-Wall, Wall-Wall) with position $[z]$ free, and those that TD at a specified state (T3: Ground-State, Wall-State) with none free; where, in all cases $[\dot{\psi}, \ddot{\psi}]$ are free. Table I, shows three unique rows which creates three unique $\mathbf{P}_l$ matrices resulting in three $\mathbf{P}_l^{-1}$ and $\mathbf{N}_l(\mathbf{P}_l(f_{n+n^*}))$ matrices to precompute, as functions of the trajectory time t_m ; including, $\mathbf{P}_0$ (all desired), $\mathbf{P}_1$ (free TD position), $\mathbf{P}_2$ (free acceleration and jerk). Since the null space basis vectors abide by the superposition principle, precomputing the null spaces $\mathbf{N}_l(\mathbf{P}_l(f_{n+n^*}))$ for more null space basis vectors than necessary, allows for any individual or combination of basis vectors to be used for each trajectory generated.

In practice, it is observed that $n^* = 2$, provides good flexibility in adjusting the generated trajectory. To set the LO keyframe α_0 , the position, velocity, and orientation (e.g., Euler angles) must be determined from state estimation along with a desired input U_{1LO} . Given the fast response time of the motors and probable desired for high thrust at LO, the acceleration, jerk, and yaw states are set as seen in Table II. To set the TD keyframe α_2 , the desired position, orientation, velocity magnitude v_{TD} , and input U_{1TD} must be first determined, and the velocity, acceleration, jerk, and yaw states are then set as seen in Table II. The TD thrust U_{1TD} is set to 20% of the body weight to ensure proper orientation for surface contact with the robot's foot, and the velocity vector $-v_{TD} \mathbf{z}_B$ is aligned

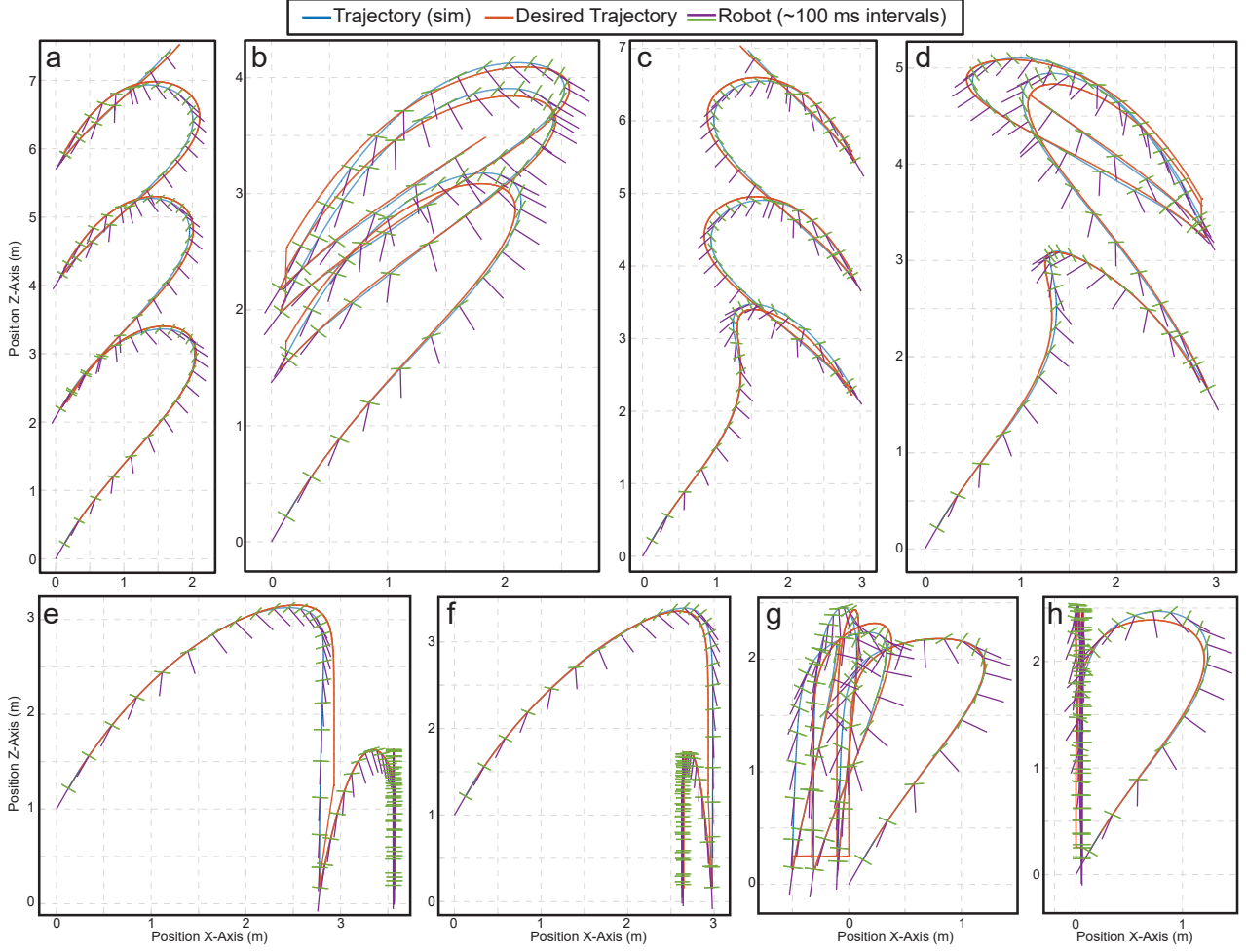


Fig. 2. Hopping trajectory generation and control (Tables II and IV show details) for constant initial conditions (keyframe α_0) including roll, pitch, and yaw of $[\phi, \theta, \psi] = [0, 30, 0]$ degrees, velocity magnitude aligned with the body z-axis of 5 m/s, trajectory time $t_m = 1.75$ seconds (LO-TD), and reorientation time $\delta_t = 0.05$ seconds; where total simulation time equals 6 seconds. The TD keyframe α_2 for all generated trajectories includes $v_{TD_d} = 5$ m/s and the desired orientation $[\phi, \theta, \psi]$ is set as: a,b) $[0, 30, 0]$, c,d) $[0, -30, 0]$, e,f) $[0, 0, 0]$, g,h) $[0, 0, 0]$ degrees.

with the orientation of the robot; i.e., the TD velocity vector is aligned with the body z-axis creating zero moment at TD.

IV. CONTROL

To follow the generated trajectory, a Lyapunov-based controller will now be developed. First the errors $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$ in position and velocity of all 6-degrees-of-freedom (6DOF) are determined, where the translational errors in position and velocity are rotated into the body frame with $\mathbf{R}_{BW}^T$. The errors are then divided into the four control errors as follows,

$$\begin{aligned} e_{U_1} &= k_{p_z} e_z + k_{d_z} \dot{e}_z \\ e_{U_2} &= k_{p_y} e_y + k_{d_y} \dot{e}_y + k_{p_\phi} e_\phi + k_{d_\phi} \dot{e}_\phi \\ e_{U_3} &= k_{p_x} e_x + k_{d_x} \dot{e}_x + k_{p_\theta} e_\theta + k_{d_\theta} \dot{e}_\theta \\ e_{U_4} &= k_{p_\psi} e_\psi + k_{d_\psi} \dot{e}_\psi \end{aligned}$$

where k_p 's ≥ 0 and k_d 's > 0 are the gains. A positive definite Lyapunov candidate function $V(\mathbf{x})$ is then,

$$V(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^4 e_{U_i}^2 \quad (18)$$

where, the derivative of V is set to be negative definite as,

$$\frac{dV}{dt} = -\frac{1}{2} \sum_{i=1}^4 k_{U_i} e_{U_i}^2. \quad (19)$$

The sum in both V and dV/dt allows the terms associated with the four inputs $[U_1, U_2, U_3, U_4]$ to be separated as,

$$\frac{1}{2} \frac{d}{dt} e_{U_i}^2 = -\frac{1}{2} k_{U_i} e_{U_i}^2 \quad (20)$$

Computing the derivative will naturally require derivatives of the 6DOF velocities resulting in acceleration errors in equation 20; e.g., $\ddot{e}_x = \ddot{x}_d - \ddot{x}$. To account for the robot dynamics, the translational accelerations (equations 3-5), rotated into the body frame by $\mathbf{R}_{BW}^T$, and rotational accelerations (equations 6-8) are substituted in for the state accelerations. Table III shows the solutions for the individual inputs U_i .

The controller represents a hybrid Lyapunov-based controller where the aerial phase (LO-TD) evolves continuously with stability guarantees and the stance phase (TD-LO) is represented as a discrete event. To maintain stability over the complete hop cycle requires maintaining stability guarantees

during the stance phase. Therefore, the difference between the candidate functions at TD and the subsequent LO must be as follows, $V(\mathbf{x}_{LO}) - V(\mathbf{x}_{TD}) \leq 0$; where, the LO states $\mathbf{x}_{LO}$ are determined by a function $h(\mathbf{x}_{TD})$ and the TD states as, $\mathbf{x}_{LO} = h(\mathbf{x}_{TD})$.

Assuming the robot's desired LO state is equal to the prior TD state with velocity direction changes, and including the stance phase energy losses, $h(\mathbf{x}_{TD})$ does not have to be directly determined. Instead, $V(\mathbf{x}_{LO}) - V(\mathbf{x}_{TD})$ will be a positive constant bound by the magnitude of the TD roll and pitch angles; where, zero desired roll and pitch will result in $V(\mathbf{x}_{LO}) - V(\mathbf{x}_{TD}) \simeq 0$. This is due to the alignment of the velocity vector and the body z-axis at TD, where the only moment during stance is that due to gravity; which tends to increase the roll and pitch angles at LO (Table II). This allows for a wide variety of trajectories as seen in Fig. 2 and 3. To stabilize at non-zero roll and pitch angles the desired TD angles can be modified as follows, $[\phi_{TD} \gamma_\phi, \theta_{TD} \gamma_\theta]$, where the adjustment parameters are learned over multiple hop cycles as,

$$\gamma_\beta = \gamma_\beta - \mu \text{sign}(\beta_{TD}) \text{sign}(\beta_{LO}) |\beta_{TD} - \beta_{LO}|.$$

The β represented the roll and pitch angles $[\phi, \theta]$, $\mu = 0.1$ is the learning rate, and the adjustment parameters are initialized to one, $[\gamma_\phi, \gamma_\theta] = 1$. This drives the LO angles to the TD angles, and therefore, when averaged over multiple hops, leads to overall stability. Finally, to maintain stability when the LO and TD angles differ, the $h(\mathbf{x}_{TD})$ must be determined. This has been achieved through conservation of angular momentum for planar motion [15] and fitted polynomials to simulated data [13]. However, a trained neural network could be capable of capturing the characteristic of the real system with the TD velocity and desired LO orientation as inputs, and the required TD orientation as outputs; and will be explored in future work.

V. TRAJECTORY TRACKING PERFORMANCE

Figures 2 and 3 show the trajectory variability both with and without drag compensation (Table IV) for constant initial conditions (keyframe α_0) including: roll, pitch, and yaw $[\phi, \theta, \psi] = [0, 30, 0]$ degrees, velocity magnitude of 5 m/s aligned with the body z-axis, trajectory time $t_m = t_2 = 1.75$ seconds, reorientation time $t_1 = t_2 - \delta_t$, and $\delta_t = 50$ ms. Each includes multiple trajectories generated over 6 seconds of operation; where Fig. 4 shows an example 3D trajectory with the individual states, operation phases, and inputs labeled. This variability from a constant α_0 shows a potential for further increases in computational efficiency by precomputing the P_l^{-1} and N_l for a single or limited set of total trajectory times t_m ; eliminating the required substitution of t_m and δ_t .

It has been shown that including linear drag in quadrotor trajectory generation can yield improvements in tracking performance [20]. However, whereas quadrotors may be able to linearize drag about an operating point, high performance hopping robots necessarily undergo significant changes in velocity over the hop cycle; necessitating the use of non-linear drag (equations 9, 10). Table IV present a comparison between drag compensated trajectories (Figs. 2.b,d,f,h, 3.b)

TABLE IV
TRAJECTORY ERROR

Trajectory Type	Fig. Ref.	Drag Comp.	RMSE Pos. (m)	RMSE Vel. (m/s)
Wall-Wall (T2)	2.a	NO	0.072	0.243
	2.b	YES	0.093	0.318
Ground-Wall (T2) to Wall-Wall (T2)	2.c	NO	0.099	0.287
	2.d	YES	0.090	0.304
Wall-Ground (T1) to Ground-Ground (T1)	2.e	NO	0.126	0.171
	2.f	YES	0.119	0.160
Ground-State (T3)	2.g	NO	0.103	0.222
	2.h	YES	0.034	0.162
Ground-Ground (T1)	3.a	NO	0.138	0.295
	3.b	YES	0.030	0.152

Error = desired (Pos./Vel.) - measured (Pos./Vel.). If two trajectory types are listed the first hop is the first listed, and the remaining hops are the second listed.

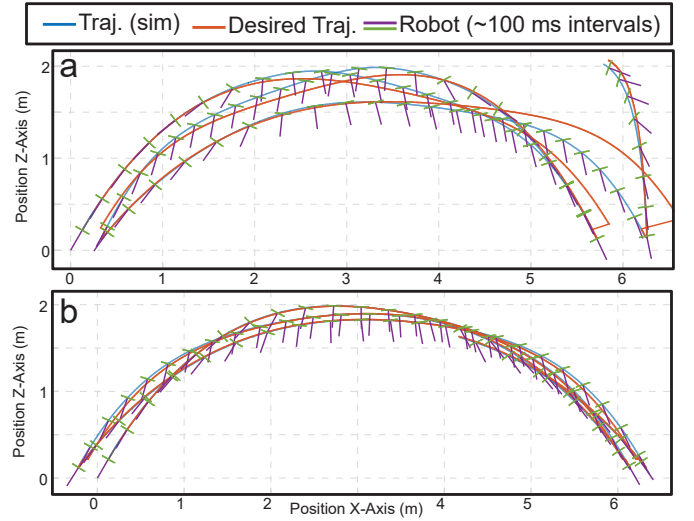


Fig. 3. Hopping trajectory generation and control (Tables II and IV show details) for constant initial conditions (keyframe α_0) including roll, pitch, and yaw of $[\phi, \theta, \psi] = [0, 30, 0]$ degrees, velocity magnitude aligned with the body z-axis of 5 m/s, trajectory time $t_m = 1.75$ seconds (LO-TD), and reorientation time $\delta_t = 0.05$ seconds; where total simulation time equals 6 seconds. The TD keyframe α_2 for all generated trajectories includes $v_{TD_d} = 5$ m/s and the desired orientation $[\phi, \theta, \psi]$ alternates between $[0, -30, 0]$ and $[0, 30, 0]$ degrees. a) No drag compensation. b) Drag compensation.

and those without (Figs. 2.a,c,e,g, 3.a); where removing the drag compensation requires setting $D_T = 0$ in equation 11 and Table II. The RMSE in position and velocity show little difference for trajectories in Fig. 2.a-f where the TD keyframe α_2 has free values as compared to the trajectories in Fig. 2.g,h with a full TD keyframe. Additionally, trajectories that maintain high horizontal velocity throughout, also show better performance when compensated for drag, as seen in Fig. 3. Therefore, as expected, drag compensation has a bigger impact on the RMSE in position and velocity for more aggressive trajectories. Finally, as seen in comparing the trajectories in Fig. 2.a,b, free values in the TD keyframe α_2 can yield very different trajectories (Table I). Therefore, if there is a generally preferred range of the free value (i.e., desire to jump-climb the wall or not), the null space N_l can be used to shift the c_j^* trajectory to achieve the desired result.

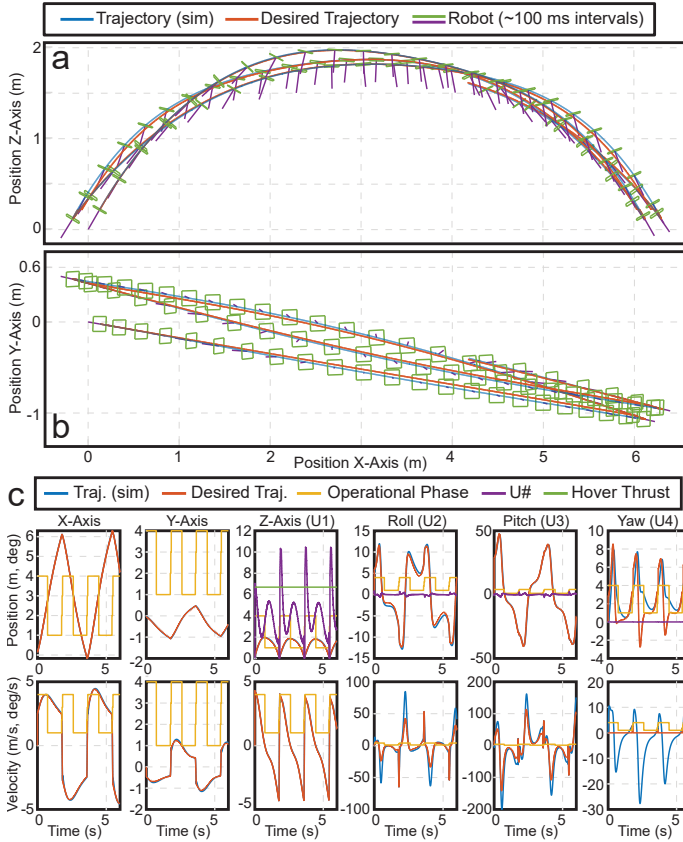


Fig. 4. Hopping trajectory generation and control (Tables II and IV show details) for constant initial conditions (keyframe α_0) including roll, pitch, and yaw of $[\phi, \theta, \psi] = [5, 30, 0]$ degrees, velocity magnitude aligned with the body z-axis of 5 m/s, trajectory time $t_m = 1.75$ seconds (LO-TD), and reorientation time $\delta_t = 0.05$ seconds; where total simulation time equals 6 seconds. The TD keyframe α_2 for all generated trajectories includes $v_{TD} = 5$ m/s and the desired orientation $[\phi, \theta, \psi]$ alternates between $[-5, -30, 0]$ and $[5, 30, 0]$ degrees. The desired trajectories are drag compensated. a) Shows the z-x plane. b) Shows the y-x plane. c) Shows the individual states.

VI. SUMMARY

This work has presented a real-time, computationally efficient, non-linear drag compensated, trajectory generation methodology and accompanying Lyapunov-based controller for hopping robot locomotion. The methodology allows for the generation of trajectories from an initial keyframe (i.e. state) at liftoff to a final desired keyframe at touchdown. This includes those leaving from, and landing on, horizontal and vertical surfaces both with and without non-linear drag compensation. The presented methodology is broadly applicable to not only hopping robots but also quadrotors that desired greater control over their orientation while maintaining computational efficiency.

REFERENCES

- [1] S. Burns and M. Woodward, "Design and Control of a High-Performance Hopping Robot," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5641–5648, Jun. 2025.
- [2] S. Burns and M. Woodward, "Optimized Kalman Filter based State Estimation and Height Control in Hopping Robots," Nov. 2024, arXiv:2408.11978 [cs].
- [3] 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, 2024.
- [4] 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," *arXiv*, 2023, arXiv: 2309.13737.
- [5] M. H. Raibert, "Hopping in Legged Systems—Modeling and Simulation for the Two-Dimensional One-Legged Case," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, 1984.
- [6] M. H. Raibert, H. B. Brown, and M. Chepponis, "Experiments in Balance with a 3D One-Legged Hopping Machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.
- [7] B. Zhu, J. Xu, A. Charway, and D. Saldana, "PogoDrone: Design, Model, and Control of a Jumping Quadrotor," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2031–2037, 2022, arXiv: 2204.00207 Publisher: IEEE ISBN: 9781728196817.
- [8] 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, 2016.
- [9] D. W. Haldane, M. Plecnik, J. K. Yim, and R. S. Fearing, "A power modulating leg mechanism for monopodal hopping," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea: IEEE, Oct. 2016, pp. 4757–4764.
- [10] M. M. Plecnik, D. W. Haldane, J. K. Yim, and R. S. Fearing, "Design exploration and kinematic tuning of a power modulating jumping monopod," *Journal of Mechanisms and Robotics*, vol. 9, no. 1, pp. 1–13, 2017.
- [11] J. S. Lee, M. Plecnik, J.-H. Yang, and R. S. Fearing, "Self-Engaging Spined Gripper with Dynamic Penetration and Release for Steep Jumps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 1–8.
- [12] 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)*. Vancouver, BC: IEEE, Sep. 2017, pp. 3345–3351.
- [13] 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)*. Madrid: IEEE, Oct. 2018, pp. 2229–2236.
- [14] J. K. Yim, E. K. Wang, and R. S. Fearing, "Drift-free roll and pitch estimation for high-acceleration hopping," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 8986–8992, 2019, ISBN: 9781538660263.
- [15] J. K. Yim, B. R. P. Singh, E. K. Wang, R. Featherstone, and R. S. Fearing, "Precision robotic leaping and landing using stance-phase balance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3422–3429, 2020.
- [16] J. Kang, Y. Wang, and X. Xiong, "Fast Decentralized State Estimation for Legged Robot Locomotion via EKF and MHE," May 2024, arXiv:2405.20567 [cs].
- [17] H. Hsiao, S. Bai, Z. Guan, S. Kim, Z. Ren, P. Chirarattananon, and Y. Chen, "Hybrid locomotion at the insect scale: Combined flying and jumping for enhanced efficiency and versatility," *Science Advances*, 2025.
- [18] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 2520–2525.
- [19] S. Bouabdallah, R. Siegwart, and G. Caprari, "Design and control of an indoor coaxial helicopter," *IEEE International Conference on Intelligent Robots and Systems*, no. April, pp. 2930–2935, 2006, ISBN: 142440259X.
- [20] J. Svacha, K. Mohta, and V. Kumar, "Improving quadrotor trajectory tracking by compensating for aerodynamic effects," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL, USA: IEEE, Jun. 2017, pp. 860–866.
- [21] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, Apr. 2018.
- [22] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems.pdf," *International Journal Robust Nonlinear Control*, vol. 8, pp. 995–1020, 1998.