Perching with a Robotic Insect using Adaptive Tracking Control and Iterative Learning Control

Pakpong Chirarattananon, Kevin Y. Ma, and Robert J. Wood*

Abstract

Inspired by the aerial prowess of flying insects, we demonstrate that their robotic counterpart, an insect-scale flapping-wing robot, can mimic an aggressive maneuver seen in natural fliers—landing on a vertical wall. Such acrobatic movement differs from simple lateral maneuvers or hover, and therefore requires additional considerations in the control strategy. In this paper, we propose a single-loop adaptive tracking flight control suite designed with an emphasis on the ability to track dynamic trajectories, and an iterative learning control algorithm to account for unmodeled dynamics and systematic errors for improved landing accuracy. Magnets were chosen to enable attachment to the vertical surface due to its simplicity. The proposed controller was verified in a series of hovering and aggressive translational flights. Furthermore, we show that by learning from previous failed attempts, the biologically-inspired robot could successfully perch on a magnetic wall after eight learning iterations.

1 Introduction

Insects are among the most diverse groups of animals on the planet. Flying insects are capable of exhibiting complex aerial feats unmatched by other flying animals. The exceptional maneuverability of these flying insects inspires scientists and engineers to advance their understanding of flapping-wing aerodynamics and insect flight (e.g. Krishnan and Sane (2014); Ristroph et al. (2012)) and translate this ubiquitous form of locomotion into man-made machines. In recent years, researchers have developed a number of biologically-inspired flapping-wing vehicles (see De Croon et al. (2012); Lentink et al. (2010); Richter and Lipson (2011); Gerdes et al. (2014)), including an insect-scale robot that successfully demonstrated unconstrained tethered flight (Ma et al., 2013; Chirarattananon et al., 2014a). The 80-mg *Micro Aerial Vehicle* (MAV) shown in figure 1 is a result of the culmination of research in mesoscale actuation and manufacturing technology (e.g. Wood et al. (2005); Sreetharan et al. (2012)).

To date, artificial flapping-wing flight at low Reynolds numbers has typically relied on passive stability to achieve hover (for examples, De Croon et al. (2012); Lentink et al. (2010); Richter and Lipson (2011); Teoh et al. (2012)). Unlike in (De Croon et al., 2012; Lentink et al., 2010; Richter and Lipson, 2011), flying insects and the insect robot in (Ma et al., 2013; Chirarattananon et al., 2014a) are inherently unstable. This instability necessitates active control, but also leads to increased maneuverability. The flapping-wing robot in figure 1 is able to generate body torques and sufficient lift force (Ma et al., 2012), satisfying the key requirements for stable flight with the aid of an active flight controller (Ma et al., 2013). In an effort to improve the flight performance demonstrated in (Ma et al., 2013), an adaptive flight controller capable of coping with model uncertainties was developed motivated by the lack of comprehensive knowledge of the system and variation caused by imperfect fabrication (Chirarattananon et al., 2014a). This brought about marked improvement in flight performance as

*This work was partially supported by the National Science Foundation (award number CCF-0926148), and the Wyss Institute for Biologically Inspired Engineering. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

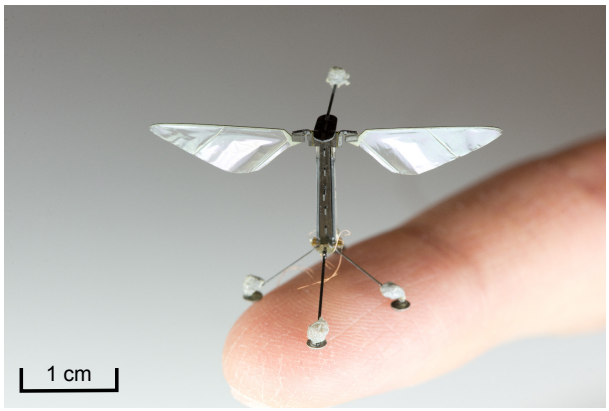

Figure 1: Photograph of a biologically-inspired robotic insect on a finger with four reflective markers for tracking purposes.

evidenced by the reduction in position errors, particularly for hovering flights. However, thus far the insect-scale robot has only performed basic flight maneuvers and stable hovering, and has yet to demonstrate any aggressive or acrobatic maneuvers as observed in natural fliers, encountering further issues in control, fast dynamics, and a lack of understanding of insect-scale unsteady aerodynamics.

To demonstrate that the biologically-inspired robot has a potential to perform acrobatic maneuvers similar to those seen in natural fliers, in this paper we address the challenges of performing an aggressive flight maneuver on the insect-scale robot. To be more specific, our ultimate objective is to design flight control algorithms that allow the robot to land, or perch, on a vertical surface. The path required to realize such goal could be divided into two steps. The first step entails a thorough re-design of the adaptive flight controller suitable for aggressive flight trajectories, and the second step involves additional control algorithms to cope with fast and unknown dynamics not captured by the simple models.

In the first part of this paper, we briefly discuss the design, manufacture, and dynamic model of the flapping-wing robot. Fundamentally, the underactuated dynamics of the robotic insect bare some similarities to the prevalent quadrotor systems (Mellinger et al., 2012; Taha et al., 2012). In section 3, we provide the derivation of the proposed adaptive tracking controller suitable for the flapping-wing robot and other classes of MAVs with similar dynamics including quadrotors. The controller was first presented with limited experimental results in (Chirarattananon et al., 2014c). This controller allows the vehicle to track trajectories in the $\mathrm{SE}(3)$ space with provable asymptotic stability under some reasonable assumptions. The nonlinear structure distinguishes the proposed controller from the more commonly adopted PID-type controllers based on linearized dynamics about hover (Deng et al., 2006; Oppenheimer et al., 2009; Huang et al., 2009). Therefore, the nonlinear controller benefits from the greater region of attraction. The proposed controller possesses sufficient fidelity to enable the robot to follow pre-defined trajectories with relatively small errors, however, it fails to capture extreme dynamics required for the robot to realize highly acrobatic maneuvers, such as perching on a vertical surface, due to the limited bandwidth and simplified dynamic model.

To further address the difficulties in performing extreme maneuvers, in section 4, we focus on the perching problem. The topic of perching an MAV has been addressed previously at larger scales (Cory and Tedrake, 2008; Kovač et al., 2009; Desbiens et al., 2011). In (Cory and Tedrake, 2008), the authors placed focus on identifying an accurate model of the dynamics and utilized a value iteration algorithm in the design of the optimal control policy. Both (Kovač et al., 2009; Desbiens et al., 2011), on the other hand, emphasized the design of novel attachment mechanisms that allowed the MAVs to perch within a large flight envelope. The very small scale and limited payload capacity of the robotic insect in this study renders elaborate perching mechanisms an impractical solution. Fortunately,

the use of magnetic force becomes more favorable at smaller scales. As the length scale decreases, weight decreases as a cubic function of the characteristic length, L^3 while magnetic forces on magnetic magnetic attachment under constant magnetic field scale approximately as a function of surface area, L^2. Essentially, magnetic forces dominate gravitational forces at small scales and when the distances are small (compared to the characteristic dimension of the object). To exploit this, we attach small steel discs on the robot to enable the robot to land on a vertical magnetic surface.

In terms of control, the *Iterative Learning Control* (ILC) technique (Bristow et al., 2006; Hehn and D'Andrea, 2014) was employed in addition to the proposed adaptive tracking controller. The formulation of the ILC algorithm in section 4, first introduced in (Chirarattananon et al., 2014b), allows the robot to learn from its previous flights and improves its flight performance through repetition of the same trajectory, accounting for the robot's dynamics that are uncaptured by the model and potentially enabling the robot to realize aggressive trajectories, such as perching, with exceptional accuracy.

Finally, the proposed methods are verified in trajectory following and vertical landing experiments. Several flights were performed to compare the performance of the proposed controller with the previously developed nonlinear controller (Chirarattananon et al., 2014a). A perching trajectory was then generated according to the constraints of the flight dynamics. The robot iteratively executed the landing trajectory several times in order to improve the tracking performance. The results are shown in section 6, followed by the conclusion and discussion.

Notation

- In equations, bold letters indicate vectors.
- Given an *unknown parameter* α, its estimate is represented by $\hat{\alpha}$. The estimation error $\tilde{\alpha}$ is defined as $\tilde{\alpha} = \hat{\alpha} - \alpha$.
- Otherwise, $\hat{\cdot}$ represents a unit vector.
- High order derivatives are denoted by bracketed superscript, i.e., $\alpha^{(n)} = d^n\alpha/dt^n$.

2 Dynamic Model of a Flapping-Wing Robot

2.1 Robot Design

The flapping-wing robot considered in this work is shown in figure 1. The robotic insect weighs 80 mg and has a wingspan of 3 cm. It is fabricated using the *Smart Composite Microstructures* (SCM) process as detailed in (Ma et al., 2012, 2013). The robot's structural components are made of lightweight laser-machined carbon fiber composite. Two wings are composed of a 1.5 μm-thick polyester membrane supported by carbon fiber spars. The actuation is provided by two piezoelectric bending actuators that can independently generate approximately linear motion at their tips when a voltage is applied. Piezoelectric actuators are chosen over electromagnetic motors for the flight muscles due to their favorable scalability (Wood et al., 2012). Due to the lack of onboard electronics in the current robot prototype, power and control signals are delivered to the robot via four 51-gauge copper wires carrying a ground signal, a 300 V voltage rail, and two drive signals. It has been projected that all necessary flight electronics (microprocessor Zhang et al. (2014), power electronics Karpelson et al. (2009), and sensors) without a battery would constitute approximately 100 mg payload (Ma et al., 2015). In parallel research, a scaled-up flapping-wing prototype with a 5.5 cm wingspan and a mass of 380 mg was designed, fabricated, and shown able to carry 115 mg dummy payload (Ma et al., 2015). The development of an autonomous insect-scale vehicle is a subject of ongoing and future research (Ma et al., 2015).

When driven, the motion of the actuator is transformed into an angular wing motion by a spherical four-bar transmission with resilient flexure joints that are fabricated from polyimide film. A passive

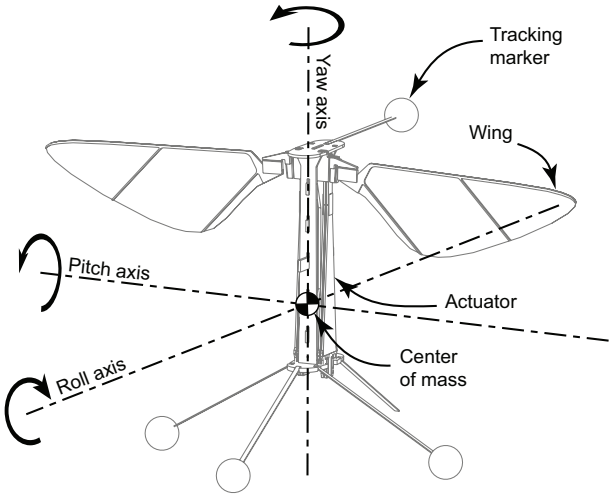

Figure 2: A schematic diagram demonstrating primary components of the robot and the definition of the roll, pitch, and yaw axes.

flexure hinge connecting the transmission to the wing interacts with the inertial and aerodynamic forces acting on the wing to produce a desired angle of attack, resulting in lift that enables the robot to fly. The actuator, transmission, and wing form a mechanical subsystem with a behavior resembling a second-order linear system (Ma et al., 2013). As a result, we nominally operate the system with sinusoidal signals near the system's resonant frequency of 120 Hz to maximize the flapping stroke amplitude and minimize reactive power expended by the wing inertia and the hinge stiffness. Lift modulation is obtained by altering the stroke amplitude. By appropriately modulating the actuator drive signals, the wing motion can be governed to create pitch, roll, and yaw torques as desired. More details on the robot design and torque generation schemes can be found in (Ma et al., 2013, 2012). In previous work (Ma et al., 2013; Chirarattananon et al., 2014a), we have developed a model that is capable of determining the drive signals given the desired stroke-averaged lift force and torques with sufficient accuracy for control proposes. In this paper, we assume that the torques can be directly commanded by the controller, and the delay in the actuation can be neglected (At 120 Hz, we anticipate that the robot would take fewer than ten flapping strokes (< 80 ms) to realize the torque commands. These assumptions are also common in the MAVs community (Lee et al., 2010; Mellinger and Kumar, 2011; Lee et al., 2009). This leaves researchers the task of designing a controller that determines the required lift and torque to stabilize the robot's flight along the pre-planned trajectories.

2.2 Flight Dynamic Model

The roll, pitch, and yaw axes of the robot are defined as depicted in figure 2. We define the inertial frame $\{\hat{I}, \hat{J}, \hat{K}\}$ such that the $\hat{K}$-axis points in the opposite direction to the gravity vector $\mathbf{g}$. The origin of the body frame $\{\hat{i}, \hat{j}, \hat{k}\}$ is located at the center of mass of the robot such that the $\hat{k}$-axis is nominally aligned with the thrust vector and against gravity when the robot is in the upright orientation as shown in figure 3. A rotation matrix $R \in \mathrm{SO}(3)$ maps the body-fixed frame to the inertial frame. The angular velocities of the robot along the $\hat{i}, \hat{j}$, and $\hat{k}$ directions are defined as $\omega = [\omega_i, \omega_j, \omega_k]^T$. The rotational dynamics of the robot depends on the total torque τ acting on the robot and is described by the Euler's equation for rigid body dynamics:

$$J\dot{\omega} = \sum \tau - (\omega \times J\omega)\,, \tag{1}$$

where J denotes the moment of inertia matrix. In this circumstance, we can choose the body-fixed frame to align with the pitch, roll, and yaw axes of the robot. The off-diagonal elements of J become

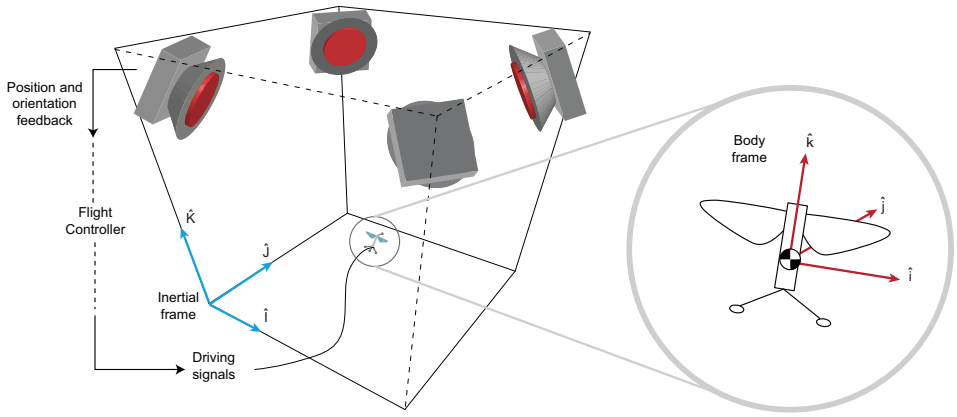

Figure 3: The flight arena is equipped with $4-8$ motion capture cameras for position and orientation feedback. Definitions of the inertial frame and the body-attached frames are shown.

negligible due to the symmetry of the vehicle and the torque vector is defined directly along the pitch, roll, and yaw axes, simplifying the dynamics problem.

The time derivative of the rotation matrix $\dot{R}$ is given as $\dot{R} = R\,[\omega\times]$, where the map $[(\cdot)\times] : \mathbb{R}^3 \mapsto \mathfrak{so}(3)$, with $\mathfrak{so}(3)$ being the Lie algebra of SO (3), is defined such that $[\mathbf{x}\times]\,\mathbf{y} = \mathbf{x}\times\mathbf{y}$ for all $\mathbf{x},\mathbf{y}\in\mathbb{R}^3$. Note that since the elements of R can also be represented using $\hat{\imath},\hat{\jmath}$, and $\hat{k}$ as $R = \begin{bmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \end{bmatrix}$, an alternative representation of $\dot{R}$ can be written as

$$\begin{aligned} \dot{R} &= \begin{bmatrix} \dot{\hat{\imath}} & \dot{\hat{\jmath}} & \dot{\hat{k}} \end{bmatrix} \\ &= \begin{bmatrix} \omega_k\hat{\jmath} - \omega_j\hat{k} & \omega_i\hat{k} - \omega_k\hat{\imath} & \omega_j\hat{\imath} - \omega_i\hat{\jmath} \end{bmatrix}. \end{aligned} \tag{2}$$

The translational dynamics of the robot also depends on the orientation of the robot. In other words, the normalized thrust Γ (which has a dimension of acceleration, not force) is nominally aligned with the $\hat{k}$ axis of the robot. It follows that we can write the equation of motion of the robot as

$$m\begin{bmatrix} \ddot{X} & \ddot{Y} & \ddot{Z} \end{bmatrix}^T = m\mathbf{g} + m\Gamma\hat{\mathrm{k}}, \tag{3}$$

where m denotes the mass of the robot, and X, Y, and Z are position of the robot in the inertial frame. The thrust produced by the robot Γ is modeled to relate to the commanded thrust T by a first-order differential equation and a gain factor γ:

$$\dot{\Gamma} = \gamma\,(T - \Gamma)\,. \tag{4}$$

In equations (1) and (3), we have not taken into consideration additional aerodynamic effects that arise in free flight. Such effects, including unsteady flow, are difficult to accurately capture using simple models that are suitable for real-time control purposes. These damping effects are often neglected in the control problems of MAVs (Huang et al., 2010; Lee et al., 2010; Mellinger and Kumar, 2011). In the study of insect flight, however, it has been shown that they could be approximately captured as linear terms in the rotational dynamics and the translational dynamics (Faruque and Sean Humbert, 2010; Ristroph et al., 2013). While these additional terms may be negligible in hovering flight, they could become significant in more aggressive flight, in which case they are treated as disturbances in this work.

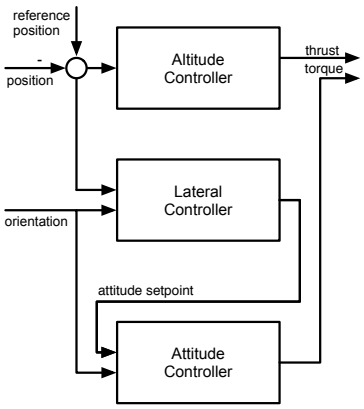

Figure 4: A block diagram illustrating the structure of the flight controller in Chirarattananon et al. (2014a). The lateral controller computes the attitude setpoint as an input for the attitude controller.

3 Single-Loop Adaptive Tracking Control

The inherent instability of flapping-wing MAVs (Ristroph et al., 2013; Orlowski and Girard, 2012) requires active flight control. To prevent the robot from crashing, the attitude controller must nominally align the robot's thrust vector against gravity. In (Ma et al., 2013), we demonstrated that using a flight controller that possessed a large region of attraction over the $\mathrm{SO}(3)$ space, the flapping-wing robot achieved stable hovering flights. One distinctive character of the proposed controller is the relaxation of control over the exact yaw orientation as it is dispensable in controlling the lateral position of the vehicle as described in equation (3).

To attain more precise hovering, we identified several critical unknown parameters that significantly affect the flight performance and re-designed an adaptive flight controller using sliding mode control techniques (Chirarattananon et al., 2014a). In this case, the lateral position of the robot is regulated by changing the attitude setpoint of the robot, while the altitude is controlled separately. In other words, the lateral controller and the attitude controller operate in a cascaded fashion. The lateral controller determines the attitude setpoint by assuming that the attitude dynamics are considerably faster than the lateral dynamics, and therefore the closed-loop attitude dynamics can be treated as a first order lowpass system. In the mean time, the attitude controller attempts to realize the attitude setpoint and minimize the angular velocity of the robot. A block diagram summarizing key components of this control architecture is shown in figure 4. This markedly improved the accuracy in position and substantially reduced visible oscillations during hovering flights as compared with (Ma et al., 2013). Simple lateral maneuvers were also demonstrated; nonetheless, there was significant room for improvement.

One downside of having a cascaded control structure as in (Chirarattananon et al., 2014a) and other related literature (Achtelik et al., 2013; Mellinger et al., 2012) is a possible loss in maneuvering precision due to the assumption that the inner control loop is considerably faster than the outer loop. Another drawback of the proposed controller in (Chirarattananon et al., 2014a) is that the attitude controller always tries to minimize the rotational rate. Since the rotational rate is related to the third order derivative of the position, it is anticipated that a controller that effectively determines a suitable angular velocity setpoint would bring about better performance in trajectory following, particularly when more aggressive movements are involved.

In (Lee et al., 2010), a nonlinear controller that explicitly tracks trajectories in $\mathrm{SE}(3)$ was proposed and proved to have exponential attractiveness over a large region. This controller requires a predefined trajectory that includes reference position, orientation, and angular velocity. A variation of this controller was implemented for quadrotors in (Mellinger and Kumar, 2011). Some existing state-of-the-art flight controllers for MAVs with similar dynamics use the differential flatness property of

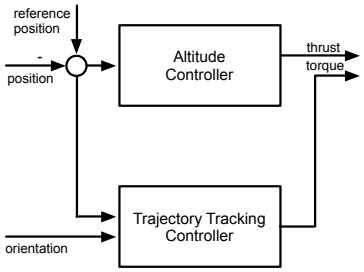

Figure 5: A simplified block diagram showing the underlying structure of the proposed single-loop tracking controller. The attitude controller in figure 4 is incorporated into the lateral controller, becoming a trajectory tracking controller, which operates in parallel to the altitude controller.

the vehicle to construct control commands for four rotors from a pre-defined set of four parameters including X, Y, Z position in the inertia frame and the yaw angle (Mellinger and Kumar, 2011; Achtelik et al., 2013). This results in controllers that command the thrust and angular velocities of the vehicle. In our case, previously developed controllers possess two primary shortcomings. First, our flapping-wing robot lacks an ability to reliably generate yaw torque via the split-cycle wing kinematics as proposed in (Oppenheimer et al., 2009) and implemented in (Ma et al., 2013; Chirarattananon et al., 2014a). This renders in impractical to directly control the heading (or yaw angle) of the robot, which is not necessary for position control. The existing controllers becomes unsuitable as it is not trivial to evaluate the angular velocity setpoint from the pre-planned trajectory and the current state of the vehicle when the heading (yaw orientation) is not directly specified. Second, adaptability is a desired property that was proved to be crucial for performance of flight at this scale (Chirarattananon et al., 2014a) (further supporting evidence is provided in the analysis of the results in section 6.1). It is not straightforward to design a controller with proven stability that satisfies the mentioned specifications. The complication arises as we try to retain provable Lyapunov stability while the exact yaw orientation of the robot is not directly controlled. Current adaptive flight controllers possess some restrictions unsuitable for our robot. For instance, the controller in (Dydek et al., 2013) was designed based on linearized dynamics, Nicol et al. (2011) presents an adaptive controller for flight stabilization without direct position control or trajectory tracking, and the backstepping method in (Huang et al., 2009) is complicated in design and was verified in simulation only. Moreover, in these examples, none considered flight control without direct yaw control. In this paper, we propose a Lyapunov function that comprises variables made of various derivatives of position error projected on to suitable directions. The outcome is a tracking controller that directly regulates the desired torques from position errors, eliminating the former cascaded structure and, thus, dropping the assumption regarding the response of the attitude controller. The resultant product is a more versatile controller that is capable of more aggressive trajectory following in addition to only steady hovering. The Lyapunov formulation of the proposed adaptive component is compatible with the sliding mode approach we used for designing the tracking controller. The scheme benefits from ease of implementation, guaranteed convergence, and compatibility with the vehicle's nonlinear dynamics, compared to alternatives such as L_1-adaptive control that requires linearized dynamics (Ioannou et al., 2014; Guerreiro et al., 2009) or neural-based adaptive method that necessitates a training period (Nicol et al., 2011; Coza et al., 2011; Madani and Benallegue, 2008).

In this section, we present the derivation of the proposed altitude controller and the trajectory tracking controller, briefly introduced in (Chirarattananon et al., 2014c), based on techniques borrowed from the sliding mode control method (Slotine et al., 1991; Bouabdallah and Siegwart, 2005). Although they are presented separately as illustrated in figure 5, they operate in parallel rather than in a cascaded configuration. As a consequence, they can technically be classified as a single control loop. For clarity, we initially restrict the presentation to a non-adaptive system. Later, the controller is extended to accommodate an adaptive component and the stability is verified via Lyapunov's direct method. At

the end of the section, we also show that it is possible to directly control the heading of the robot without theoretically violating any the stability conditions nor deteriorating the tracking performance.

3.1 Altitude Control

To begin, we define a position vector ($\mathbf{r}$) and the desired position vector ($\mathbf{r}_d$) of the robot with respect to the inertial frame:

$$\begin{aligned} \mathbf{r} &= \begin{bmatrix} X & Y & Z \end{bmatrix}^T \\ \mathbf{r}_d &= \begin{bmatrix} X_d & Y_d & Z_d \end{bmatrix}^T. \end{aligned} \tag{5}$$

Given the robot's thrust Γ and the gravity vector $\mathbf{g}$ from equation (3), the translational dynamics of the robot are described by

$$\begin{aligned} \ddot{\mathbf{r}} &= \Gamma\hat{k} + \mathbf{g} = \Gamma R \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T + \mathbf{g} \\ &= \Gamma \begin{bmatrix} R_{13} & R_{23} & R_{33} \end{bmatrix}^T - \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T. \end{aligned} \tag{6}$$

The altitude dynamics are given by the third row of equation (6).

$$\ddot{Z} = \ddot{\mathbf{r}} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T = \Gamma R_{33} - g \tag{7}$$

For control purposes, we define an auxiliary variable S_Γ and the variable $\ddot{Z}_r$ as the following:

$$\begin{aligned} S_\Gamma &= \left(\ddot{Z} - \ddot{Z}_d\right) + \Lambda_1 \left(\dot{Z} - \dot{Z}_d\right) + \Lambda_2 \left(Z - Z_d\right) \\ &= \ddot{Z} - \ddot{Z}_r, \end{aligned} \tag{8}$$

where Λ_i's are positive constants and the subscript d denotes "desired" or reference trajectories. The sliding surface is described by $S_\Gamma = 0$. On this sliding surface, the tracking error, or the difference between Z and Z_d exponentially reduces to zero. The controller is designed to drive the system towards this sliding surface.

According to equations (4), (7), and (8), the time derivative of S_Γ is

$$\begin{aligned} \dot{S}_\Gamma &= \dot{\Gamma} R_{33} + \Gamma \dot{R}_{33} - \frac{d}{dt}\ddot{Z}_r \\ &= \gamma \left(T - \Gamma\right) R_{33} + \Gamma \left(-R_{32}\omega_i + R_{31}\omega_j\right) - \frac{d}{dt}\ddot{Z}_r. \end{aligned} \tag{9}$$

Here we propose a Lyapunov function candidate $V_\Gamma\left(S_\Gamma\right) : \mathbb{R} \mapsto \mathbb{R}$,

$$V_\Gamma = \frac{1}{2} S_\Gamma^2 \geq 0. \tag{10}$$

The proposed function V_Γ satisfies the radially unbounded property, that is $S_\Gamma \to \infty \Rightarrow V_\Gamma \to \infty$. Subsequently, the following control law

$$T = \Gamma - \gamma^{-1} R_{33}^{-1} \left[\Gamma \left(-R_{32}\omega_i + R_{31}\omega_j\right) - \frac{d}{dt}\ddot{Z}_r + K_\Gamma S_\Gamma\right], \tag{11}$$

with a positive constant gain K_Γ and the measured thrust from equation (4) given as $\Gamma = \|\ddot{\mathbf{r}} - \mathbf{g}\|$ renders the derivative of the Lyapunov function candidate negative definite

$$\dot{V}_\Gamma = S_\Gamma \dot{S}_\Gamma = -K_\Gamma S_\Gamma^2 \leq 0\ . \tag{12}$$

According to the invariant set theorem, the system is globally asymptotically stable in a Lyapunov sense (Slotine et al., 1991). However, in the physical system, the region of attraction is reduced owing to the thrust limits of the robot and its inability to generate negative thrust. Note that the proposed control law in equation (11) includes the first order derivative of $\ddot{Z}_r$. From the definition in equation (8), $\ddot{Z}_r$ contains the second derivative of the reference Z_d and the first derivative of Z, resulting in the control law that requires up to the second derivative in altitude measurement. In practice, the noise level in $\ddot{Z}$ is acceptable.

3.2 Trajectory Tracking Control

Since the angular velocity is related to the third-order derivative of the robot's position, we consider an auxiliary variable $\mathbf{e}$—a vector quantity we would like to minimize, made up of the differences between the robot's position and the setpoint and their derivatives.

$$\begin{aligned} \mathbf{e} &= \left(\mathbf{r}^{(3)} - \mathbf{r}_d^{(3)}\right) + \lambda_1\left(\ddot{\mathbf{r}} - \ddot{\mathbf{r}}_d\right) + \lambda_2\left(\dot{\mathbf{r}} - \dot{\mathbf{r}}_d\right) + \lambda_3\left(\mathbf{r} - \mathbf{r}_d\right) \\ &= \mathbf{r}^{(3)} - \mathbf{r}_r^{(3)}, \end{aligned} \tag{13}$$

where we have defined $\mathbf{r}_r^{(3)}$ accordingly. Using equations (2) and (6), the third derivative of $\mathbf{r}$ then becomes

$$\mathbf{r}^{(3)} = \Gamma\dot{\hat{k}} + \dot{\Gamma}\hat{k} = \Gamma\left(-\omega_i\hat{j} + \omega_j\hat{i}\right) + \dot{\Gamma}\hat{k}. \tag{14}$$

We propose the following composite variable $\mathbf{S}_\tau$ based on the projection of the auxiliary variable $\mathbf{e}$ along the body axes of the robot, and the Lyapunov function candidate V_τ:

$$\begin{aligned} \mathbf{S}_\tau &= \begin{bmatrix} -\mathbf{e}\cdot\hat{j}/\Gamma & \mathbf{e}\cdot\hat{i}/\Gamma & \omega_k \end{bmatrix}^T \\ &= \begin{bmatrix} \omega_i + \Gamma^{-1}\left(\mathbf{r}_r^{(3)}\cdot\hat{y}\right) \\ \omega_j - \Gamma^{-1}\left(\mathbf{r}_r^{(3)}\cdot\hat{x}\right) \\ \omega_k \end{bmatrix}, \end{aligned} \tag{15}$$

$$V_\tau = \frac{1}{2}\mathbf{S}_\tau^T J \mathbf{S}_\tau. \tag{16}$$

The sliding surface $\mathbf{S}_\tau = 0$ is obtained when $\mathbf{e}$ becomes zero (with the rare exception when $\mathbf{e}$ is parallel to $\hat{k}$) and the yaw rate (ω_k) is minimized. Notice that angular velocity terms appear in equation (15), linking the attitude dynamics to the lateral dynamics. The third element of $\mathbf{S}_\tau$ is chosen to be ω_k, consistent with the decision not to directly control the heading of the robot, but only to damp out the yaw rotation rate that may arise. The quadratic structure means that the Lyapunov function candidate has the positive definite and radially unbounded properties as desired. Using equations (1) and (15), we can write the derivative of the composite variable as

$$J\dot{\mathbf{S}}_\tau = \tau - (\omega \times J\omega) - \Gamma^{-1} J \frac{d}{dt}\begin{bmatrix} -\left(\mathbf{r}_r^{(3)}\cdot\hat{j}\right) \\ \left(\mathbf{r}_r^{(3)}\cdot\hat{i}\right) \\ 0 \end{bmatrix}.$$

This suggests the commanded body torque

$$\tau = -\Gamma^{-1}\begin{bmatrix} \mathbf{r}_r^{(3)}\cdot\hat{j} \\ -\mathbf{r}_r^{(3)}\cdot\hat{i} \\ 0 \end{bmatrix} \times J\omega + \Gamma^{-1} J \frac{d}{dt}\begin{bmatrix} -\left(\mathbf{r}_r^{(3)}\cdot\hat{j}\right) \\ \left(\mathbf{r}_r^{(3)}\cdot\hat{i}\right) \\ 0 \end{bmatrix} - K_\tau \mathbf{S}_\tau, \tag{17}$$

so that the time derivative of the proposed Lyapunov function candidate is negative definite and the system is proven asymptotically stable:

$$\begin{aligned} \dot{V}_\tau &= -\mathbf{S}_\tau^T K_\tau \mathbf{S}_\tau - \mathbf{S}_\tau^T\left(\mathbf{S}_\tau \times J\omega\right) \\ &= -\mathbf{S}_\tau^T K_\tau \mathbf{S}_\tau \le 0. \end{aligned} \tag{18}$$

To evaluate the domain of attraction of the controller, observe that the Lyapunov function candidate in equation (16) is zero only when the robot has no angular velocity ω_k, and the auxiliary variable $\mathbf{e}$ is

zero or parallel to the $\hat{k}$-axis. The former only happens when the robot tracks the reference trajectory perfectly, and the latter implies that the error is along the thrust direction, in which case it will be taken care of by the altitude controller as demonstrated in section 3.1. The exception occurs when the vector $\mathbf{e}$ points in the opposite direction to the $\hat{k}$-axis. In that circumstance, the control signal makes no attempt to generate any torques and correct for the error. Therefore, the proposed control law is almost globally asymptotically stable. One contribution to the large region of attraction achieved here is owing to the absence of singularities associated with representations of SO (3) such as Euler angles or ambiguities of quaternions.

Examining the control law in equation (17), it can be seen that the third derivative of the measurement of $\mathbf{r}$ embedded in the term $-K_\tau \mathbf{S}_\tau$ could be written in terms of the angular velocity as given in equation (14). Therefore, only the second derivative of $\mathbf{r}$ is required, alongside the body orientation and its first derivative. Furthermore, that fact that $\mathbf{r}_d^{(3)}$ and $\ddot{\mathbf{r}}_d$ is included in $\mathbf{r}_r^{(3)}$ implies that the controller offers an improved trajectory tracking performance by effectively tracking a setpoint for the angular velocity in addition to the acceleration—the property lacking in previous work (Chirarattananon et al., 2014a). In the absence of the tracking error ($\mathbf{S}_\tau = 0$), the first two terms in equation (17) remain non-zero unless $\mathbf{r}_d^{(3)}$ is zero. In other words, they serve as a feedforward term in the proposed control law.

3.3 Adaptive Control

In (Chirarattananon et al., 2014a), we identified six unknown parameters that were crucial to accomplish steady hover for the millimeter-scale robot: the misalignment of the thrust vector from the $\hat{k}$ axis (ϵ_i and ϵ_j), three unknown torque offsets ($\tau_o = \begin{bmatrix} \tau_{oi} & \tau_{oj} & \tau_{ok} \end{bmatrix}^T$), and the normalized thrust offset (T_o). In this section, we present how the proposed controller is modified to take into account the effects of these unknowns, starting by including those effects into the dynamic model. We then present how the composite variables and the control laws should be altered. A predictor and an adaptive component are implemented to ensure that the estimates of the unknowns converge to their true values and stability is still guaranteed as shown in the Lyapunov analysis at the end of the section.

3.3.1 Altitude control law

For a small deviation of the thrust vector from the presumed robot $\hat{k}$-axis, the thrust takes on small lateral components along the $\hat{\imath}$ and $\hat{\jmath}$ axes ($\epsilon_i, \epsilon_j \ll 1$), resulting in a slight modification to equation (6),

$$\ddot{\mathbf{r}} = \Gamma \left(\hat{k} + \epsilon_i \hat{\imath} - \epsilon_j \hat{\jmath} \right) - \mathbf{g}. \tag{19}$$

Similarly, the thrust dynamics are modified to include the unknown offset T_o by substituting T by $T_c - T_o$ into equation (4), where T_c is the commanded thrust input. This yields

$$\dot{\Gamma} = \gamma \left(T_c - T_o - \Gamma \right), \tag{20}$$

The derivative of the composite variable defined in equation (8) becomes

$$\begin{aligned} \dot{S}_\Gamma &= \gamma \left(T_c - T_o - \Gamma \right) \left(R_{33} + \epsilon_i R_{31} - \epsilon_j R_{32} \right) + \Gamma \left(-R_{32}\omega_i + R_{31}\omega_j \right) \\ &\quad + \Gamma \epsilon_i \left(-R_{33}\omega_j + R_{32}\omega_k \right) + \Gamma \epsilon_j \left(-R_{33}\omega_i + R_{31}\omega_k \right) - \frac{d}{dt} \ddot{Z}_r. \end{aligned} \tag{21}$$

We define

$$\mu = \Gamma \hat{\epsilon}_i \left(-R_{33}\omega_j + R_{32}\omega_k \right) + \Gamma \hat{\epsilon}_j \left(-R_{33}\omega_i + R_{31}\omega_k \right) + \Gamma \left(-R_{32}\omega_i + R_{31}\omega_j \right) - d\ddot{Z}_r/dt, \tag{22}$$

and propose the following control signal

$$T_c = \hat{T}_o + \Gamma - \gamma^{-1} R_{33}^{-1} \left(1 - \hat{\epsilon}_i R_{31} R_{33}^{-1} + \hat{\epsilon}_j R_{32} R_{33}^{-1} \right) \left(\mu + K_\Gamma S_\Gamma \right). \tag{23}$$

By using the approximation

$$\left(1-\hat{\epsilon}_i R_{31}R_{33}^{-1}+\hat{\epsilon}_j R_{32}R_{33}^{-1}\right)=\left(1+\hat{\epsilon}_i R_{31}R_{33}^{-1}-\hat{\epsilon}_j R_{32}R_{33}^{-1}\right)^{-1}+\mathcal{O}\left(\hat{\epsilon}^2\right) \tag{24}$$

it can be shown that the proposed control law makes the derivative of the composite variable in equation (21) expressible as

$$\begin{aligned}\dot{S}_\Gamma &= -K_\Gamma S_\Gamma+\begin{bmatrix}\gamma\left(R_{33}+R_{31}\hat{\epsilon}_i-R_{32}\hat{\epsilon}_j\right)\\ \Gamma R_{33}\omega_j-\Gamma R_{32}\omega_k+R_{31}R_{33}^{-1}\mu\\ \Gamma R_{33}\omega_i-\Gamma R_{31}\omega_k-R_{32}R_{33}^{-1}\mu\end{bmatrix}^T\begin{bmatrix}\tilde{T}_o\\ \tilde{\epsilon}_i\\ \tilde{\epsilon}_j\end{bmatrix}+\gamma\tilde{T}_o\left(-R_{31}\tilde{\epsilon}_i+R_{32}\tilde{\epsilon}_j\right)+\mathcal{O}\left(\hat{\epsilon}^2\right)\\ &\approx -K_\Gamma S_\Gamma+Y_\Gamma^T\tilde{\mathbf{a}}+\gamma\tilde{T}_o\left(-R_{31}\tilde{\epsilon}_i+R_{32}\tilde{\epsilon}_j\right),\end{aligned} \tag{25}$$

where we have defined $\mathbf{a}$ as a vector consisting of the three unknown parameters and the vector Y_Γ accordingly. For small deviations ($\epsilon_i, \epsilon_j \approx 0.1$), and a reasonably large tilt angle ($R_{31}R_{33}^{-1}, R_{32}R_{33}^{-1} \approx 1$) the error caused from the approximation in equation (24) is less than one percent. The first term in equation (25) is identical to the non-adaptive case in equation (12). The vector Y_Γ in the second term contains only known and measurable quantities. These two terms are the typical form that usually appear in the derivation of adaptive sliding mode controllers (Slotine et al., 1991; Chirarattananon et al., 2014a). To achieve Lyapunov stability, the last term has to be handled explicitly as described in the last paragraph of section 3.3.4.

3.3.2 Trajectory tracking control law

Including the effect of ϵ_i and ϵ_j, we define an estimate of $\mathbf{r}^{(3)}$ based on the estimates of ϵ_i and ϵ_j based on equation (14):

$$\hat{\mathbf{r}}^{(3)}=\Gamma\left(\dot{\hat{k}}+\hat{\epsilon}_i\dot{\hat{i}}-\hat{\epsilon}_j\dot{\hat{j}}\right)+\dot{\Gamma}\left(\hat{k}+\hat{\epsilon}_i\hat{i}-\hat{\epsilon}_j\hat{k}\right). \tag{26}$$

It follows that we can also define an estimate of $\mathbf{e}$ from equation (13) as $\hat{\mathbf{e}}=\hat{\mathbf{r}}^{(3)}-\mathbf{r}_r^{(3)}$, the definition of $\mathbf{r}_r^{(3)}$ from equation (13) remains unchanged. As a consequence, the composite variable of the tracking controller is re-defined as $\hat{\mathbf{S}}_\tau$:

$$\hat{\mathbf{S}}_\tau = \begin{bmatrix}-\left(\hat{\mathbf{e}}\cdot\hat{j}\right)\Gamma^{-1}\\ \left(\hat{\mathbf{e}}\cdot\hat{i}\right)\Gamma^{-1}\\ \omega_k\end{bmatrix}=\begin{bmatrix}\omega_i-\hat{\epsilon}_i\omega_k+\Gamma^{-1}\left(\mathbf{r}_r^{(3)}\cdot\hat{j}\right)\\ \omega_j+\hat{\epsilon}_j\omega_k-\Gamma^{-1}\left(\mathbf{r}_r^{(3)}\cdot\hat{i}\right)\\ \omega_k\end{bmatrix}+\frac{\dot{\Gamma}}{\Gamma}\begin{bmatrix}\hat{\epsilon}_j\\ -\hat{\epsilon}_i\\ 0\end{bmatrix} \tag{27}$$

In general maneuvers, Γ does not vary appreciably from g. The last term in equation (27) can be neglected. Without the knowledge of the true ϵ_i and ϵ_j, it is sensible to minimize $\hat{\mathbf{S}}_\tau$.

The next step in the derivation involves taking the time derivative of $\hat{\mathbf{S}}_\tau$, which inevitably introduces the $\dot{\mathbf{r}}_r^{(3)}$ term. This $\dot{\mathbf{r}}_r^{(3)}$ term contains unknown variables (ϵ_i, ϵ_j). To ensure that they are absent from the control law, we define $\dot{\hat{\mathbf{r}}}_r^{(3)}$ from equation (13) and (26) to exclude unknown variables (ϵ_i, ϵ_j) as

$$\begin{aligned}\dot{\hat{\mathbf{r}}}_r^{(3)} &= \mathbf{r}_d^{(4)}-\lambda_1\left(\hat{\mathbf{r}}^{(3)}-\mathbf{r}_d^{(3)}\right)-\lambda_2\left(\ddot{\mathbf{r}}-\ddot{\mathbf{r}}_d\right)-\lambda_3\left(\dot{\mathbf{r}}-\dot{\mathbf{r}}_d\right)\\ &= \dot{\mathbf{r}}_r^{(3)}-\lambda_1\left(\hat{\mathbf{r}}^{(3)}-\mathbf{r}^{(3)}\right),\end{aligned} \tag{28}$$

such that when the terms with $\dot{\Gamma}$ are neglected, equations (26) and (28) give

$$\begin{aligned}\dot{\mathbf{r}}_r^{(3)}\cdot\hat{j} &= \dot{\hat{\mathbf{r}}}_r^{(3)}\cdot\hat{j}+\lambda_1\Gamma\omega_k\tilde{\epsilon}_i\\ \dot{\mathbf{r}}_r^{(3)}\cdot\hat{i} &= \dot{\hat{\mathbf{r}}}_r^{(3)}\cdot\hat{i}+\lambda_1\Gamma\omega_k\tilde{\epsilon}_j.\end{aligned} \tag{29}$$

Expressing the body torque as the commanded torque τ_c and the unknown offset τ_o, $\tau = \tau_c - \tau_o$, we propose the following control law:

$$\begin{aligned}\tau_c &= \hat{\tau}_o - \begin{bmatrix} \Gamma^{-1}\mathbf{r}_r^{(3)} \cdot \hat{j} - \omega_k \hat{\epsilon}_i \\ -\Gamma^{-1}\mathbf{r}_r^{(3)} \cdot \hat{i} + \omega_k \hat{\epsilon}_j \\ 0 \end{bmatrix} \times J\omega + \mathbf{\Gamma^{-1}J} \begin{bmatrix} -\left(\mathbf{r}_r^{(3)} \cdot \dot{\hat{j}} + \dot{\hat{\mathbf{r}}}_r^{(3)} \cdot \hat{j}\right) \\ \left(\mathbf{r}_r^{(3)} \cdot \dot{\hat{i}} + \dot{\hat{\mathbf{r}}}_r^{(3)} \cdot \hat{i}\right) \\ 0 \end{bmatrix} \\ &\quad + J \begin{bmatrix} \hat{\epsilon}_i \dot{\omega}_k + \dot{\hat{\epsilon}}_i \omega_k \\ \text{-}\hat{\epsilon}_j \dot{\omega}_k \text{-} \dot{\hat{\epsilon}}_j \omega_k \\ 0 \end{bmatrix} - K_\tau \hat{\mathbf{S}}_\tau. \end{aligned} \tag{30}$$

It can be shown that using the proposed control law and equations (16), (27), and (29) the time derivative of the re-defined composite variable is

$$J\dot{\hat{\mathbf{S}}}_\tau = -K_\tau \hat{\mathbf{S}}_\tau - \hat{\mathbf{S}}_\tau \times J\omega + \left(\tilde{\tau}_o + Y_\tau \tilde{\mathbf{a}}\right), \tag{31}$$

where

$$Y_\tau = \begin{bmatrix} 0 & \lambda_1 J_i \omega_k & 0 \\ 0 & 0 & -\lambda_1 J_j \omega_k \\ 0 & 0 & 0 \end{bmatrix}. \tag{32}$$

Equation (31) is affine in the estimation errors $\tilde{\tau}_o$ and $\tilde{\mathbf{a}}$, rendering it possible to apply an adaptive algorithm later. Observe that this proposed commanded torque τ_c only contains measurable variables and the adaptive parameters ($\dot{\hat{\epsilon}}_i$ and $\dot{\hat{\epsilon}}_j$) that will be given in section 3.3.4.

3.3.3 Predictor

Prior to presenting the adaptive algorithm, we first design a predictor. The idea is that some parameter errors are reflected in the prediction errors. This information could be used in conjunction with the regular tracking error to estimate the unknown parameters. This strategy is generally known as composite adaptation (Slotine et al., 1991). In our case, the predictor also has a vital role in the stability property of the Lyapunov function candidate shown in the next section.

First, we consider the altitude dynamics. For a Laplace variable s, we define a first-order lowpass filter function $f_\alpha(\cdot) = \alpha(s+\alpha)^{-1}$. The thrust dynamics in equation (20) can be rewritten as $\Gamma = f_\alpha(T_C - T_o) = f_\alpha(T_c) - T_o$. The translational dynamics of the robot in equation (19) could be expressed as

$$\ddot{\mathbf{r}} - f_\alpha(T_C)\hat{k} - \mathbf{g} = \epsilon_x f_\alpha(T_C)\hat{i} - \epsilon_y f_\alpha(T_C)\hat{j} - T_o\hat{k} + \mathcal{O}(\epsilon T_o). \tag{33}$$

By neglecting the second-order uncertainties, equation (33) can be re-arranged such that the right hand side is segregated as a linear function of $\mathbf{a}$. A lowpass filter function $f_\alpha(\cdot)$ is then applied twice throughout in order to make all terms in the equation causal

$$\begin{aligned} f_\alpha^2\left(\ddot{\mathbf{r}} - f_\alpha(T_C)\hat{k} - \mathbf{g}\right) &\approx f_\alpha^2\left(\begin{bmatrix} -\hat{z} & f_\alpha(\Gamma_c)\hat{i} & f_\alpha(\Gamma_c)\hat{j} \end{bmatrix}\right)\mathbf{a} \\ &= -W\mathbf{a}. \end{aligned} \tag{34}$$

At this step, we can split $\mathbf{a}$ as its estimate $\hat{\mathbf{a}}$ and the estimation error $\tilde{\mathbf{a}}$ and organize them such that all measurable and known quantities are on the left hand side of the equation and define them as ε

$$\begin{aligned} f_\alpha^2\left(\ddot{\mathbf{r}} - f_\alpha(T_C)\hat{k} - \mathbf{g}\right) + W\hat{\mathbf{a}} &= W\tilde{\mathbf{a}} \\ \varepsilon &= W\tilde{\mathbf{a}}. \end{aligned} \tag{35}$$

The interpretation of this equation is that estimation errors $\tilde{\mathbf{a}}$ are reflected into measurable quantities ε via a projection W. This will be combined with the tracking errors to produce a control law that guarantees the convergence of unknown parameters in the next section.

3.3.4 Lyapunov analysis

Based on results presented thus far from sections 3.3.1-3.3.3, in this part, we propose a single Lyapunov function candidate for position control of the flapping-wing robot of interest:

$$V = \frac{1}{2}S_\Gamma^2 + \frac{1}{2}\hat{S}_\tau^T J \hat{S}_\tau + \frac{1}{2}\tilde{\mathbf{a}}^T \Upsilon^{-1}\tilde{\mathbf{a}} + \frac{1}{2}\tilde{\tau}_o \Psi^{-1}\tilde{\tau}_o. \tag{36}$$

The first two terms in equation (36) correspond to errors in the altitude and translational dynamics similar to the non-adaptive Lyapunov function candidates proposed in section 3.1 and 3.2. The latter two terms penalize the estimation errors of unknown parameters. The variables Υ and Ψ are positive diagonal matrices acting as adaptive gains. It follows that the time derivative of the Lyapunov function candidate could be found by substituting in the results from equation (25) and (31),

$$\begin{aligned} \dot{V} &= -K_\Gamma S_\Gamma^2 - \hat{\mathbf{S}}_\tau^T K_\tau \hat{\mathbf{S}}_\tau \\ &\quad + \dot{\hat{\tau}}^T \Psi^{-1}\tilde{\tau}_o + \hat{\mathbf{S}}_\tau^T\left(\tilde{\tau}_o + Y_\tau \tilde{\mathbf{a}}\right) \\ &\quad + \dot{\hat{\mathbf{a}}}^T \Upsilon^{-1}\tilde{\mathbf{a}} + + S_\Gamma Y_\Gamma \tilde{\mathbf{a}} + \gamma S_\Gamma \tilde{T}_o\left(-R_{31}\tilde{\epsilon}_i + R_{32}\tilde{\epsilon}_j\right). \end{aligned} \tag{37}$$

The first two terms are desirable as they are upper bounded by zero, driving the Lyapunov function candidate towards zero. The adaptive schemes are designed to get rid of the other terms. To eliminate the estimation error terms, the adaptive law for the unknown torque offsets is

$$\dot{\hat{\tau}}_o = -\Psi \hat{\mathbf{S}}_\tau. \tag{38}$$

and the adaptive law for $\dot{\hat{\mathbf{a}}}$ is

$$\begin{aligned} \dot{\hat{\mathbf{a}}} &= -\Upsilon\left(Y_\Gamma^T S_\Gamma + Y_\tau^T \hat{\mathbf{S}}_\tau\right) \\ &\quad -\Upsilon\left(\Delta + \Sigma S_T\right)\left(W^T W\right)^{-1} W^T \varepsilon, \end{aligned} \tag{39}$$

where Δ is a positive diagonal adaptive gain matrix and

$$\Sigma = \frac{\gamma}{2}\begin{bmatrix} 0 & -R_{31} & 0 \\ -R_{31} & 0 & R_{32} \\ 0 & R_{32} & 0 \end{bmatrix}. \tag{40}$$

Since ε can be written in terms of $\tilde{\mathbf{a}}$ as given in equation (35), the adaptive algorithms in equations (38) and (39) can be verified to cancel out unwanted terms in equation (37), leaving the time derivative of the Lyapunov function candidate negative definite,

$$\dot{V} = -K_\Gamma S_\Gamma^2 - \hat{\mathbf{S}}_\tau^T K_\tau \hat{\mathbf{S}}_\tau - \tilde{\mathbf{a}}^T \Delta \tilde{\mathbf{a}} \leq 0. \tag{41}$$

To finalize the stability proof, the invariant set theorem is applied. The value of V keeps diminishing as long as S_Γ, $\hat{\mathbf{S}}_\tau$, and $\tilde{\mathbf{a}}$ are not all zeros. The fact that $\hat{\mathbf{S}}_\tau$ approaches zero does not immediately imply that the translational dynamics would be stabilized since when $\hat{\mathbf{S}}_\tau$ was defined in equation (27), it includes $\hat{\epsilon}_i$ and $\hat{\epsilon}_j$ rather than their true values. It is the inclusion of information from the predictor that results in the last term of equation (41) which ensures that the parameter estimates converge to their true values, and hence $\hat{\mathbf{S}}_\tau$ eventually approaches $\mathbf{S}_\tau$ and translational stability is satisfied along with altitude stability.

3.4 Optional Heading Control

Thus far, we have designed the adaptive tracking controller for the robot without a direct control over the yaw orientation—the heading—of the robot. This follows from the fact that, similar to quadrotors and some multi-rotor systems (Lee et al., 2010; Mellinger et al., 2012), the thrust produced by the

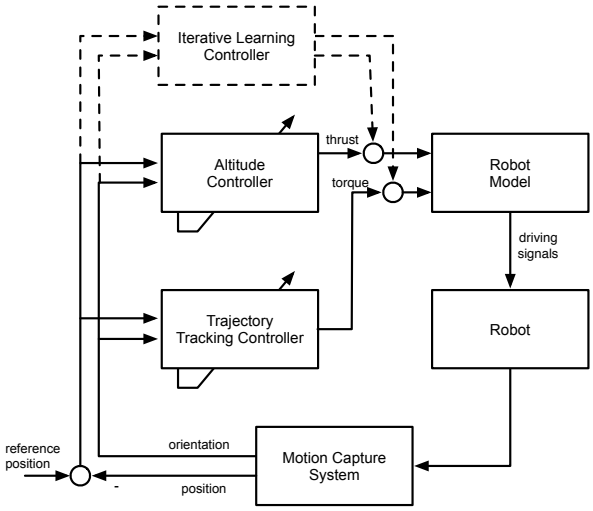

Figure 6: A block diagram illustrating the incorporation of the iterative learning control algorithm (in dashed lines) into the previous feedback control architecture of the robot.

flapping-wing robot is nominally aligned with the vertical or $\hat{k}$-axis of the robot, independent of its yaw orientation. The translational dynamics of the robot is unconnected to the robot's heading. This is reflected in the control law for the yaw torque of the robot as seen in equations (17) and (30). The yaw command only damps out the yaw rotational rate in order to prevent the robot from spinning uncontrollably. The proposed control scheme is sufficient for commanding the trajectory of the robot in three dimensional space.

In many applications, however, it may be preferable to have control over the heading of the aerial robots. Examples include vision-based and photographic applications (Duhamel et al., 2013), 3D printing using aerial robots (Hunt et al., 2014), and aerial manipulators (Orsag et al., 2013). It turns out that a simple modification to the composite variable defined in equation (15) or (27) would allow the robot to follow a commanded yaw orientation.

Here we present one possible implementation of the heading control not previously discussed in (Chirarattananon et al., 2014c). Consider the scenario where we would like the $\hat{\imath}$-axis of the robot to point to the direction of the vector $\hat{I}_h$ defined on the $\hat{I} - \hat{J}$ plane in the inertial frame. Let θ_h be the angle between the vector $\hat{I}_h$ and the projection of $\hat{\imath}$ on to the $\hat{I} - \hat{J}$ plane (with the sign defined by the right hand rule about the $\hat{K}$-axis). Then we can introduce an additional term $\lambda_h \theta_h$, for some positive constant λ_h to the composite variable $\mathbf{S}_\tau$ from equation (15) as

$$\mathbf{S}_\tau = \begin{bmatrix} -\mathbf{e} \cdot \hat{\jmath}/\Gamma & \mathbf{e} \cdot \hat{\imath}/\Gamma & \omega_k + \lambda_h \theta_h \end{bmatrix}^T. \tag{42}$$

As a consequence, two additional terms $\left(-\lambda_h \dot{\theta}_h - K_\tau \lambda_h \theta_h\right)$ arise in the third element of the control torque τ_c from the inclusion of $\lambda_h \theta_h$ in $\mathbf{S}_\tau$. The $\dot{\theta}_h$ term can be expressed as a function of the orientation and rotational rate of the robot. The full expression is omitted for brevity.

4 Iterative Learning Control for Perching on a Vertical Surface

In theory, the proposed adaptive tracking controller is capable of trajectory following flight thanks to the consideration of higher-order dynamics and inclusion of the feedforward term. The adaptive part is also capable of correcting for some model uncertainties. However, the limited bandwidth of the controller (for example, the actuation delay for torque generation is not taken into account in the

proposed controller) and the simplified dynamic and aerodynamic models mean that there remains some dynamics uncaptured by the model and the proposed controller. While these effects are not critical to hovering flights and basic maneuvers, they become significant in more aggressive flight such as landing on a vertical surface. In this section, we rely on the fact that systematic modeling errors on a particular trajectory are repeatable across multiple iterations and, hence, can be learned iteratively as preliminarily shown in (Chirarattananon et al., 2014b). The use of iterative learning control in MAV community was pioneered in (Lupashin et al., 2010; Mellinger et al., 2012) using simple formulations. The idea was developed further with an elaborate framework in (Schöllig and D'Andrea, 2009; Schoellig et al., 2012). As illustrated in (Mueller et al., 2012), implementation of iterative learning radically improved trajectory tracking performance of a quadrotor by acausally correcting for errors before they occur, essentially compensating for actuation delay and limited control bandwidth. For the task of perching a flapping-wing robot on a vertical surface—a highly dynamic maneuver well outside of the typical flight envelope for MAVs, the iterative learning control technique potentially permits us to successfully achieve this task, leveraging the fact that a sub-100-mg robot does not always fail upon crashing, whereas a 500-g quadrotors are more likely to suffer mechanical damage. As seen in (Mellinger et al., 2012), a quadrotor must be able to detect a failed perching attempt and recover.

To begin, we first constrain the landing dynamics to a two-dimensional plane. Based on the 2D dynamics, a landing trajectory is constructed and a method is proposed to find an additional control signal to compensate for the unaccounted dynamics learned from the previous iterations.

4.1 2D Model

To simplify the problem of landing on a vertical wall, we restrict our analysis to the two dimensional plane defined as $\hat{I} - \hat{K}$ in the inertial coordinate frame. The state variables of interest consists of the position and velocity of the robot along the $\hat{I}$ and $\hat{K}$ directions, the *tilt* angle (θ) of the robot defined as the angle between the $\hat{k}$ axis and the $\hat{K}$ axis as projected onto the $\hat{I} - \hat{K}$ plane (with $\hat{k}$ tilted in the $+\hat{I}$ direction defined as positive), and the normalized thrust (with the dimension of acceleration) produced by the robot (Γ). The vector containing state variables is denoted as $\mathbf{X}$ and is given in equation (43).

$$\mathbf{X} = \begin{bmatrix} X & \dot{X} & Z & \dot{Z} & \theta & \dot{\theta} & \Gamma \end{bmatrix}^T. \tag{43}$$

The dynamics of $\dot{\theta}$ are assumed to depend on the normalized projected torque $\bar{\tau}$ as $\ddot{\theta} = \bar{\tau}$ (the damping is neglected owing to the limited range of rotation and speed) and the thrust is related to the thrust input T by the first order dynamics, $\dot{\Gamma} = -\gamma(\Gamma - T)$, as given by equation (4). Therefore, T and $\bar{\tau}$ are regarded as two inputs to the system:

$$\mathbf{U} = \begin{bmatrix} T & \bar{\tau} \end{bmatrix}^T. \tag{44}$$

When constraining the dynamics of the robot on the vertical plane, the accelerations of the robot along and perpendicular to the $\hat{I}$ direction are functions of the tilt angle θ. That is, to maneuver laterally, the robot must tilt its body so that the thrust vector takes on a lateral component. Moreover, we assume a linear damping term in the lateral dynamics (this was neglected in the controller design earlier). The linear damping behavior (as opposed to the usual quadratic velocity dependence) arises from the domination of the drag force from the flapping wings over the body drag as reported in (Ristroph et al., 2013; Chirarattananon and Wood, 2013). As a consequence, the time derivative of the state vector $\mathbf{X}$ can be found from the following expression:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \dot{X} \\ \dot{Z} \\ \dot{\theta} \\ \Gamma \end{bmatrix} = \begin{bmatrix} -\Gamma \sin\theta - \xi \dot{X} \\ \Gamma \cos\theta - g \\ \bar{\tau} \\ -\gamma(\Gamma - T) \end{bmatrix}, \tag{45}$$

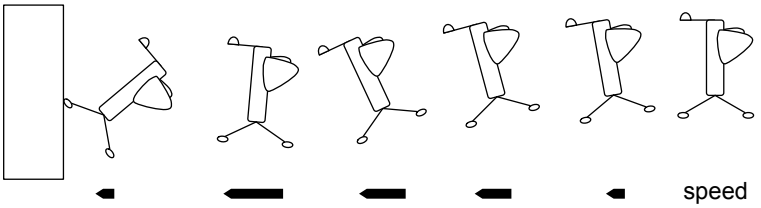

Figure 7: A schematic diagram demonstrating how the robot can perch on a vertical wall. Initially it needs to build up sufficient forward momentum to retain it when the robot rotates to the opposite direction at the moment of landing on the wall.

where ξ is a linear damping coefficient. This equation formulates a framework for the analysis of trajectory generation and ILC in sections 4.2 and 4.3.

4.2 Trajectory Generation

One requirement for perching on a vertical surface is to find a plausible trajectory that satisfies the constraints imposed by the dynamics of the robot as given in equation (45). To land on a vertical surface, there are some specifications on the trajectory, particularly at the end of the trajectory. The robot has to come into contact with the wall at steep tilt angle (preferably larger than 45°). Hence, it needs to generate a significant amount of torque at the very end of the trajectory. Since the torque generated is coupled with the thrust and the thrust vector is aligned with the body angle, this would decelerate the robot and potentially causes it to move away from the wall at the same time. Consequently, to perch on a wall, the robot has to carry sufficient forward momentum to ensure that the robot does not move backwards. A schematic diagram illustrating a perching trajectory is shown in figure 7.

More specifically, for perching robot on a vertical wall, we would like to design a trajectory with the desired terminal position and angle (X, Z, and θ). The two-dimensional dynamics of the robot as given in equation (45) is differentially flat (Van Nieuwstadt and Murray, 1997) which only two quantities, namely X and Z, can be chosen as flat outputs for torque and thrust as control inputs. Hence, to achieve required landing conditions that involve more than two parameters (for example, X, Z, θ, $\dot{X}$, and $\dot{\theta}$) a suitable trajectory must be generated. Mathematically, the trajectory generation problem can be reformulated as an optimization problem with a quadratic cost structure. Though, the true purpose of the proposed framework in this section is to find a feasible (or locally optimal) trajectory that satisfies the constraints rather than searching for the truly optimal trajectory. Here, the cost function J is comprised of an instantaneous cost $g(\cdot)$ and a terminal cost $h(\cdot)$. These can be expressed in term of the desired states as written in equation (46).

$$\begin{aligned} J &= \int_{t_i}^{t_f} g(\mathbf{X}, \mathbf{U})\, \mathrm{d}t + h\left(\mathbf{X}_{t_f}\right) \\ &= \int_{t_i}^{t_f} \begin{bmatrix} \mathbf{X} & \mathbf{U} \end{bmatrix} \Lambda_g \begin{bmatrix} \mathbf{X} & \mathbf{U} \end{bmatrix}^T - \lambda_g \dot{X}\, \mathrm{d}t \\ &\quad + \left(\mathbf{X}_{t_f} - \mathbf{X}_{t_f,ref}\right)^T \Lambda_{gt_f} \left(\mathbf{X}_{t_f} - \mathbf{X}_{t_f,ref}\right) - \lambda_{gt_f} \dot{X}_{t_f}, \end{aligned} \tag{46}$$

where $\mathbf{X}_{t_f}$ and $\mathbf{X}_{t_f,ref}$ denote the terminal state vector and the desired terminal state vector, Λ_g, Λ_{gt_f}, λ_g, and λ_{gt_f} are diagonal matrices and scalar constants, and t_i and t_f indicate initial and final times of the perching trajectory. The presence of Λ_g ensures that the robot always maintains a reasonable altitude and imposes soft constraints on the input signals. The term λ_g encourages the robot to build up a forward velocity. Similarly, the final cost enforced by Λ_{gt_f} and λ_{gt_f} influences the optimizer to search for a trajectory that ends at a desired landing position and orientation with some final forward velocity.

A common practice for such optimization problems is to confine the search space. In this circumstance, we limit the inputs to be polynomial functions of time as:

$$\Gamma(t) = \left(\sum_{i=0}^{i=N_\Gamma} a_i t^i\right)^2 \qquad \tau(t) = \Gamma(t) \sum_{i=0}^{i=N_\tau} b_i t^i, \tag{47}$$

here a_i and b_i are polynomial coefficients to be searched for. The use of polynomial structure has some benefits. For instance, by constraining b_0 to zero guarantees that a robot starting in the upright orientation will have $dX/dt = d^2X/dt^2 = d^3X/dt^3 = d^4X/dt^4 = 0$ and the trajectory is smooth at the beginning. Notice the quadratic structure of the thrust which is implemented ensures the thrust is always non-negative. Also, the existence of Γ in the expression of τ renders the model to be more realistic as the generation of torque is coupled with the generated thrust in the flapping-wing robot.

The proposed objective function in equation 46 optimizes for desired trajectory properties. The approach here is different from the method often taken in robotics or MAV communities (Mellinger and Kumar, 2011), where k^{th} derivative of position squared is minimized, and other specifications are treated as constraints. The primary motivation is that, for a perching task, the landing position, for example, is not constrained in practice as we have can choose the target position at an arbitrary distance from the starting position. On the other hand, we emphasize on reaching sufficient terminal angle and velocity by putting them in the objective function rather treating them as constraints. The disadvantage is the resultant trajectory may not be optimized for control inputs as seen in related work.

To find a locally optimal or feasible solution of equations (46) and (47), several tools could be employed. Most tools significantly benefit from a Jacobian—the derivative of the cost function J with respect to a parameter to optimize—if it can be directly calculated. Here we compute the gradient with the *Real-Time Recurrent Learning* (RTRL) method (Williams and Zipser, 1989; Hoburg and Tedrake, 2009).

To begin, we express the dynamics of the state vector as $\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U})$. For a parameter to optimize α (which, in this circumstance, could be a_i or b_i), we have

$$\begin{aligned} \frac{\partial}{\partial \alpha}\left(\dot{\mathbf{X}}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathbf{X}}{\partial \alpha}\right) &&= \frac{\partial f}{\partial \mathbf{X}}\frac{\partial \mathbf{X}}{\partial \alpha} + \frac{\partial f}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \alpha} \\ &= \frac{\mathrm{d}}{\mathrm{d}t}P &&= \frac{\partial f}{\partial \mathbf{X}}P + \frac{\partial f}{\partial \mathbf{U}}Q, \end{aligned} \tag{48}$$

where P and Q have been defined as $\partial \mathbf{X}/\partial \alpha$ and $\partial \mathbf{U}/\partial \alpha$ respectively. According to equation (46), the Jacobian $\partial J/\partial \alpha$ is then simply given as

$$\frac{\partial J}{\partial \alpha} = \int_{t_i}^{t_f} \left(\frac{\partial g}{\partial \mathbf{X}}P + \frac{\partial g}{\partial \mathbf{U}}Q\right) dt + \frac{\partial h}{\partial \mathbf{X}}P + \frac{\partial h}{\partial \mathbf{U}}Q. \tag{49}$$

It is straightforward to obtain $\partial f/\partial \mathbf{X}$ and $\partial f/\partial \mathbf{U}$ from equation (45). Thus, by integrating forward equation (48) to find P, the gradient $\partial J/\partial \alpha$ can be evaluated from equation (49). For the next step in the optimization, we simply use a gradient method to perform a gradient descent, the cost function is expanded to its second order approximation as

$$J(\alpha + \delta\alpha) \approx J(\alpha) + \left(\frac{\partial J}{\partial \alpha}\right)^T \delta\alpha + \frac{1}{2}\delta\alpha^T \left(\frac{\partial^2 J}{\partial \alpha^2}\right)\delta\alpha. \tag{50}$$

Inspired by the Gauss-Newton algorithm, here we opt to approximate the Hessian by taking a derivative of equation (49) with respect to α again but neglecting the $\partial^2 P/\partial \alpha^2$ and $\partial^2 Q/\partial \alpha^2$ terms. This reduces the complexity and the computational time. It follows that we can then solve equation (50) for an incremental change in α:

$$\delta\alpha = -\eta \left(\frac{\partial^2 J}{\partial \alpha^2}\right)^{-1} \frac{\partial J}{\partial \alpha}.$$

The step size parameter η keeps the update gradual, improving the stability. Performance could also be tweaked by altering the cost parameters and the reference state.

4.3 Iterative Learning Control

In perching experiments, we command the robot to follow the pre-generated trajectory. After each flight iteration, the recorded trajectory is analyzed for the iterative learning controller to compute a set of corrective commands as inputs for the next flight so that the flight trajectory will eventually converge to the reference trajectory. The major distinction between a closed-loop controller and the iterative learning controller is that the former does not primarily learn from prior experiences (except for the adaptive part; nevertheless, the adaptive algorithm is not time or trajectory specific). The iterative learning controller, on the other hand, relies on repetition to cope with repetitive disturbances or systematic errors in the modeling.

The formulation of the iterative control below is inspired from (Schöllig and D'Andrea, 2009; Schoellig et al., 2012). Herein, the scheme iteratively updates low-level commands (i.e., torque and thrust) to minimize the difference between the planned trajectory and the observed trajectory. In fact, more recent work suggests that learning performance of quadrotors performing a slalom trajectory is significantly improved when the learning is applied to reference position instead of low-level commands (Mueller et al., 2012). However, such scheme is not directly applicable to our situation, where positions and body angle are dynamically coupled and cannot be commanded independently as explained in section (4.2).

To begin, we consider the dynamics of an entire trajectory using a lifted representation, similar to the approach taken in (Schöllig and D'Andrea, 2009; Schoellig et al., 2012). That is, we discretize and consolidate the state vectors and the inputs into long vectors as given by the following:

$$\begin{aligned} \mathbf{X}^* &= \begin{bmatrix} \mathbf{X}(t_1) & \mathbf{X}(t_2) & \dots & \mathbf{X}(t_N) \end{bmatrix}^T \\ \mathbf{U}^* &= \begin{bmatrix} \mathbf{U}(t_1) & \mathbf{U}(t_2) & \dots & \mathbf{U}(t_N) \end{bmatrix}^T. \end{aligned} \tag{51}$$

The model of the lifted dynamics is given by a function $f^*(\cdot)$. If we assume a perfect model (which can be derived from the two-dimensional dynamic in equation (45)), then the reference trajectory $\mathbf{X}^*_{ref}$ can be realized using a feedforward input $\mathbf{U}^*_{ff}$ as

$$\dot{\mathbf{X}}^*_{ref} = f^*\left(\mathbf{U}^*_{ff}\right). \tag{52}$$

In practice, it is not anticipated that the model will be perfect. Instead of attempting to find a better model, we assume that the proposed dynamic model is sufficiently accurate, but the input into the system can be regarded as a combination of the command input $\mathbf{U}^*_c$ and the unknown disturbance input $\mathbf{U}^*_d$ such that $\mathbf{U}^* = \mathbf{U}^*_c - \mathbf{U}^*_d$, and the ultimate goal of the algorithm is to find the estimate of the unknown disturbance input $\hat{\mathbf{U}}^*_d$. When we have an accurate estimate of the unknown disturbance input, we can achieve the reference trajectory by using the command input $\mathbf{U}^*_c = \mathbf{U}^*_{ff} + \hat{\mathbf{U}}^*_d$ as given below:

$$\begin{aligned} \dot{\mathbf{X}}^* &= f^*\left(\mathbf{U}^*\right) \\ &= f^*\left(\mathbf{U}^*_c - \mathbf{U}^*_d\right) \\ &= f^*\left(\mathbf{U}^*_{ff} + \hat{\mathbf{U}}^*_d - \mathbf{U}^*_d\right). \end{aligned} \tag{53}$$

The calculation of the unknown disturbance input is carried out in an iterative fashion. The current estimate is implemented for one flight experiment, and the resultant trajectory is analyzed for a new estimate. To illustrate this procedure in more details, we first define the estimation error at iteration j as

$$\tilde{\mathbf{U}}^*_{d,j} = \hat{\mathbf{U}}^*_{d,j} - \mathbf{U}^*_d. \tag{54}$$

Then the lifted dynamics at the j^{th} iteration can be expanded about the ideal operating point $\mathbf{U}^*_{ff}$,

$$\dot{\mathbf{X}}^*_j \quad = f^*\left(\mathbf{U}^*_{ff} + \tilde{\mathbf{U}}^*_{d,j}\right) \approx \quad f^*\left(\mathbf{U}^*_{ff}\right) + \left(\frac{\mathrm{d}}{\mathrm{d}\mathbf{U}^*} f^*\Big|_{\mathbf{U}^*_{ff}}\right) \cdot \tilde{\mathbf{U}}^*_{d,j}. \tag{55}$$

The vector $\dot{\mathbf{X}}^*_j$ on the left hand side of equation (55) can be obtained by post-processing the recorded trajectory. Let an error vector δ_j denotes the difference between the measured dynamics $\dot{\mathbf{X}}^*_j$ and the reference dynamics $\dot{\mathbf{X}}^*_{ref}$, equation (55) allows us to write δ_j as a function of the estimation error $\tilde{\mathbf{U}}^*_{d,j}$ as

$$\delta_j = \dot{\mathbf{X}}^*_j - \dot{\mathbf{X}}^*_{ref} = \left(\frac{\mathrm{d}}{\mathrm{d}\mathbf{U}^*} f^*\Big|_{\mathbf{U}^*_{ff}}\right) \tilde{\mathbf{U}}^*_{d,j} = F\tilde{\mathbf{U}}^*_{d,j}. \tag{56}$$

It can be seen that matrix F is only a function of the reference trajectory and independent of the current trajectory, so it only needs to be computed once. At this point, we propose an update law for the estimate of $\mathbf{U}^*_d$ for the next iteration:

$$\hat{\mathbf{U}}^*_{d,j+1} = \hat{\mathbf{U}}^*_{d,j} - F^T \kappa \delta_j, \tag{57}$$

for some positive diagonal matrix κ. It follows that we could express the l_2-norm of the error vector from two consecutive iterations as

$$\delta^T_{j+1}\delta_{j+1} = \delta^T_j \left(I - FF^T\kappa\right)^T \left(I - FF^T\kappa\right)\delta_j. \tag{58}$$

Since FF^T is always positive definite, for a sufficiently small κ, the norm of the error vector always decreases. In other words, we have

$$\delta^T_{j+1}\delta_{j+1} \leq \sigma \delta^T_j \delta_j \qquad \text{for} \quad \exists\, 0 \leq \sigma < 1. \tag{59}$$

In the implementation, F can be computed by representing the robot's dynamics given by equation (45) along the reference trajectory using a *Linear Time Varying* (LTV) configuration.

$$\dot{\mathbf{X}}(t) = A(t)\mathbf{X}(t) + B(t)\mathbf{U}(t), \tag{60}$$

and the matrix F is simply given by

$$\begin{aligned} F \quad &= \quad \begin{bmatrix} \frac{\partial f_{t_1}}{\partial \mathbf{U}_1} & \cdots & \frac{\partial f_{t_1}}{\partial \mathbf{U}_{t_N}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{t_N}}{\partial \mathbf{U}_1} & \cdots & \frac{\partial f_{t_N}}{\partial \mathbf{U}_{t_N}} \end{bmatrix} \\ &= \quad \begin{bmatrix} F_{(1,1)} & \cdots & F_{(1,N)} \\ \vdots & \ddots & \vdots \\ F_{(N,1)} & \cdots & F_{(N,N)} \end{bmatrix}, \end{aligned} \tag{61}$$

where the elements can be shown to be

$$F_{(\mathrm{m,n})} \quad = \quad \begin{cases} 0 & \text{if } m < n \\ B(t_n) & \text{if } m = n \\ B(t_n) \sum_{j=n+1}^{j=m} \left(\prod_{i=j}^{i=m} A(t_i)dt\right) & \text{if } m > n. \end{cases} \tag{62}$$

Once computed, F is used to yield the new estimate of $\mathbf{U}^*_d$ according to equation (57). In the case that the whole trajectory is unavailable (for instance, when the robot crashes before completing the entire trajectory), F can be truncated to match the length of $\mathbf{e}$, and the partial update of $\hat{\mathbf{U}}^*_d$ is obtained.

4.4 Consideration of Initial Conditions

In the previous section, it is shown that the norm of the error vector should gradually decrease after each iteration. However, the presented analysis is based on the assumption that each trajectory always starts with the same initial conditions such that minimizing the error in $\dot{\mathbf{X}}^*$ is sufficient to bring the actual trajectory close to the reference trajectory. In experiments, where the initial condition involves a robot hovering at the setpoint, the assumed initial condition is approximately satisfied for the tilt angle and the initial velocity, but not for the position. In prior work, it was shown that the RMS of the position error during hovering flights was just below one centimeter (Chirarattananon et al., 2014a). This means that even if the error vector $\mathbf{e}$ becomes zero, the robot could end up attempting to perch up to one centimeter away from the wall.

To avoid a situation similar to the one mentioned above, we allow the robot to initialize a perching attempt only when the attitude dynamics are stable as measured by the weighted l_2 norm of the composite variable $\hat{\mathbf{S}}_\tau$ defined in equation (27). Furthermore, the starting position of the robot is taken into consideration—the trajectory tracking for perching attempt would not start from the beginning of the pre-planned trajectory. To elaborate, suppose the robot starts perching at time t_0 with $X(t_0) = X_0$ and the reference trajectory is defined for $\mathbf{X}_{ref}(t)$ for $t_i \leq t \leq t_f$, we seek to find t_0' that satisfies the equation

$$X_0 = X_{ref}(t_0') + \int_{t_0'}^{t_f} \dot{X}_{ref}(t)dt, \tag{63}$$

and command the robot to follow the trajectory from $\mathbf{X}_{ref}(t_0')$ to $\mathbf{X}_{ref}(t_f)$. The idea is that, to a first order approximation, the extra distance at the beginning would eventually be cancelled out by the deficit in the initial velocity. As a result, the robot's trajectory will not match the reference at the beginning ($X(t_0) \neq X_{ref}(t_0')$), but the discrepancy should theoretically diminish towards the end.

4.5 Implementation in Three Dimensions

In experiments, the tracking controller proposed in section 3 only takes into consideration the direction of the $\hat{k}$ axis of the robot and does not directly control the heading direction (yaw orientation) of the robot. In other words, the robot would need to perform both pitch and roll maneuvers to realize the reference trajectory depending on its current orientation. In the case that the robot follows a trajectory to perch on a wall in the positive $\hat{I}$ direction, the reference torque input $\bar{\tau}$ points along a negative $\hat{J}$ direction. With the knowledge of the current orientation of the robot and the assumption that the moment of inertia along the pitch and roll axes are approximately equal (the difference is approximately 5% Ma et al. (2013)), it is possible to find a combination of pitch and roll torques in the body frame that points in the negative $\hat{J}$ direction with the specified magnitude.

5 Experimental Results

5.1 Apparatus and Experimental Setup

The current prototype of the flapping-wing robot is not fitted with sensors, power source, or controller units. Without such components, the robot is operated in a laboratory environment and depends on an external motion capture system to provide position and orientation feedback. Eight *VICON* cameras running at 500Hz, covering the tracking volume of $0.3 \times 0.3 \times 0.3$m, track the position of four retroreflective markers placed on the robot and triangulate the pose of the robot. Control computation is carried out on a computer running an xPC Target (*MathWorks*) environment at the rate of 10kHz. Power is supplied to the robot via a bundle of four 51-gauge copper wires from a high voltage amplifier that receives a command from the xPC Target with a digital-to-analog converter. The effect of the wire tether is not taken into consideration due to its unpredictable nature. However, simple calculations suggest its contribution should not affect the flight dynamics significantly and could be taken care of

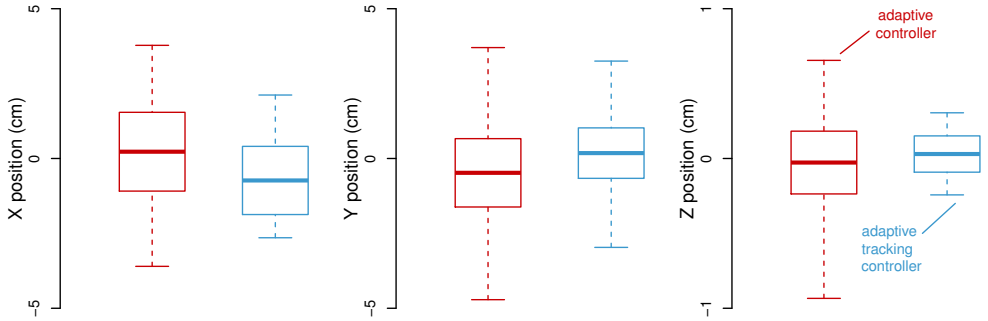

Figure 8: Box plots showing the averages and the standard deviations in positions of the robots from several flights using the controller from Chirarattananon et al. (2014a) (115 seconds or 13,800 wingbeats) and the proposed adaptive tracking controller (23.5 seconds or 3,820 wingbeats).

by the controller (Fuller et al., 2014). Direct measurements of the robot's velocity and rotation rates are not available, so they were substituted by their filtered derivative representations.

5.2 Hovering Flight

To verify that the proposed adaptive tracking controller is capable of stabilizing the robot, we first commanded the robot to hover around a stationary setpoint. Prior to flight experiments, the robot had to be verified for its flight capability. This involves the characterization of the robot's flapping amplitude at various operating frequencies. After validating that the robot possesses sufficiently large and symmetrical flapping amplitudes on both wings, the robot needed to be trimmed for flight. The trimming process starts with short, unstable open-loop flights that last less than 0.4 s each to determine a set of driving signals that minimize the residual torque exhibited by the robot due to unavoidable mechanical asymmetries. These were followed by a closed-loop trimming process, in which the adaptive part of the controller corrected for torque offsets and other parameters further until they converged.

In the absence of mechanical fatigue, the robot mostly stayed close to the setpoint with the position errors smaller than one body length (see Extension 1). As shown in figure 8, the *Root Mean Square* (RMS) errors in position of example hovering flights were found to be comparable to those from the previous adaptive controller in (Chirarattananon et al., 2014a). The box plots illustrating the averages and the RMS errors are shown in figure 8. The results shown here are consistent with the expectation that the tracking controller may not show any significant improvement for stationary setpoints. Theoretically, the robot is able to stay aloft indefinitely without crashing. In practice, we attempt to minimize the total operating time to prevent mechanical fatigue as the lifetime of the robot is approximately a couple minutes (Malka et al., 2014).

5.3 Trajectory Following

To demonstrate the tracking ability of the proposed single-loop controller, we commanded our flapping-wing robot to follow a smooth trajectory generated by a simplified version of the algorithm described in section 4.2. The aim here is to compare the trajectory tracking performance of the previous adaptive controller from (Chirarattananon et al., 2014a) and the proposed tracking controller. The trajectory in this experiment was generated from three setpoints. The robot was initially set to hover at the starting position, navigate to the middle point at the specified time, and come to stop at the final waypoint in 1.5 seconds. First to fourth order derivatives of the position at the starting point and end point were set to zero. The generated trajectory, along with its respective velocity, acceleration, jerk, and snap

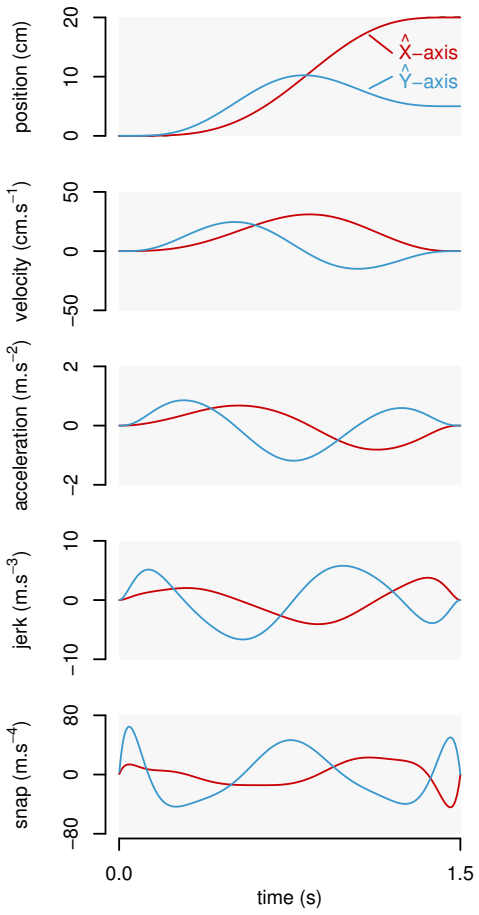

Figure 9: The trajectory generated for the experiment. Its derivatives are comparable to a circular trajectory with the radius of 5.0 cm and the period of 1.25 s.

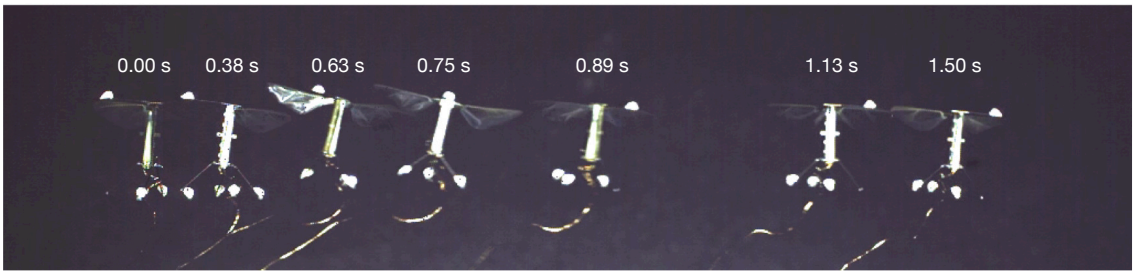

Figure 10: A composite image reconstructed from a video footage recorded at 240 frames per second (see Extension 1). The image shows the positions of the robot at various timestamps on the 1.5-second trajectory as seen on the $\hat{I} - \hat{K}$ plane.

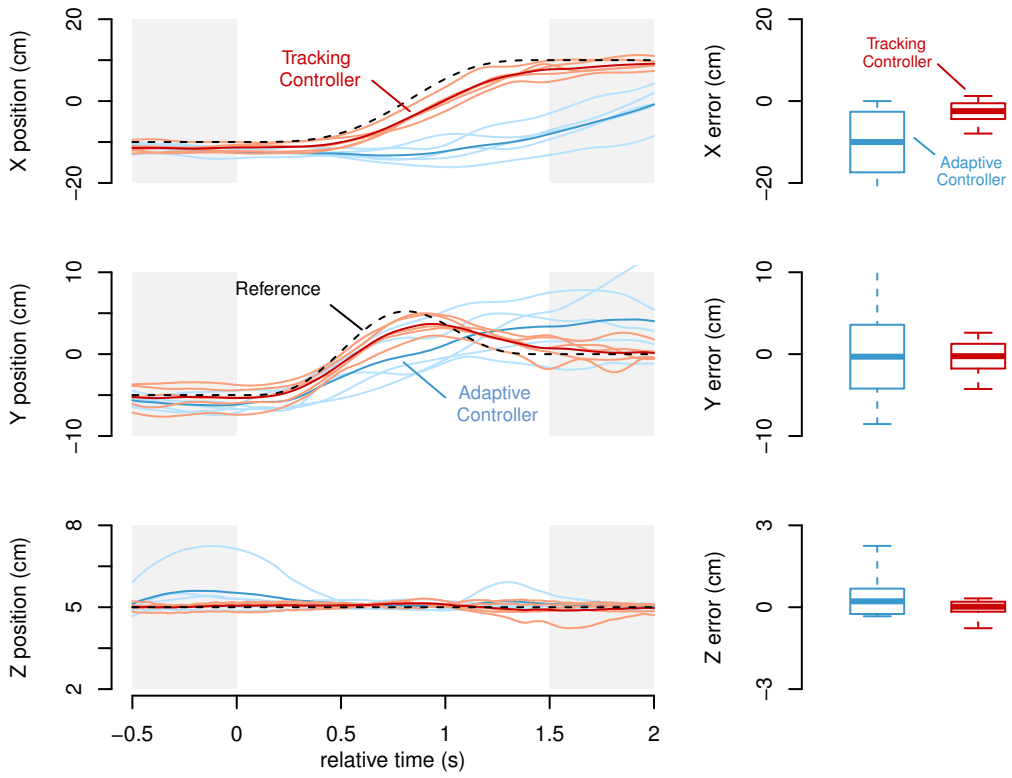

Figure 11: The trajectory following flights obtained from using the adaptive controller from Chiraratananon et al. (2014a) and the proposed single-loop controller. Left plots show the position of the robot with respect to the inertial frame, compared with the reference trajectory. Solid color lines represent the average trajectories from each controller, while light-colored lines represent individual flights. Box plots showing the average errors and RMS errors along three directions are shown on the right.

are plotted in figure 9. This trajectory was selected to represent more general trajectories with similar properties. For instance, a circular trajectory with the radius of 5.0 cm and the period of 1.25 s appears to have comparable velocity, acceleration, jerk and snap. The generated trajectory is chosen for the experiments as their first to fourth order derivatives are zero according to the imposed constraints. Moreover, the tether wire could severely interfere with the flight dynamics in the case of prolonged cyclic trajectories. We performed trajectory following experiments on this generated trajectory. Five flights were obtained from each controller.

Figure 10 and Extension 1 show an example of the robot following the pre-defined trajectory. Ten flight trajectories recorded from the motion capture system achieved from both controllers (five from each) are illustrated in figure 11. The RMS position errors for these flights are also plotted alongside. It can be seen that the proposed tracking controller offers significant improvements over the previous adaptive controller as the RMS errors in position are markedly smaller in all directions. This verifies that the feedforward component in the tracking controller enables the robot to follow more aggressive trajectories with greater precision.

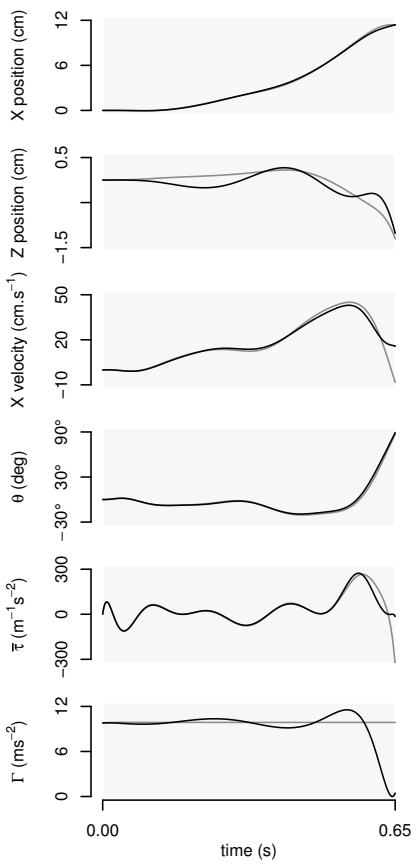

Figure 12: A candidate trajectory for perching. The grey lines show a parametrized version of a hand-designed trajectory. Black lines represent the final trajectory after the optimization.

5.4 Generation of Perching Trajectory

Initially, the perching trajectory is crudely hand-designed with a target distance near 12 cm from the starting position and the trajectory duration of 0.65 s. This was then parametrized by the polynomial functions as formulated in equation (47) with $N_\Gamma = 8$ and $N_\tau = 12$. The parametrized trajectory is shown as grey lines in figure 12. Next, the trajectory is optimized using the strategy outlined in section 4.2 and the results are shown as black lines in figure 12. It can be seen that the most evident difference is in the terminal velocity which is, in fact, undesirably negative before the optimization. The optimized thrust is smaller at the end of the trajectory at the point when the robot is approaching to wall. This is in order to reduce the amount of deceleration and preserve the forward momentum. The resultant trajectory is 11.4 cm long with the projected terminal tilt angle close to 90°.

5.5 Landing Mechanism

The magnetic wall for perching experiments was constructed from a flexible magnetic sheet manufactured from Ferrite bonded with synthetic rubber. This type of magnet is generally classified to have very low magnetic pull. The maximum pull (defined as the maximum pull force to a thick flat steel plate in an ideal laboratory condition) is 1,100 kg·m^{-2}. On the robot, four discs of 6 mil (0.15 mm) steel shims, each with a diameter of 2 mm, were attached to the landing gear. This brought up the

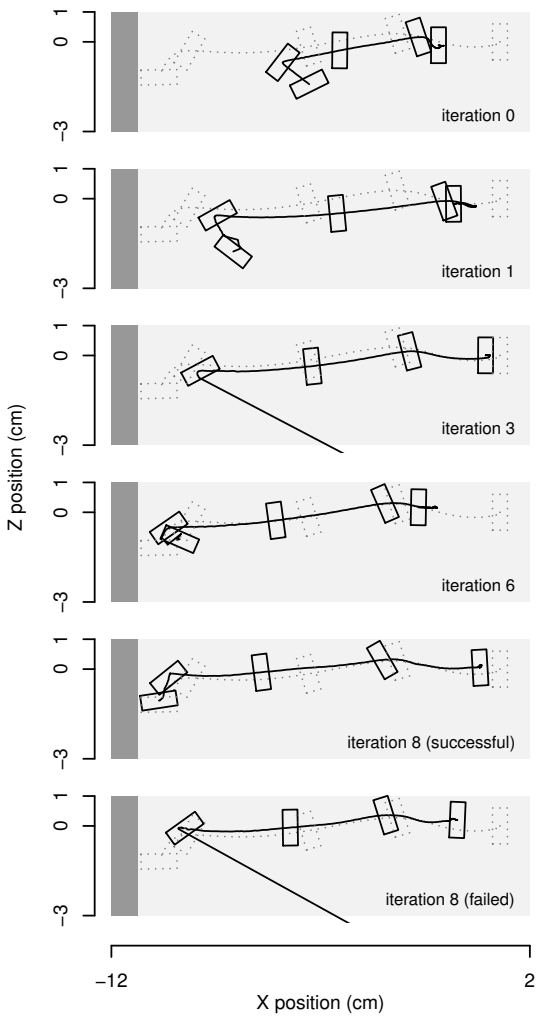

Figure 13: Diagrams showing the reconstruction of perching trajectories at various iterations. The plots show the projection on the $\hat{I}$-$\hat{K}$ plane. Reference trajectory is drawn in dashed lines.

total weight of the robot from 80 mg to 100 mg. To achieve stable flight with additional components, the controller had to be tuned to account for the change in mass and moment of inertia.

We experimentally found that one disc of steel shims could hold a weight of up to 230mg. In an ideal case—neglecting a force required to counter any kinetic energy, a simple calculation reveals that one disc must be able to support at least ≈ 60 mg to hold the robot to the wall in a static condition. Taking other factors into consideration, the strength of the magnet and the size of the steel shims offer appropriate attraction for the landing task. Furthermore, the field of a magnetic sheet is expected to decay faster than that of a magnetic dipole, which is a cubic function of distance, or r^{-3} (Jackson and Jackson, 1962). As a result, the contribution of the magnetic force should be negligible when the robot is not in contact with the magnetic wall.

5.6 Perching Experiments

Prior to the perching experiments, the robot had be to trimmed and tuned for hovering flights in the same way as it was for the trajectory following experiments.

In the perching attempts, the complication arises as it is impossible to ensure that the robot would be precisely at the origin where the perching trajectory begins. As a workaround, the consideration of initial conditions as outlined in section 4.4 takes into account the starting position of the robot and ensure that the robot would start its trajectory on or somewhere in front of the prescribed perching trajectory's starting point. Because of the one-centimeter uncertainty in the actual starting position of the robot, we defined the start of the trajectory to be at -1.0 cm from the hovering setpoint of the robot at 0.0 cm. The target vertical wall was placed at 10.4 cm (not 11.4 cm) from the hovering setpoint. The controller allowed the perching attempt to begin only when the robot is less than one centimeter away from the hovering setpoint (-1.0 cm $\leq X \leq$ 1.0 cm). Thus, given the 11.4-cm prescribed trajectory, the robot will start the perching attempt in front of the trajectory's starting point.

The first perching flight (iteration 0) was executed using the adaptive tracking controller alone without any correction from the ILC algorithm. The reconstruction of this iteration's trajectory (projected onto the $\hat{I} - \hat{K}$ plane) is illustrated in figure 14, and the trajectory-versus-time plots are also shown in figure 14. Possibly due to the lack of sufficient fidelity in the model, particularly for fast dynamics, the robot only reached the distance of 6.3 cm and the tilt angle of 41°before falling out of the air.

Using the recorded trajectory from iteration 0 and the proposed ILC scheme, the first estimate of $\mathbf{U}_d^*$ was calculated and incorporated. In iteration 1, the robot could get closer to the wall, i.e., it achieved the distance of 8.5 cm before losing all forward momentum, at which point the robot had the tilt angle of 78°. The corresponding trajectory and some subsequent trajectories are presented in figure 13-14. It can be seen that, over multiple iterations, the robot went increasingly closer to the wall before losing the forward momentum, primarily to reduce the terminal cost of the final position along the $\hat{I}$ axis as defined in the objective function in equation (46). Note that the term *iteration* here refers to the version of $\hat{\mathbf{U}}_d$. There could be many flight attempts in each iteration as some flights might be discarded and not used for updating the estimate $\hat{\mathbf{U}}_d$ owing to possible deteriorated performance explained below.

Due to the nature of the experiment and the delicate character of the robot, it is impossible to ensure that the physical properties of the robot remain unaltered over the course of several flights. Sources of the uncertainty include mechanical fatigue of the wing hinges, wings, and actuators (which unfortunately do occur in the timescale of the experiments), structural damage from crashing to the ground, and electrical connection failure due to impact and wire fatigue. After subsequent repairs to the robot, open-loop and closed-loop flight trimming experiments must be repeated—allowing the adaptive part of the controller to trim the robot for a good operating condition.

On this occasion, both wing hinges on the robot mechanically failed after six iterations just as the robot was close to achieving a successful vertical landing. The authors replaced both wing hinges and wings and carried out the trimming process to achieve a steady hover again. It is noted that the operating point of the repaired robot was different than that of the robot prior to wing hinge failure.

Using the same command resulted in slightly different trajectories compared with those obtained before wing hinge failure. Nevertheless, we carried on applying the ILC algorithm and updated the estimate of $\mathbf{U}_d^*$ from the previous iteration instead of resetting the iterative process. After two subsequent iterations (iteration 8), the robot successfully landed on the vertical wall. Out of five landing attempts using the same estimate of $\mathbf{U}_d^*$ (hence, all five flights are classified as iteration 8), the robot managed to land on the vertical surface three times, whereas the other two flights the robot failed to attach to the wall. One example of a successful attempt is demonstrated in figure 13 and 15(a)-(b), and Extension 1. In this attempt, the robot first contacted the wall when the tilt angle was around 45°. A failed perching is also presented in figure 15(c). The recorded trajectory reveals that the robot missed the target setpoint by 1-2mm. This also suggests that the effect of the magnetic force is minimal when the robot is not attached to the wall. More quantitative information of the successful and failed perching trajectories using the same estimate of $\mathbf{U}_d^*$ can be found on the right hand side of figure 14. It confirms that the major difference between the successful and failed perching trajectories is the final position of the robot along the $\hat{I}$-axis. Notice that, for an unknown reason, in one of the successful flight, the robot dropped a few centimeter from the reference trajectory. We believe that a

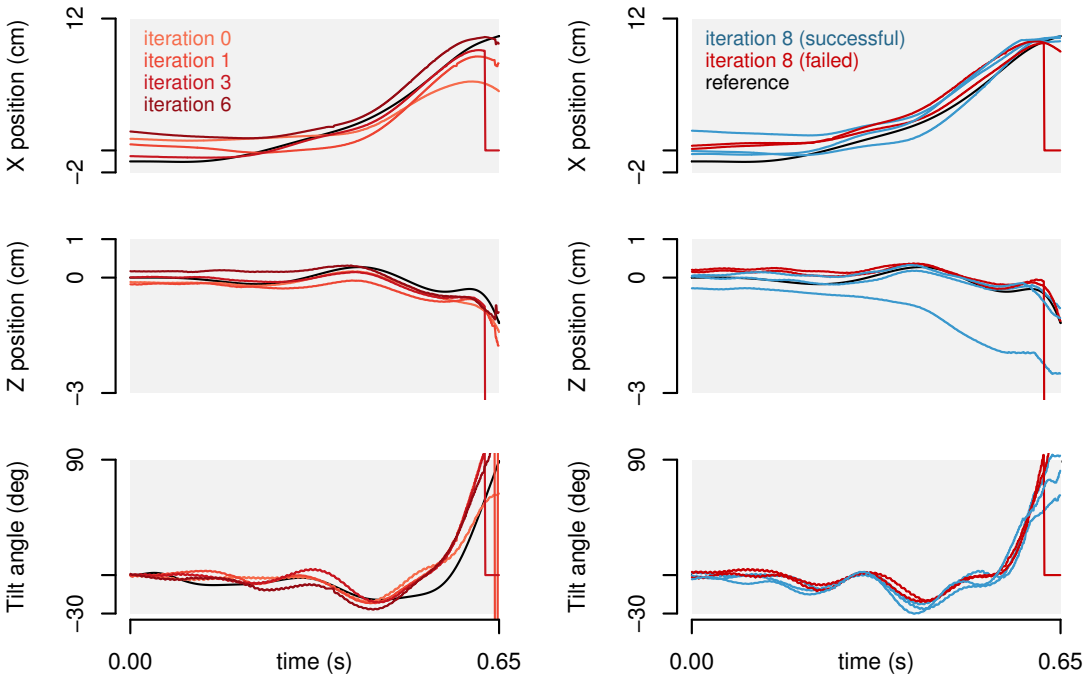

Figure 14: Position and tilt angle of the robot in several vertical landing attempts. (left) The robot failed to gain enough forward displacement and momentum in early iterations, but improvement is observed after multiple iterations. (right) Three successful and two failed perching trajectories obtained from executing identical ILC command. One successful attempt was achieved with a large deviation from the reference height (Z position). The jumps near the end of the trajectory seen in the plots are due to losses in motion tracking.

crash prior to that flight may have slightly altered the thrust generation ability of the robot as can be observed that the robot did not fully reach the altitude setpoint before beginning the perching attempt. Fortunately, the precise height of the robot is not critical to the perching dynamics, as opposed to the lateral position and the tilt angle.

6 Conclusion and Discussion

We have shown that a millimeter-scale flapping-wing robot is capable of performing an aggressive aerial maneuver—perching on a vertical wall. From a control perspective, such a maneuver differs considerably from hover or slow maneuvers that have been demonstrated before. To land on a vertical surface, we first re-designed the adaptive tracking flight controller and constructed a nominal perching trajectory via an optimization method that assumes a simplified dynamics model in two dimensions. For precise control, we opted to implement an iterative learning control algorithm in addition to the proposed adaptive flight controller. This learning algorithm computed an updated feedforward command for the robot after each perching attempt, in order to improve the trajectory tracking performance in an iterative fashion. Due to stringent payload constraints and scalability challenges, magnetic force was utilized as the wall attachment mechanism, enabling the robot to perch on a vertical magnetic surface. The magnetic force is only sufficient to hold the robot when it makes surface contact and has an insignificant effect on broadening the flight trajectory envelope for successful wall perches. It is shown that after eight iterations, the proposed control strategies enabled the robot to successfully

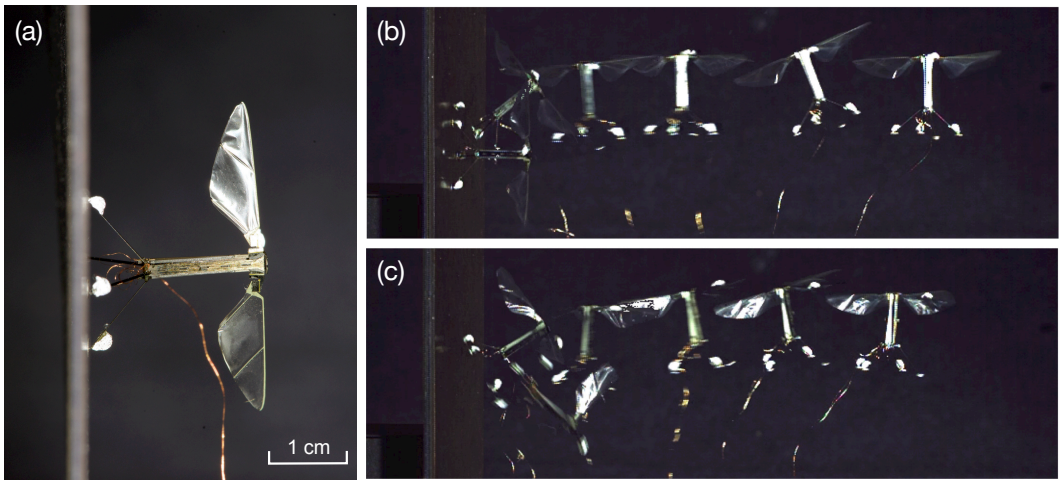

Figure 15: (a) An image showing the flapping-wing robot attached to a magnetic wall via the aid of steel shims adhered to the base of the landing gear. (b) A composite image constructed from a video footage demonstrating a successful perching flight after eight iterations of learning (see Extension 1). (c) A composite image constructed from a video footage showing a failed perching flight obtained from the same set of commands as the successful flight.

land on a vertical surface as desired.

Without the adaptive tracking controller and the learning algorithm used in this paper, the previous controller is unable to command the robot to accurately follow the prescribed trajectory. Understandably, like most robotic systems, the dynamic model assumed by the general controller is limited in bandwidth, only captures slow system dynamics, and lacks fidelity for unmodeled high frequency components required to perform an aggressive maneuver. Yet, this nominal model is sufficiently accurate to form a basis for the search for a feasible trajectory, and the learning process to compensate for the inaccuracies, by repeating the trajectory following attempts, taking into consideration only the first-order approximations of the dynamics. We show that the robot is able to land on a vertical surface—a task that requires millimeter-accuracy for a robot in which the position error of its stationary hovering flight is in the range of one centimeter (Chirarattananon et al., 2014a).

The utilization of magnetic force is convenient for a robot at this scale because its implementation adds minimal payload. Unfortunately, while it is sufficient to enable demonstrations of aggressive trajectory-following, it does not allow the robot to autonomously takeoff from the wall's surface. A more finely tuned or altogether different attachment mechanism is required. It is nontrivial to construct a detachable attachment mechanism similar to those seen in (Hawkes et al., 2013; Mengüç et al., 2014), let alone at this much smaller scale. However, we predict that once a more elaborate attachment mechanism is developed, the control strategy illustrated in this paper is suitable for direct application.

6.1 Convergence and Robustness of the Adaptive Controller

As stated in section 5.2, the robot has to undergo a trimming process involving several closed-loop flight for the parameters to adapt before stable hovering is achieved. The pitch and roll torque offset values are highly sensitive to how the robot is constructed and vary widely between prototypes. To demonstrate how quickly those values converge and how they affect the flight performance, we provide a plot of flight-averaged RMS error in positions of nine consecutive trimming flights in figure 16. In this circumstance, the flight-capable robot was repaired due to fatigue of wing hinges and wings and needed new parameter estimates to account for changes in physical properties. The initial non-zero

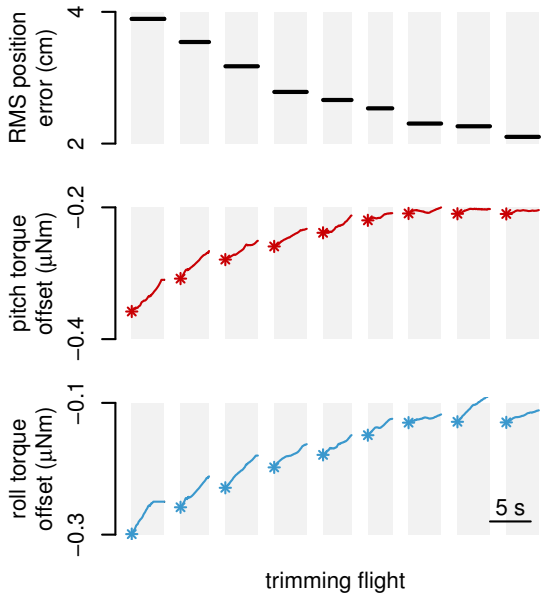

Figure 16: Plots illustrating the RMS position errors of the robot reduced over continuously over nine closed-loop trimming flights as the adaptive algorithm adjusted the estimates of pitch and roll torque offsets. White space indicates breaks between flights and asterisks mark the beginning of each flight. The initial estimates at the beginning of each flight were manually set according to the previous estimates.

estimates were chosen according to the open-loop trimming process. The values after convergence are generally different to those before the repair process. As seen in figure 16, the estimates of pitch and roll torques are plotted to show that the RMS errors reduced noticeably as the torque offset estimates from the adaptive controller converged. This corresponds to the robot becoming more stable over nine flights, totaling the period of over 30 s. The RMS errors decreased to from ~ 4 cm to an acceptable value of ~ 2 cm after estimates of pitch and roll offsets changed by 0.2 μNm.

According to equation (25), we can theoretically predict how significant the position error would be given the errors in torque offset estimates. We consistently obtain stable flight after parameter convergence with $K_\tau \approx 70 \times 10^{-9}$ kg.m^2.s^{-1}, $\lambda_1 \approx 8.5$ s^{-1}, $\lambda_2 \approx 200$ s^{-2}, and $\lambda_3 \approx 1,150$ s^{-3}. Near the hovering condition, where $\Gamma \approx g$, equation (25) predicts that the position error in steady state is ~ 2.4 cm for a torque offset error of 0.2 μNm. This is in accordance with the two-centimeter reduction in error after estimates of pitch and roll offsets changed by 0.2 μNm as evidenced in figure 16. We believe the RMS error of two centimeters after parameter convergence stems from various factors, such as unmodeled unsteady aerodynamics, latency, wire tether, and other disturbances. Furthermore, the 0.2 μNm error in torque offset estimate is translated to approximately 20° error in pitch or roll angle in the transient state—close to the amount that could destabilize the robot in flight. This explains why prior to closed-loop experiments, the open-loop trimming procedure is necessary to determine the torque offsets to with in $0.2 - 0.3$ μNm of their true values. To put these numbers into perspective, the robot is capable of producing a maximum torque in the order of one μNm and the inherent torque offset could be similar in magnitude, while less than ± 0.3 μNm is produced in a regular hovering flight. Given the small moment of inertia, a 0.1-μNm torque would result in the angular acceleration of about 50 rad.s^{-1}.

Further inspection of equation (25) suggests that error in the estimates of ϵ_x and ϵ_y do not affect the flight stability as much as the torque offsets as they influence the convergence of $\hat{\mathbf{S}}_\tau$ through the

term $Y_\tau \tilde{\mathbf{a}}$. For $\omega_k \sim 5$ rad.s^{-1}, and $\epsilon \approx 0.2$ rad or 10°, the error in position in steady state is expected to be 2 mm, an order of magnitude smaller than those caused by the unknown torque offsets.

Alternative to the current adaptive scheme where constant adaptive gains are adopted, optimal gain based on optimal filtering techniques (Brian and Moore, 2005) could be employed. Since the performance of the current scheme is adequate, we chose to keep a simple adaptive scheme, while the optimal approach could be employed for more complicated unknown disturbances in future work.

6.2 Importance of Experimental Learning for Aggressive Maneuvers

The primary role of the learning flights was to compensate for unmodeled dynamics, including actuator dynamics and aerodynamic effects. We demonstrated that the proposed iterative learning control method allowed the robot to achieve the perching task after eight learning iterations. However, the method suffers from a limitation that the learned model is specific to a particular trajectory and cannot be transferred to other aggressive maneuvers without additional learning attempts.

It is perceivable that an alternative or complementary approach to iterative learning is a comprehensive identification of the system. Better understanding of in-flight dynamics could lead to an accurate simulation environment to assist with the learning process. Thus far, our attempts to better understand the robot's dynamics have been challenged by the lack of force/torque sensors with suitable range and resolution. In-flight identification is complicated by the inherent nonlinear dynamics and instability of the robot—the stabilizing feedback may interfere with the identification process (Forssell and Ljung, 1999). Additionally, the motion required for identification often destabilizes the vehicle. To characterize the damping torque about the pitch or roll axis of the robot, for example, requires the robot to rotate for a prolonged period and distance which inevitably results in crashes. The lack of realistic a dynamic model renders flight simulation inaccurate, and therefore, not suitable for use in the learning process. It is possible that a comprehensive computational fluid dynamic simulations could alleviate the problem, but the approach is prohibitively expensive computationally and does not offer insights into the interaction between actuator dynamics and aerodynamics. Taking all these points into consideration, we believe that the proposed experimental approach is appropriate for realizing an aggressive trajectory by a small flapping-wing robot, despite some limitations mentioned.

References

Achtelik MW, Lynen S, Chli M and Siegwart R (2013) Inversion based direct position control and trajectory following for micro aerial vehicles. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 2933–2939.

Bouabdallah S and Siegwart R (2005) Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, pp. 2247–2252.

Brian DA and Moore JB (2005) Optimal filtering. *Dower, Jan* .

Bristow DA, Tharayil M and Alleyne AG (2006) A survey of iterative learning control. *Control Systems, IEEE* 26(3): 96–114.

Chirarattananon P, Ma KY and Wood RJ (2014a) Adaptive control of a millimeter-scale flapping-wing robot". *Bioinspiration & biomimetics* To appear.

Chirarattananon P, Ma KY and Wood RJ (2014b) Fly on the wall. In: *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*. IEEE, pp. 1001–1008.

Chirarattananon P, Ma KY and Wood RJ (2014c) Single-loop control and trajectory following of a flapping-wing microrobot. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 37–44.

Chirarattananon P and Wood RJ (2013) Identification of flight aerodynamics for flapping-wing microrobots. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, pp. 1389–1396.

Cory R and Tedrake R (2008) Experiments in fixed-wing uav perching. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. AIAA Reston, VA, pp. 1–12.

Coza C, Nicol C, Macnab C and Ramirez-Serrano A (2011) Adaptive fuzzy control for a quadrotor helicopter robust to wind buffeting. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology* 22(5, 6): 267–283.

De Croon G, Groen M, De Wagter C, Remes B, Ruijsink R and Van Oudheusden B (2012) Design, aerodynamics and autonomy of the delfly. *Bioinspiration & biomimetics* 7(2): 025003.

Deng X, Schenato L and Sastry SS (2006) Flapping flight for biomimetic robotic insects: Part ii-flight control design. *Robotics, IEEE Transactions on* 22(4): 789–803.

Desbiens AL, Asbeck AT and Cutkosky MR (2011) Landing, perching and taking off from vertical surfaces. *The International Journal of Robotics Research* : 0278364910393286.

Duhamel PE, Perez-Arancibia NO, Barrows GL and Wood RJ (2013) Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots. *Mechatronics, IEEE/ASME Transactions on* 18(2): 556–568.

Dydek ZT, Annaswamy AM and Lavretsky E (2013) Adaptive control of quadrotor uavs: A design trade study with flight evaluations. *Control Systems Technology, IEEE Transactions on* 21(4): 1400–1406.

Faruque I and Sean Humbert J (2010) Dipteran insect flight dynamics. part 1 longitudinal motion about hover. *Journal of Theoretical Biology* 264(2): 538–552.

Forssell U and Ljung L (1999) Closed-loop identification revisited. *Automatica* 35(7): 1215–1241.

Fuller SB, Karpelson M, Censi A, Ma KY and Wood RJ (2014) Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli. *Journal of The Royal Society Interface* 11(97): 20140281.

Gerdes J, Holness A, Perez-Rosado A, Roberts L, Greisinger A, Barnett E, Kempny J, Lingam D, Yeh CH, Bruck HA et al. (2014) Robo raven: A flapping-wing air vehicle with highly compliant and independently controlled wings. *Soft Robotics* 1(4): 275–288.

Guerreiro BJ, Silvestre C, Cunha R, Cao C and Hovakimyan N (2009) L 1 adaptive control for autonomous rotorcraft. In: *American Control Conference, 2009. ACC'09.* IEEE, pp. 3250–3255.

Hawkes EW, Christensen DL, Eason EV, Estrada MA, Heverly M, Hilgemann E, Jiang H, Pope MT, Parness A and Cutkosky MR (2013) Dynamic surface grasping with directional adhesion. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 5487–5493.

Hehn M and D'Andrea R (2014) A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers. *Mechatronics* 24(8): 954–965.

Hoburg W and Tedrake R (2009) System identification of post stall aerodynamics for UAV perching. In: *Proceedings of the AIAA Infotech@ Aerospace Conference*. pp. 1–9.

Huang H, Hoffmann GM, Waslander SL and Tomlin CJ (2009) Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 3277–3282.

Huang M, Xian B, Diao C, Yang K and Feng Y (2010) Adaptive tracking control of underactuated quadrotor unmanned aerial vehicles via backstepping. In: *American Control Conference (ACC), 2010*. IEEE, pp. 2076–2081.

Hunt G, Mitzalis F, Alhinai T, Hooper PA and Kovac M (2014) 3d printing with flying robots. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 4493–4499.

Ioannou P, Annaswamy AM, Narendra K, Jafari S, Rudd L, Ortega R, Boskovic J et al. (2014) - adaptive control: Stability, robustness, and interpretations. *Automatic Control, IEEE Transactions on* 59(11): 3075–3080.

Jackson JD and Jackson JD (1962) *Classical electrodynamics*, volume 3. Wiley New York etc.

Karpelson M, Wei GY and Wood RJ (2009) Milligram-scale high-voltage power electronics for piezoelectric microrobots. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, pp. 2217–2224.

Kovač M, Germann J, Hürzeler C, Siegwart RY and Floreano D (2009) A perching mechanism for micro aerial vehicles. *Journal of Micro-Nano Mechatronics* 5(3-4): 77–91.

Krishnan A and Sane SP (2014) Visual feedback influences antennal positioning in flying hawk moths. *The Journal of experimental biology* 217(6): 908–917.

Lee D, Kim HJ and Sastry S (2009) Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of control, Automation and systems* 7(3): 419–428.

Lee T, Leoky M and McClamroch NH (2010) Geometric tracking control of a quadrotor UAV on SE(3). In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, pp. 5420–5425.

Lentink D, Jongerius SR and Bradshaw NL (2010) The scalable design of flapping micro-air vehicles inspired by insect flight. In: *Flying insects and robots*. Springer, pp. 185–205.

Lupashin S, Schöllig A, Sherback M and D'Andrea R (2010) A simple learning strategy for high-speed quadrocopter multi-flips. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, pp. 1642–1648.

Ma KY, Chirarattananon P, Fuller SB and Wood RJ (2013) Controlled flight of a biologically inspired, insect-scale robot. *Science* 340(6132): 603–607.

Ma KY, Chirarattananon P and Wood RJ (2015) Design and fabrication of an insect-scale flying robot for control autonomy. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE.

Ma KY, Felton SM and Wood RJ (2012) Design, fabrication, and modeling of the split actuator microrobotic bee. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 1133–1140.

Madani T and Benallegue A (2008) Adaptive control via backstepping technique and neural networks of a quadrotor helicopter. In: *Proceedings of the 17th World Congress of The Internaional Federation of Automatic Control*.

Malka R, Desbiens AL, Chen Y and Wood RJ (2014) Principles of microscale flexure hinge design for enhanced endurance. In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, pp. 2879–2885.

Mellinger D and Kumar V (2011) Minimum snap trajectory generation and control for quadrotors. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 2520–2525.

Mellinger D, Michael N and Kumar V (2012) Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research* 31(5): 664–674.

Mengüç Y, Röhrig M, Abusomwan U, Hölscher H and Sitti M (2014) Staying sticky: contact self-cleaning of gecko-inspired adhesives. *Journal of The Royal Society Interface* 11(94): 20131205.

Mueller FL, Schoellig AP and D'Andrea R (2012) Iterative learning of feed-forward corrections for high-performance tracking. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 3276–3281.

Nicol C, Macnab C and Ramirez-Serrano A (2011) Robust adaptive control of a quadrotor helicopter. *Mechatronics* 21(6): 927–938.

Oppenheimer MW, Doman DB and Sigthorsson D (2009) Dynamics and control of a minimally actuated biomimetic vehicle: Part ii-control. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference*.

Orlowski CT and Girard AR (2012) Dynamics, stability, and control analyses of flapping wing micro-air vehicles. *Progress in Aerospace Sciences* 51: 18–30.

Orsag M, Korpela C and Oh P (2013) Modeling and control of mm-uav: Mobile manipulating unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems* 69(1-4): 227–240.

Richter C and Lipson H (2011) Untethered hovering flapping flight of a 3d-printed mechanical insect. *Artificial life* 17(2): 73–86.

Ristroph L, Bergou AJ, Berman GJ, Guckenheimer J, Wang ZJ and Cohen I (2012) Dynamics, control, and stabilization of turning flight in fruit flies. In: *Natural locomotion in fluids and on surfaces*. Springer, pp. 83–99.

Ristroph L, Ristroph G, Morozova S, Bergou AJ, Chang S, Guckenheimer J, Wang ZJ and Cohen I (2013) Active and passive stabilization of body pitch in insect flight. *Journal of The Royal Society Interface* 10(85): 20130237.

Schoellig AP, Mueller FL and D'Andrea R (2012) Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots* 33(1-2): 103–127.

Schöllig A and D'Andrea R (2009) Optimization-based iterative learning control for trajectory tracking. In: *Proceedings of the European control conference (ECC)*. pp. 1505–1510.

Slotine JJE, Li W et al. (1991) *Applied nonlinear control*, volume 199. Prentice-Hall Englewood Cliffs, NJ.

Sreetharan PS, Whitney JP, Strauss MD and Wood RJ (2012) Monolithic fabrication of millimeter-scale machines. *Journal of Micromechanics and Microengineering* 22(5): 055027.

Taha HE, Hajj MR and Nayfeh AH (2012) Flight dynamics and control of flapping-wing MAVs: a review. *Nonlinear Dynamics* 70(2): 907–939.

Teoh ZE, Fuller SB, Chirarattananon P, Prez-Arancibia N, Greenberg JD and Wood RJ (2012) A hovering flapping-wing microrobot with altitude control and passive upright stability. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 3209–3216.

Van Nieuwstadt MJ and Murray RM (1997) Real time trajectory generation for differentially flat systems .

Williams RJ and Zipser D (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2): 270–280.

Wood R, Steltz E and Fearing R (2005) Optimal energy density piezoelectric bending actuators. *Sensors and Actuators A: Physical* 119(2): 476–488.

Wood RJ, Finio B, Karpelson M, Ma K, Pérez-Arancibia NO, Sreetharan PS, Tanaka H and Whitney JP (2012) Progress on 'pico' air vehicles. *The International Journal of Robotics Research* 31(11): 1292–1302.

Zhang X, Tong T, Brooks D and Wei GY (2014) Evaluating adaptive clocking for supply-noise resilience in battery-powered aerial microrobotic system-on-chip. *Circuits and Systems I: Regular Papers, IEEE Transactions on* 61(8): 2309–2317.

Appendix A: Index to Multimedia Extensions

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org.
After 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table 1: Table of Multimedia Extensions

Extension	Media	Description
1	Video	Open-loop, hovering, trajectory following, and perching flights