2014 5th IEEE RAS & EMBS International Conference on
Biomedical Robotics and Biomechatronics (BioRob)
August 12-15, 2014. São Paulo, Brazil

Fly on the Wall

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. Magnetic force was chosen to enable attachment to the vertical surface due to its simplicity. We show that by learning from previous failed attempts, the robotic fly could successfully perch on a magnetic wall after eight iterations.**

I. Introduction

Over the last few decades, insect flight has inspired scientists and engineers to understand and translate this ubiquitous form of locomotion into man-made machines. Flies are a convenient model organism for studying insect flight—evolving sophisticated flight mechanics and exhibiting exceptional agility and maneuverability. Researchers have developed a number of biologically-inspired, flapping-wing flying vehicles [1], [2], including an insect-scale robotic fly that successfully demonstrated unconstrained but tethered flight [3].

To date, artificial flapping wing flight at low Reynolds numbers usually relies on passive stability to achieve hover [1], [2]. Unlike [1], [2], flying insects and the robotic fly in [3] are inherently unstable. This instability necessitates active control, but also leads to increased maneuverability. Thus far, the robotic fly has only performed basic flight maneuvers and stable hovering [3], [4], [5], and has yet to demonstrate any aggressive or acrobatic maneuvers, encountering issues in control, fast dynamics, and lack of understanding in the small-scale unsteady aerodynamics.

Other classes of Micro Aerial Vehicles (MAVs) such as quadrotors and helicopters have demonstrated highly aggressive maneuvers such as flying through narrow vertical gaps [6], performing multiple flips [7], and inverted flight [8]. In these examples, a common theme in control methods is "learning". In [8], reinforcement learning was used with information from a human pilot's commands to design a controller for inverted helicopter flight. In [6], [7], iterative learning approaches were taken. It is anticipated that similar strategies could be employed for enabling aggressive aerial maneuvers with a flapping-wing flying robot.

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.

The authors are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA, and the Wyss institute for Biologically Inspired Engineering, Harvard University, Boston, MA, 02115, USA (email: chirarat@fas.harvard.edu; kevinma@seas.harvard.edu; rjwood@eecs.harvard.edu).

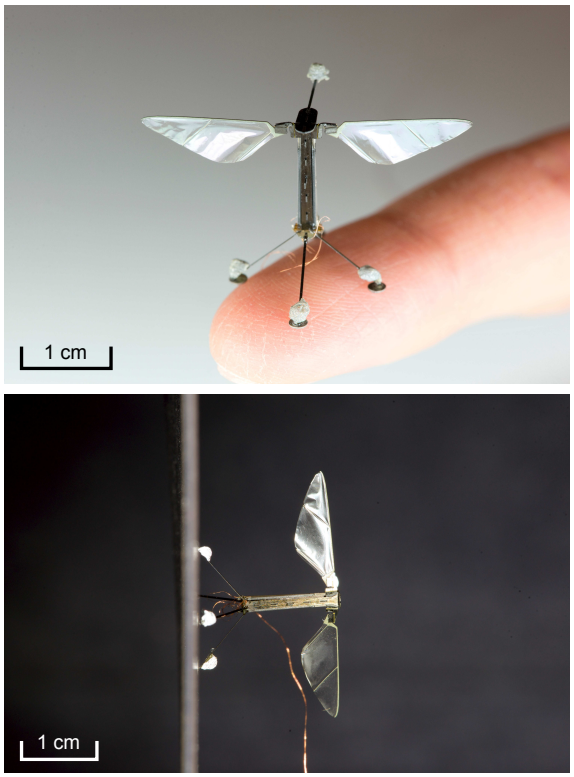

Fig. 1. (top) Photograph of a biologically-inspired robotic fly on a finger with four retro-reflective markers for tracking purposes.. (bottom) A robot attached to a magnetic wall via the aid of 6mil steel shims attached to the base of the landing gear.

In this paper, we address the challenge of performing an aggressive flight maneuver with the robotic fly shown in figure 1 (top). Specifically, our objective is to design a flight controller and a simple attachment mechanism to allow the robot to land, or perch, on a vertical surface as illustrated in figure 1 (bottom)

The topic of perching an MAV has been addressed previously at larger scales [9], [10], [11]. In [9], 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 [10], [11], 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 payload capacity of the robotic fly in this study renders elaborate perching mechanisms as infeasible options. Fortunately, the use of magnetic force

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

The *Iterative Learning Control* (ILC) [12] technique was used in addition to the existing feedback control loop. The ILC algorithm allows the robot to learn from its previous flights and improves its flight performance through repetition of the same trajectory.

In section II, we briefly discuss the properties of the robot, the control strategies, and the dynamic model of the perching flight. We then present a method of generating a perching trajectory followed by the formulation on the proposed ILC method in section III and IV. Experimental results are shown in section V followed by discussion and conclusion.

II. Robot Description and Control Strategies

A. Robot Description

The insect-scale flapping-wing robot shown in figure 1 was first presented in [13]. The robot has a wingspan of 3cm and weighs under 100mg. It is fabricated in-house using the *Smart Composite Microstructures* (SCM) process. Piezoelectric bimorph actuators are chosen over electromagnetic motors for flight muscle due to their favorable scalability [14]. In the current prototype, the robot is equipped with two piezoelectric bending actuators, each capable of independently driving a single wing. When a voltage is applied to actuator, it induces motion at the tip of the actuator. This approximately linear displacement is transformed into an angular wing motion by a spherical four-bar transmission. A passive 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 [3]. As a result, we nominally operate the system with sinusoidal signals near the system's resonant frequency of 120Hz to maximize the flapping stroke amplitude and minimize reactive power expended by 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. This leaves researchers the task of designing a controller that determines the required lift and torque to stabilize the robot's flight. More details on the robot design and torque generation schemes can be found in [3], [13].

In this work, we define pitch, roll, and yaw rotations along the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes of the body-fitted right-hand coordinates as illustrated in figure 2. The inertial coordinate frame is labeled by $\hat{X}$, $\hat{Y}$, and $\hat{Z}$. A rotation matrix R relates the orientation between the body-fixed frame and the inertial frame.

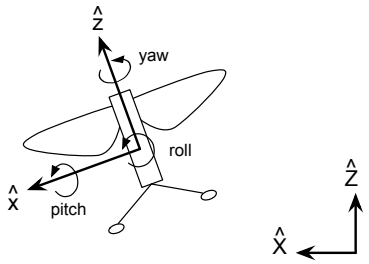

Fig. 2. Definitions of the inertial frame, the body-attached frame, and roll, pitch, and yaw axes.

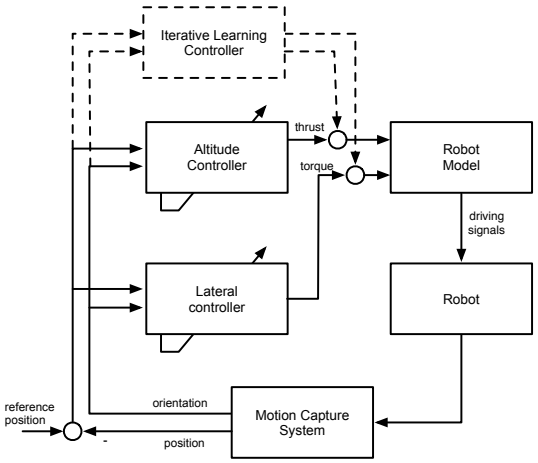

Fig. 3. A block diagram illustrating the feedback control architecture of the robot. The ILC control block (in dashed lines) is implemented in additional to the existing closed-loop system.

B. Control Strategies

Similar to its insect counterparts, the flapping-wing robot in figure 1 is inherently unstable and requires active feedback control [15]. We have demonstrated that, even with advances in the manufacturing process, there are still uncertainties in the system and noticeable variations from robot to robot. In [4], it is shown that a controller able to identify and correct for some unknown parameters can improve the flight performance by reducing the position errors in hovering flights by approximately 50% (to less than 1cm).

Achieving a stable hovering flight at this scale is challenging for a number of reasons: inherent vehicle instability, extremely fast dynamics, and high susceptibility to disturbances. From a controls perspective, a hovering flight is a simplified case of more general maneuvers with no feedforward components. In order to realize a series of rapid maneuvers, another controller was designed in [5]. This controller eliminates the assumption that the attitude dynamics is generally much faster than the lateral dynamics as typically assumed in MAV literature [1], [6], while retaining an adaptive ability that was found to be crucial for flight at

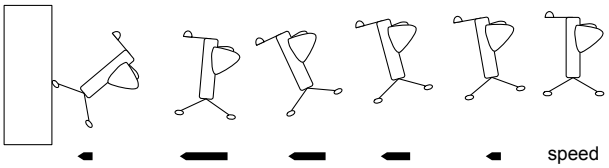

Fig. 4. A schematic diagram demonstrating how the robot can perch on a vertical wall. Initially it needs to build up sufficient forward momentum to retain it when the robot rotates to the opposite direction at the moment of landing on the wall.

this scale.

In this paper, we employ this controller for preliminary trajectory following. The simplified block diagram is shown in figure 3. Since the dynamic model of the robot assumed by the controller is not sufficiently accurate to capture high-frequency dynamics, additional steps are required to realize more aggressive maneuvers such as perching on a vertical wall. The approach we take here is iterative learning control, which can be implemented in additional to the current closed-loop system as highlighted in figure 3. Feedback is executed in real time, whereas the iterative control part is updated offline between flights.

C. 2D Model of Perching Flight

To simplify the problem of landing on a vertical wall, we restrict our analysis to the two dimensional plane defined as $\hat{X} - \hat{Z}$ in the inertial coordinate frame. The state variables of interest consists of the position and velocity of the robot along the $\hat{X}$ and $\hat{Z}$ directions, the *tilt* angle (θ) of the robot defined as the angle between the $\hat{z}$ axis and the $\hat{Z}$ axis as projected onto the $\hat{X} - \hat{Z}$ plane (with $\hat{z}$ tilted in the $-\hat{X}$ 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 (1).

$$\mathbf{X} = \begin{bmatrix} X & \dot{X} & Z & \dot{Z} & \theta & \dot{\theta} & \Gamma \end{bmatrix}^T. \tag{1}$$

The dynamics of $\dot{\theta}$ are assumed to depend on the normalized projected torque $\bar{\tau}$ as $\ddot{\theta} = \bar{\tau}$ and the thrust is related to the thrust input T by the first order dynamics as $\dot{\Gamma} = -\gamma(\Gamma - T)$ for a positive constant γ. Therefore, T and $\bar{\tau}$ are regarded as two inputs to the system:

$$\mathbf{U} = \begin{bmatrix} T & \bar{\tau} \end{bmatrix}^T. \tag{2}$$

The robotic fly is an under-actuated system, similar in some regards to quadrotors [6], [7]. 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. As a consequence, the time derivative of the state vector $\mathbf{X}$ can be found from the following expression:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \dot{X} \\ \dot{Z} \\ \dot{\theta} \\ \Gamma \end{bmatrix} = \begin{bmatrix} -\Gamma\sin\theta - \xi\dot{X} \\ \Gamma\cos\theta - g \\ \bar{\tau} \\ -\gamma(\Gamma - T) \end{bmatrix}, \tag{3}$$

where ξ is a damping coefficient and g is the gravitational constant. This equation formulates a framework for the analysis of trajectory generation and ILC in sections (III) and (IV).

III. 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 (3). 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 to contact with the wall at steep tilt angle (preferably larger than $45°$). Hence, it requires to generate significant amount of torque at the very end of the trajectory. Since the torque generated is coupled with the thrust, 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 assure that the robot does not move backwards. A schematic diagram illustrating a perching trajectory is shown in figure 4.

Mathematically, the 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 (4)

$$\begin{aligned} J &= \int g(\mathbf{X}, \mathbf{U})\,\mathrm{d}t + h(\mathbf{X}_T) \\ &= \int \begin{bmatrix} \mathbf{X} & \mathbf{U} \end{bmatrix} \Lambda_g \begin{bmatrix} \mathbf{X} & \mathbf{U} \end{bmatrix}^T - \lambda_g \dot{X}\,\mathrm{d}t \\ &\quad + (\mathbf{X}_T - \mathbf{X}_{T,ref})^T \Lambda_{gT} (\mathbf{X}_T - \mathbf{X}_{T,ref}) - \lambda_{gT}\dot{X}_T, \end{aligned} \tag{4}$$

where $\mathbf{X}_T$ and $\mathbf{X}_{T,ref}$ denote the terminal state vector and the desired terminal state vector, Λ_g, Λ_{gT}, λ_g, and λ_{gT} are diagonal matrices and scalar constants. 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} and λ_{gT} 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:

$$\Gamma = \left(\sum_{i=0}^{i=N_\Gamma} a_i t^i\right)^2 \qquad \tau = \Gamma \sum_{i=0}^{i=N_\tau} b_i t^i, \tag{5}$$

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 to force the thrust to always be 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.

To find a locally optimal solution of equations (4) and (5) using gradient methods, the difficulty lies on a procedure for calculating a Jacobian. Here we compute the gradient with the *Real-Time Recurrent Learning* (RTRL) method [16], [17].

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 could be a_i or b_i), we have

$$\begin{aligned} \frac{\partial}{\partial \alpha}\left(\dot{\mathbf{X}}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathbf{X}}{\partial \alpha}\right) = \frac{\partial f}{\partial \mathbf{X}}\frac{\partial \mathbf{X}}{\partial \alpha} + \frac{\partial f}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \alpha} \\ &= \frac{\mathrm{d}}{\mathrm{d}t}P = \frac{\partial f}{\partial \mathbf{X}}P + \frac{\partial f}{\partial \mathbf{U}}Q, \end{aligned} \tag{6}$$

where P and Q have been defined as $\partial \mathbf{X}/\partial \alpha$ and $\partial \mathbf{U}/\partial \alpha$ respectively. According to equation (4), the Jacobian $\partial J/\partial \alpha$ is then simply given as

$$\frac{\partial J}{\partial \alpha} = \int \left(\frac{\partial g}{\partial \mathbf{X}}P + \frac{\partial g}{\partial \mathbf{U}}Q\right) dt + \frac{\partial h}{\partial \mathbf{X}}P + \frac{\partial h}{\partial \mathbf{U}}Q. \tag{7}$$

It is straightforward to obtain $\partial f/\partial \mathbf{X}$ and $\partial f/\partial \mathbf{U}$ from equation (3). Thus, by integrating forward equation (6) to find P, the gradient $\partial J/\partial \alpha$ can be evaluated from equation (7). To perform a gradient descent, the cost function is expanded to its second order approximation as

$$J(\alpha + \delta\alpha) \approx J(\alpha) + \left(\frac{\partial J}{\partial \alpha}\right)^T \delta\alpha + \frac{1}{2}\delta\alpha^T \left(\frac{\partial^2 J}{\partial \alpha^2}\right) \delta\alpha. \tag{8}$$

Inspired by the Gauss-Newton algorithm, here we opt to approximate the Hessian by taking a derivative of equation (7) 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 (8) 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, which improves the stability. Performance could also be tweaked by altering the cost and the reference state.

IV. Iterative Learning Control for Perching on a Vertical Surface

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 learning controller, on the other hand, relies solely on repetition and repetitive disturbances or systematic errors in the modeling.

To consider a whole trajectory, we consider the dynamics using a lifted representation, similar to the approach taken in [18]. That is, we discretize and consolidate the state vectors and the inputs into a long vector given by the following

$$\begin{aligned} \mathbf{X}^* &= \begin{bmatrix} \mathbf{X}(t_1) & \mathbf{X}(t_2) & \dots & \mathbf{X}(t_N) \end{bmatrix}^T \\ \mathbf{U}^* &= \begin{bmatrix} \mathbf{U}(t_1) & \mathbf{U}(t_2) & \dots & \mathbf{U}(t_N) \end{bmatrix}^T. \end{aligned} \tag{9}$$

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

$$\dot{\mathbf{X}}^*_{ref} = f^*\left(\mathbf{U}^*_{ff}\right). \tag{10}$$

In reality, it is not anticipated that the model will be perfect. Instead of attempting to find a better model, we assume that the input into the system can be regarded as a combination of the command input and the unknown disturbance input $\mathbf{U}^* = \mathbf{U}^*_c - \mathbf{U}^*_d$, 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:

$$\begin{aligned} \dot{\mathbf{X}}^* &= f^*(\mathbf{U}^*) \\ &= f^*(\mathbf{U}^*_c - \mathbf{U}^*_d) \\ &= f^*\left(\mathbf{U}^*_{ff} + \hat{\mathbf{U}}^*_d - \mathbf{U}^*_d\right). \end{aligned} \tag{11}$$

In order to calculate the unknown disturbance input, we first define the estimation error at iteration j as

$$\tilde{\mathbf{U}}^*_{d,j} = \hat{\mathbf{U}}^*_{d,j} - \mathbf{U}^*_d. \tag{12}$$

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

$$\dot{\mathbf{X}}^*_j \approx f^*\left(\mathbf{U}^*_{ff}\right) + \left(\frac{\mathrm{d}}{\mathrm{d}\mathbf{U}^*} f^*\Big|_{\mathbf{U}^*_{ff}}\right) \cdot \tilde{\mathbf{U}}^*_{d,j}. \tag{13}$$

The quantity on the left hand side of equation (13) could be obtained by post-processing the recorded trajectory. The difference of the measured $\dot{\mathbf{X}}^*_j$ and $\dot{\mathbf{X}}^*_{ref}$ forms an error vector $\mathbf{e}_j$ that is a function of $\tilde{\mathbf{U}}^*_{d,j}$

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

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:

$$\hat{\mathbf{U}}^*_{d,j+1} = \hat{\mathbf{U}}^*_{d,j} - F^T \Delta \mathbf{e}_j,$$

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

$$\mathbf{e}_{j+1}^T \mathbf{e}_{j+1} = \mathbf{e}_j^T \left(I - FF^T\Delta\right)^T \left(I - FF^T\Delta\right) \mathbf{e}_j.$$

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

$$\mathbf{e}_{j+1}^T \mathbf{e}_{j+1} \leq \sigma \mathbf{e}_j^T \mathbf{e}_j \qquad \text{for} \quad \exists\, 0 \leq \sigma < 1.$$

In this implementation, F can be computed by representing the robot's dynamics given by equation (3) along the reference trajectory using a *Linear Time Varying* (LTV) configuration.

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

and the matrix F is simply given by

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

where the elements can be shown to be

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

A. Consideration of Initial Conditions

In the previous section, it is shown that the norm of error vector should gradually decrease after each iteration. However, the presented analysis assumed that each trajectory always starts with the same initial conditions, in a way that minimizing the error in $\dot{\mathbf{X}}^*$ is sufficient to bring the actual trajectory close to the reference trajectory. In our case, where the initial condition involves a robot hovering in place, that condition is approximately satisfied for the tilt angle and the initial velocity, but not for the position. In our prior work, it was shown that the RMS of the position error during hover was just below 1cm [4]. This means that even if the error vector 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 is stable as measured by a metric given by the controller in [5]. Furthermore, when the starting position is not zero, trajectory tracking will not start from the beginning of the pre-planned trajectory. To demonstrate, 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 $0 \leq t' \leq T_f$, we seek to find t_0' that satisfies the equation

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

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.

B. Implementation in Three Dimensions

In practice, the controller only takes into consideration the direction of the $\hat{z}$ 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 $\hat{X}$ direction, the reference torque input $\bar{\tau}$ points along a negative $\hat{Y}$ 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, it is possible to find a combination of pitch and roll torques in the body frame that point in the $-\hat{Y}$ direction with the specified magnitude.

V. Experiments

A. Experimental setup

The current prototype of the robotic fly 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 × 0.3 × 0.3m—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. Direct measurements of the robot's velocity and rotation rates are not available, so they were substituted by their filtered derivative representations.

B. Trajectory Optimization

Initially, the perching trajectory is crudely hand-designed with a target distance near 12cm from the starting position and the trajectory duration of 0.65s. This was then

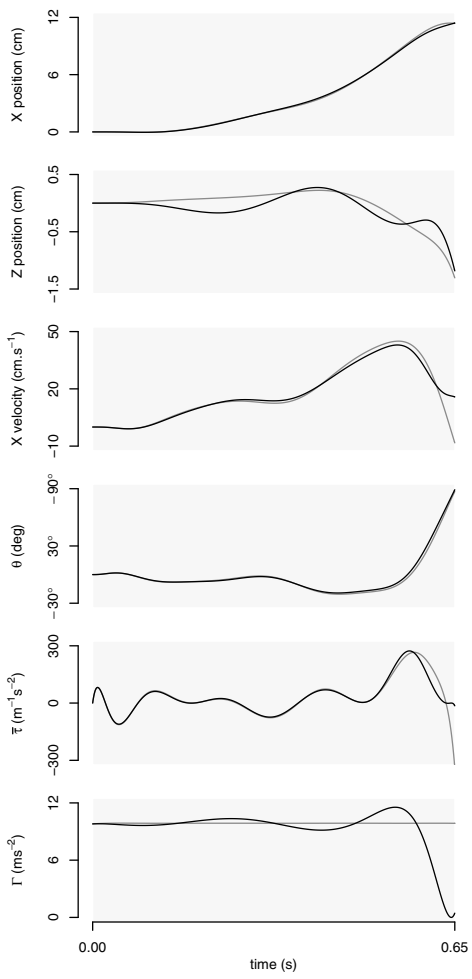

Fig. 5. A candidate trajectory for perching. The grey lines show a parametrized version of a hand-designed trajectory. Black lines represent the final trajectory after the optimization.

parametrized by polynomial functions as given by equation (5) with $N_\Gamma = 8$ and $N_\tau = 12$. The parametrized trajectory is illustrated with grey lines in figure 5. Next, the trajectory is optimized using the strategy proposed in section III and the results are shown as black lines in figure 5. It can be seen that the most significant difference is in the terminal velocity which is, in fact, negative before the optimization. The optimized thrust is also small at the end. Understandably this is in order to reduce the amount of deceleration and preserve the forward momentum. The resultant trajectory is 11.4cm long with the projected terminal tilt angle close to $90°$.

C. Landing Mechanism

The magnetic wall 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 $1100\text{kg}\cdot\text{m}^{-2}$. On the robot, four discs of 6mil (0.15mm) steel shims, each with a diameter of 2mm, were attached to the landing gear. This brought up the total weight of the robot from 80mg to 100mg.

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 ≈ 60mg 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} [19]. As a result, the contribution of the magnetic force should be negligible when the robot is not in contact with the magnetic wall.

D. Experimental Results

Prior to perching experiments, the robot has 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 needs to be trimmed for flight. The trimming process starts with short, unstable open-loop flights that each lasts less than 0.4s to determine a set of driving signals that minimize the residual torque exhibited by the robot, due to unavoidable mechanical asymmetries. That is followed by a closed-loop trimming process, in which the adaptive part of the controller corrects for torque offsets further until the robot can hover with position errors on the order of 1cm or less.

We must ensure the robot will start its trajectory on or somewhere in front of the prescribed perching trajectory's starting point. Because of this 1.0cm uncertainty in the starting position of the robot, we actually define the start of the trajectory to be at -1.0cm from the hovering setpoint of the robot at 0cm. The target vertical wall is placed at 10.4cm from the hovering setpoint. The controller allowed the perching attempt to begin only when the robot is less than one centimeter away from the setpoint ($-1.0\text{cm} \leq X \leq 1.0\text{cm}$). Thus, given the 11.4cm prescribed trajectory, the robot will start with an initial condition lying in front of the trajectory's starting point.

The first perching flight (iteration 0) was executed with no correction from the ILC algorithm. The plots of this iteration's trajectory are illustrated in figure 6(a). The robot only reaches a distance of 6.3cm and a tilt angle of $41°$ before falling out of the air.

In the iteration 1, after implementing the first estimate of $\mathbf{U}_d^*$, the robot could get closer to the wall, i.e., it achieved the distance of 8.5cm before losing all the forward momentum, at which point the robot had a tilt angle of $78°$. The corresponding trajectory is shown in figure 6(b).

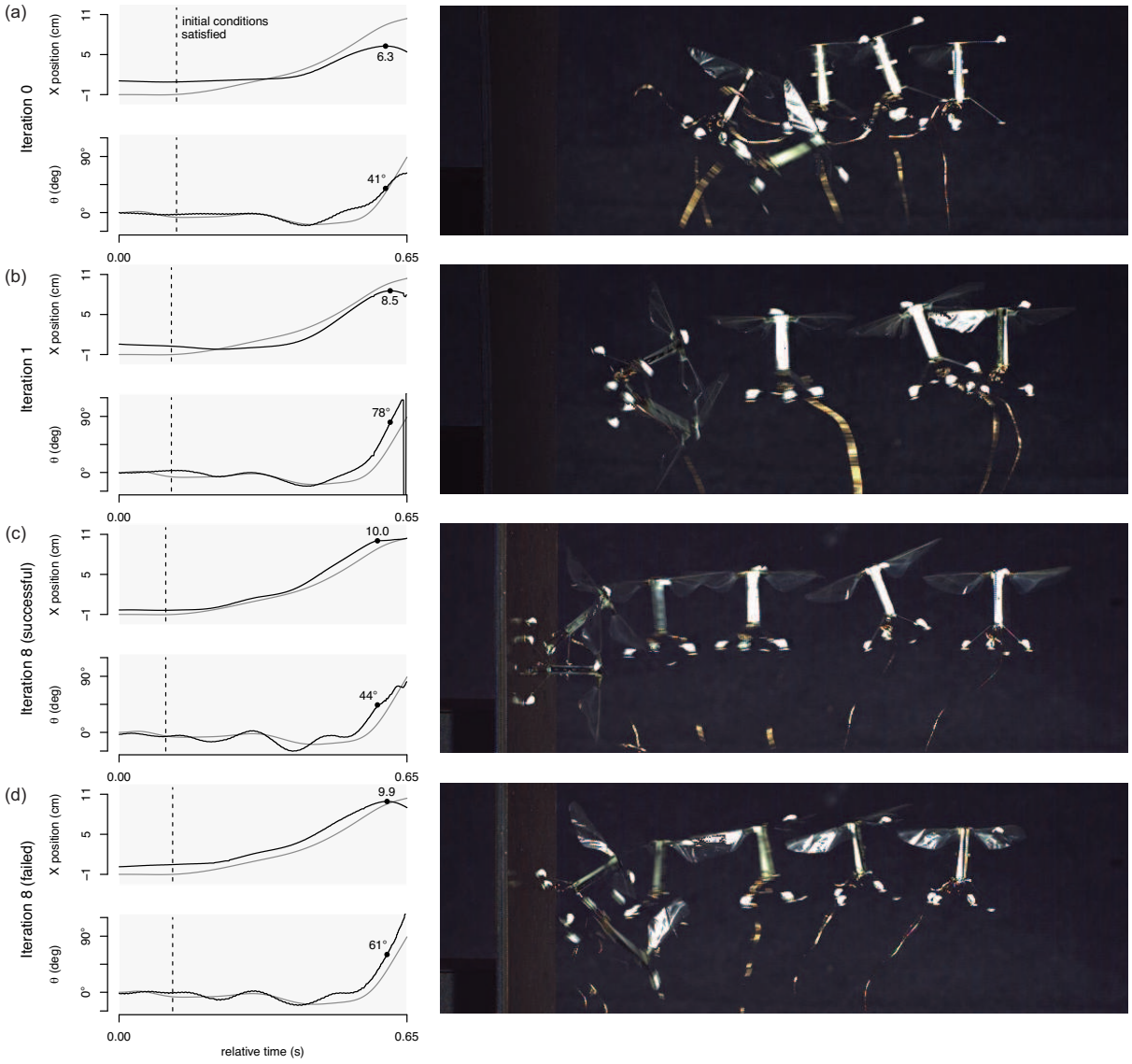

Fig. 6. Recorded trajectories accompanied by composite images constructed from the videos. References are shown in grey lines and the actual trajectories are in black lines. (a) The trajectory obtained without any command from the ILC algorithm. (b) The trajectory generated after one learning iteration. (c) A successful trajectory obtained after 8 iterations of learning. (c) A failed perching attempt obtained from the same command as in (c).

Due to the nature of the experiment and the delicate character of the robot, it is impossible to ensure that physical properties of the robot remain unaltered over the course of several flights. Sources of 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 wire fatigue. After subsequent repairs to the robot, 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 6 iterations just as the robot was nearing a successful perching trajectory. 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. A few subsequent flights using the same command resulted in a mix of successful and failed wall perchings. One example of a successful attempt is demonstrated in figure 6(c). In this attempt, the robot first contacted the wall when the tilt angle was around $45°$. A failed perching is also presented in figure 6(d). 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.

VI. Discussion and Conclusion

In this work, we have shown that a centimeter-scale, flapping-wing flying robot is capable of performing an aggressive aerial maneuver—perching on a vertical wall. From a controls perspective, such a maneuver differs considerably from hover or slow maneuvers that have been demonstrated before. To land on a vertical surface, first we constructed a nominal perching trajectory via an optimization method that assumes a simplified dynamics model in two dimensions. For control, we opted to implement an iterative learning control algorithm in addition to the existing 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 8 iterations, the proposed control strategies enabled the robot to successfully land on a vertical surface as desired.

Without the learning algorithm used in this paper, the existing controller is unable to command the robot in following the prescribed trajectory. Understandably, like most controllers, the dynamic model assumed by the previous controller only accurately captures slow system dynamics, lacking 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 [4].

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 needed. It is nontrivial to construct a detachable attachment mechanism similar to those seen in [20], [21], 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.

References

[1] G. de Croon, M. Groen, C. De Wagter, B. Remes, R. Ruijsink, and B. van Oudheusden, "Design, aerodynamics and autonomy of the delfly," *Bioinspiration & biomimetics*, vol. 7, no. 2, p. 025003, 2012.

[2] C. Richter and H. Lipson, "Untethered hovering flapping flight of a 3d-printed mechanical insect," *Artificial life*, vol. 17, no. 2, pp. 73–86, 2011.

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

[4] P. Chirarattananon, K. Y. Ma, and R. J. Wood, "Adaptive control for takeoff, hovering, and landing of a robotic fly," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3808–3815.

[5] ——, "Single-loop control and trajectory following of a flapping-wing microrobot," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, to appear.

[6] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

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

[8] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*. Springer, 2006, pp. 363–372.

[9] R. Cory and R. Tedrake, "Experiments in fixed-wing uav perching," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.

[10] A. L. Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 355–370, 2011.

[11] M. Kovač, J. Germann, C. Hürzeler, R. Y. Siegwart, and D. Floreano, "A perching mechanism for micro aerial vehicles," *Journal of Micro-Nano Mechatronics*, vol. 5, no. 3-4, pp. 77–91, 2009.

[12] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *Control Systems, IEEE*, vol. 26, no. 3, pp. 96–114, 2006.

[13] K. Y. Ma, S. M. Felton, and R. J. Wood, "Design, fabrication, and modeling of the split actuator microrobotic bee," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1133–1140.

[14] R. J. Wood, B. Finio, M. Karpelson, K. Ma, N. O. Pérez-Arancibia, P. S. Sreetharan, H. Tanaka, and J. P. Whitney, "Progress on 'pico' air vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1292–1302, 2012.

[15] L. Ristroph, G. Ristroph, S. Morozova, A. J. Bergou, S. Chang, J. Guckenheimer, Z. J. Wang, and I. Cohen, "Active and passive stabilization of body pitch in insect flight," *Journal of The Royal Society Interface*, vol. 10, no. 85, p. 20130237, 2013.

[16] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[17] W. Hoburg and R. Tedrake, "System identification of post stall aerodynamics for uav perching," in *Proceedings of the AIAA Infotech@ Aerospace Conference*, 2009.

[18] A. Schöllig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European control conference (ECC)*, 2009, pp. 1505–1510.

[19] J. D. Jackson, "Classical electrodynamics," *Classical Electrodynamics, 3rd Edition, by John David Jackson, pp. 832. ISBN 0-471-30932-X. Wiley-VCH, July 1998.*, vol. 1, 1998.

[20] E. W. Hawkes, D. L. Christensen, E. V. Eason, M. A. Estrada, M. Heverly, E. Hilgemann, H. Jiang, M. T. Pope, A. Parness, and M. R. Cutkosky, "Dynamic surface grasping with directional adhesion," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5487–5493.

[21] Y. Mengüç, M. Röhrig, U. Abusomwan, H. Hölscher, and M. Sitti, "Staying sticky: contact self-cleaning of gecko-inspired adhesives," *Journal of The Royal Society Interface*, vol. 11, no. 94, p. 20131205, 2014.