2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Robust, High-Rate Trajectory Tracking on Insect-Scale Soft-Actuated Aerial Robots with Deep-Learned Tube MPC

Andrea Tagliabue[1,*], Yi-Hsuan Hsiao[1,*], Urban Fasel[2], J. Nathan Kutz[3], Steven L. Brunton[3], YuFeng Chen[1], Jonathan P. How[1]

***Abstract*— Accurate and agile trajectory tracking in sub-gram Micro Aerial Vehicles (MAVs) is challenging, as the small scale of the robot induces large model uncertainties, demanding robust feedback controllers, while the fast dynamics and computational constraints prevent the deployment of computationally expensive strategies. In this work, we present an approach for agile and computationally efficient trajectory tracking on the MIT SoftFly [1], a sub-gram MAV (0.7 grams). Our strategy employs a cascaded control scheme, where an adaptive attitude controller is combined with a neural network (NN) policy trained to imitate a trajectory tracking robust tube model predictive controller (RTMPC). The NN policy is obtained using our recent work [2], which enables the policy to preserve the robustness of RTMPC, but at a fraction of its computational cost. We experimentally evaluate our approach, achieving position Root Mean Square Errors (RMSEs) lower than 1.8 cm even in the more challenging maneuvers, obtaining a 60% reduction in maximum position error compared to [3], and demonstrating robustness to large external disturbances.**

I. Introduction

Flying insects exhibit incredibly agile flight abilities, being capable of performing a flip in only 0.4 ms [4], flying under large wind disturbances [5]–[7], and withstanding collisions [8], [9]. Insect-scale flapping-wing MAVs [1], [10]–[12] have the potential to replicate these robustness and agile flight properties, extending their applications to tight and narrow spaces that become difficult for larger scale MAVs [13]–[17]. A key capability needed for the deployment of sub-gram MAVs in real-world missions is the ability to accurately track desired agile trajectories while being robust to real-world uncertainties, such as collisions and wind disturbances.

However, achieving robust, accurate, and agile trajectory tracking in sub-gram MAVs has significant challenges. First, their exceptionally fast dynamics [3] demand high-rate feedback control loops to ensure stability and rapid disturbance rejection, while the small payload limits onboard computation capabilities. Additionally, in order to maximize the lifespan of the robot components, control actions need to be planned in a way that is aware of actuation constraints. For instance, soft dielectric elastomer actuators (DEAs) suffer dielectric breakdown under a high electric field, posing a hard restraint on the maximum operating voltage [18]. Furthermore, the lifetime of passively-rotating wing hinges can be substantially extended under moderate control inputs. Lastly, manufacturing imperfection due to the small scale, and hard-to-model unsteady flapping-wing aerodynamics make it difficult to identify accurate models for simulation and control. Existing sub-gram MAVs have demonstrated promising agile flight capabilities [3], [19], [20], but none of the existing controllers *explicitly* account for environment and model uncertainties, and for actuation constraints and usage.

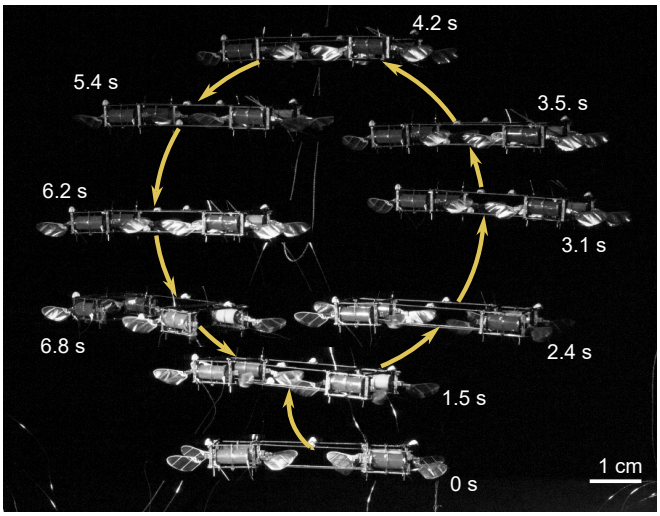

Fig. 1: Composite image showing a 7.5-second flight where the MIT SoftFly [1], a soft-actuated, insect-scale MAV, follows a vertical circle with 5 cm radius. The robot is controlled by a neural network (NN) policy, trained to reproduce the response of a robust model predictive controller. Thanks to its computational efficiency, the NN controls the robot at 2 kHz while running on a small offboard computer.

An agile trajectory tracking strategy that has found success on larger-scale MAVs (e.g., palm-sized quadrotors) consists of decoupling position and attitude control via a cascaded scheme, where a fast feedback loop (*inner*) controls the attitude of the MAV, while a potentially slower loop (*outer*) tracks the desired trajectory by generating commands for the attitude controller. An *outer loop* controller that enables agile, robust, actuation-aware trajectory tracking is model predictive control (MPC) [21]–[27]. This strategy generates actions by minimizing an objective function that explicitly trades tracking accuracy for actuation usage, taking into account the state and actuation constraints. This is achieved by solving a constrained optimization problem online, where a model of the robot is employed to plan along a predefined temporal horizon by taking into account the effects of future actions. Robust variants of MPC, such as robust tube model predictive controller (RTMPC) [23], [28], can additionally take into account uncertainties (disturbances, model errors) when generating their plans and control actions. This is done

*Equal contribution. Work funded in part by the AFOSR MURI FA9550-19-1-0386 and the National Science Foundation (FRR-2202477).

[1] Massachusetts Institute of Technology. {atagliab, yhhsiao, yufengc, jhow}@mit.edu
[2] Imperial College London. u.fasel@imperial.ac.uk
[3] University of Washington. {kutz, sbrunton}@uw.edu

by employing an auxiliary (ancillary) controller capable of maintaining the system within some distance ("cross-section" of a tube) from the nominal plan regardless of the realization of uncertainty. While MPC and RTMPC enable impressive performance on complex, agile robots, their computational cost limits the opportunities for onboard, high-rate deployment on computationally constrained platforms.

In this work, we present and experimentally demonstrate a computationally efficient method for accurate, robust, MPC-based trajectory tracking on sub-gram-MAVs. Our method uses a cascaded control scheme, where the attitude is controlled via the geometric attitude controller in [29], which presents a large region of attraction (initial attitude error should be <180 deg). This controller is additionally modified with a parametric adaptation scheme, where a torque observer estimates and compensates for the effects of slowly varying torque disturbances. Agile, robust, and real-time implementable trajectory tracking is achieved by employing a computationally efficient deep-NN policy, trained to imitate the response of a RTMPC, given a desired trajectory and the current state of the robot. The NN policy is obtained using our recent Imitation Learning (IL) method [2], which uses a high-fidelity simulator and properties of the controller to generate training data. A key benefit of our method [2] is the ability to train a computationally efficient policy in a computationally efficient way (e.g., a new policy can be obtained in a few minutes), greatly accelerating the tuning phase of the NN controller. We experimentally evaluate our robust, agile trajectory tracking approach on the MIT sub-gram-MAV SoftFly [3], demonstrating that our method can consistently achieve low position tracking error on a variety of trajectories, which include a circular trajectory (Figure 1) and a ramp, while running at 2 kHz on a Baseline Target Machine, SpeedGoat offboard computer. We additionally demonstrate that our strategy is robust to large external disturbances, intentionally applied while the robot tracks a given trajectory. In summary, our work presents the following **contributions**:

- We present the first computationally-efficient strategy for robust, MPC-like control of sub-gram MAVs. Our approach employs a deep-learned NN policy that is trained to reproduce a trajectory tracking RTMPC, leveraging our previous IL work [2].
- We present a cascaded control strategy, where the attitude controller in [29] is modified with a model adaptation method to compensate for the effects of uncertainties.
- We perform an experimental evaluation on the MIT SoftFly [3], an agile sub-gram MAV (0.7 g), showing a 60% reduction in maximum trajectory tracking errors over [3], while being real-time implementable (2 kHz) on a small computational platform.

II. Robot Design and Model

Reference frames. We consider an inertial reference frame $W = \{{}_W\mathbf{x}, {}_W\mathbf{y}, {}_W\mathbf{z}\}$ and a body-fixed frame $B = \{{}_B\mathbf{x}, {}_B\mathbf{y}, {}_B\mathbf{z}\}$ attached to the Center of Mass (CoM) of the robot, as shown in Figure 2.

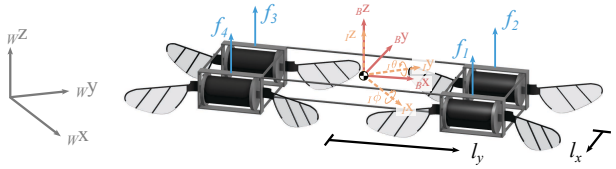

Fig. 2: CAD model of sub-gram MAV SoftFly that consists of four soft artificial muscles (DEAs). We additionally show the following reference frames: inertial W (grey), body-fixed B (red), yaw-fixed frame I (orange) used for control.

Mechanical design. The sub-gram flapping-wing robot (Figure 2) consists of four individually-controlled DEAs [1], [18] with newly-designed enhanced-endurance wing hinges [30]. Unlike natural flying insects that actively control wing stroke and pitch motions [31], our robot leverages system resonance (400 Hz) and passive fluid-wing interaction to generate lift forces and support flight. [1], [18]. This design allows each of the four robot modules to generate lift forces without producing significant torques.

Actuation model. The voltage inputs to the actuator are controlled to produce desired time-averaged lift forces, and a linear voltage-to-lift-force mapping, $f_i = \alpha_i v_i + \beta_i$, is implemented as previously shown in [1], [18]. The time-averaged lift force f_i produce by each actuator i, with $i = 1, \ldots, 4$, is utilized in the model for control purpose since the wing inertia is order of magnitude smaller than that of the robot thorax. These forces can be mapped to the total torque produced by the actuators $\tau_{\text{cmd},x}$ and $\tau_{\text{cmd},y}$ (around ${}_B\mathbf{x}$ and ${}_B\mathbf{y}$, respectively) and f_{cmd} as the total thrust force on ${}_B\mathbf{z}$ via a linear mapping (*mixer* or *allocation* matrix) $\mathcal{A}$:

$$\begin{bmatrix} f_{\text{cmd}} \\ \tau_{\text{cmd},x} \\ \tau_{\text{cmd},y} \end{bmatrix} = \mathcal{A} \begin{bmatrix} f_1 \\ \vdots \\ f_4 \end{bmatrix}, \; \mathcal{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -l_y & l_y & l_y & -l_y \\ -l_x & -l_x & l_x & l_x \end{bmatrix}, \quad (1)$$

where l_x, l_y represent the distance of the actuators from the CoM of the robot, as shown in Figure 2. This configuration (actuators' placement and near-resonant-frequency operating condition) does not generate controlled torques with respect to the body z-axis.

Translational and rotational dynamics. The MAV is modeled as a rigid body with six degrees of freedom, with mass m and diagonal inertia tensor $\mathbf{J}$, subject to gravitational acceleration g. The following set of Newton-Euler equations describes the robot's dynamics:

$$\begin{aligned} m\,\dot{\mathbf{v}} &= f_{\text{cmd}}\mathbf{R}\,{}_B\mathbf{z} - mg\,{}_W\mathbf{z} + \mathbf{f}_{\text{drag}} + \mathbf{f}_{\text{ext}}, \\ \mathbf{J}\,\dot{\boldsymbol{\omega}} &= -\,\boldsymbol{\omega} \times \mathbf{J}\,\boldsymbol{\omega} + \boldsymbol{\tau}_{\text{cmd}} + \boldsymbol{\tau}_{\text{drag}} + \boldsymbol{\tau}_{\text{ext}}, \\ \dot{\mathbf{p}} &= \mathbf{v}, \\ \dot{\mathbf{R}} &= \mathbf{R}\,\boldsymbol{\omega}^{\wedge}. \end{aligned} \quad (2)$$

Position $\mathbf{p} \in \mathbb{R}^3$ and velocity $\mathbf{v} \in \mathbb{R}^3$ are expressed in W; a rotation matrix $\mathbf{R} \in SO(3)$ defines the attitude, and the angular velocity $\boldsymbol{\omega}$ is expressed in B; $\boldsymbol{\omega}^{\wedge}$ denotes the skew-symmetric matrix of $\boldsymbol{\omega}$. We assume that the dynamics are affected by external force and torque $\mathbf{f}_{\text{ext}} \in \mathbb{R}^3$ and $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^3$, capturing the effects of unknown disturbances, such as the forces/torques applied by the power tethers, imperfections in the assembly and mismatches of model parameters (e.g, mass). Assuming no wind in the environment, we also include

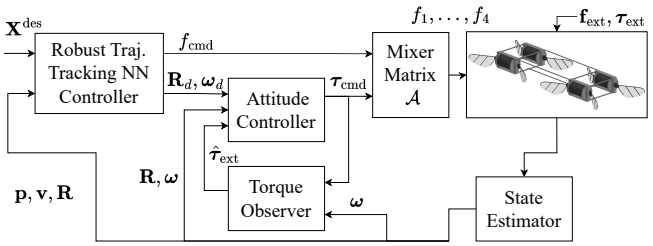

Fig. 3: Cascaded architecture for the proposed robust trajectory tracking control strategy. The robust trajectory tracking NN controller constitutes the *outer* loop, and its task is to track a desired trajectory $\mathbf{x}_0^{\text{des}}, \ldots, \mathbf{x}_N^{\text{des}}$ by generating setpoints $\mathbf{R}_d$, $\boldsymbol{\omega}_d$ for the cascaded attitude controller. The attitude controller and the torque observer constitute the *inner* loop. Thanks to the robustness and adaptive properties of the *outer* and *inner* loops, our approach can withstand the external force/torque disturbances $\mathbf{f}_{\text{ext}}, \boldsymbol{\tau}_{\text{ext}}$.

an isotropic drag force $\mathbf{f}_{\text{drag}} = -c_{Dv}\,\mathbf{v}$ and torque $\boldsymbol{\tau}_{\text{drag}} = -c_{D\omega}\,\boldsymbol{\omega}$, with $c_{Dv} > 0, c_{D\omega} > 0$.

III. Flight Control Strategy

We decouple trajectory tracking and attitude control via a cascaded scheme, as shown in Figure 3. Given an $N+1$-step reference trajectory $\mathbf{X}^{\text{des}}$, a trajectory tracking controller generates desired thrust f_{cmd}, attitude $\mathbf{R}_d$ and angular velocity $\boldsymbol{\omega}_d$ setpoints. A nested attitude controller then tracks the attitude commands by generating a desired torque $\boldsymbol{\tau}_{\text{cmd}}$ and by leveraging the estimated torque disturbance $\hat{\boldsymbol{\tau}}_{\text{ext}}$ provided by a torque observer. The commands $\boldsymbol{\tau}_{\text{cmd}}, f_{\text{cmd}}$ are converted (in the Mixer Matrix) to desired mean lift forces $f_1, \ldots, f_4$ using the Moore-Penrose inverse $\mathcal{A}^\dagger$ of (1).

In the following paragraphs, we describe, first, the adaptive attitude controller (Section III-A). Then, we present the computationally expensive trajectory-tracking RTMPC (Section III-B) and, last, the computationally efficient procedure to generate the computationally efficient, robust NN tracking policy (Section III-C) used to control the real robot.

A. Attitude Control and Adaptation Strategy

Attitude control law. The control law employed to regulate the attitude of the robot is based on the Geometric attitude controller in [29]:

$$\begin{aligned}\boldsymbol{\tau}_{\text{cmd}} = &-\mathbf{K}_R\mathbf{e}_R - \mathbf{K}_\omega\mathbf{e}_\omega + \boldsymbol{\omega}\times\mathbf{J}\,\boldsymbol{\omega} \\ &- \mathbf{J}(\boldsymbol{\omega}^\wedge\mathbf{R}^\top\mathbf{R}_d\boldsymbol{\omega}_d - \mathbf{R}^\top\mathbf{R}_d\,\dot{\boldsymbol{\omega}}_d) - \hat{\boldsymbol{\tau}}_{\text{ext}},\end{aligned} \tag{3}$$

where $\mathbf{K}_R, \mathbf{K}_\omega$ of size 3×3 are diagonal matrices, tuning parameters of the controller, and the attitude error $\mathbf{e}_R$ and its time derivative $\mathbf{e}_\omega$ are defined as in [29]:

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_d^\top\mathbf{R} - \mathbf{R}^\top\mathbf{R}_d)^\vee,\ \mathbf{e}_\omega = \boldsymbol{\omega} - \mathbf{R}^\top\mathbf{R}_d\,\boldsymbol{\omega}_d. \tag{4}$$

The symbol $(\mathbf{r}^\wedge)^\vee = \mathbf{r}$ denotes the operation transforming a 3×3 skew-symmetric matrix $\mathbf{r}^\wedge$ in a vector $\mathbf{r}\in\mathbb{R}^3$. Different from [29], we assume $\dot{\boldsymbol{\omega}}_d = \mathbf{0}_3$. While this could result in larger tracking errors under very aggressive attitude changes, it avoids taking derivatives of potentially discontinuous angular velocity commands, reducing actuation noise. We additionally augment (3) with the adaptive term $\hat{\boldsymbol{\tau}}_{\text{ext}}$, which is computed via a torque observer. We note that only the first two components of $\boldsymbol{\tau}_{\text{cmd}}$ are used for control, as the actuators cannot produce torque $\tau_{\text{cmd},z}$.

Torque observer. We compensate for the effects of uncertainties in the rotational dynamics by estimating torque disturbances $\boldsymbol{\tau}_{\text{ext}}$ via a steady state Kalman filter. These disturbances are assumed to be slowly varying when expressed in the body frame B. The state of the filter is $\mathbf{x}_o = [\boldsymbol{\omega}^\top, \boldsymbol{\tau}_{\text{ext}}^\top]^\top$. We assume that the rotational dynamics, employed to compute the prediction (*a priori*) step, evolve according to:

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\mathbf{u}_o + \boldsymbol{\tau}_{\text{ext}}) + \boldsymbol{\eta}_\omega, \quad \dot{\boldsymbol{\tau}}_{\text{ext}} = \boldsymbol{\eta}_\tau, \tag{5}$$

where $\boldsymbol{\eta}_\omega$, $\boldsymbol{\eta}_\tau$ are assumed to be zero-mean Gaussian noise, whose covariance is a tuning parameter of the filter. Additionally, the prediction step is performed assuming the control input $\mathbf{u}_o = \boldsymbol{\tau}_{\text{cmd}} - \boldsymbol{\omega}\times\mathbf{J}\,\boldsymbol{\omega}$, which enables us to take into account the nonlinear gyroscopic effects $\boldsymbol{\omega}\times\mathbf{J}\,\boldsymbol{\omega}$ while using a computationally efficient linear observer. The measurement update (a *posteriori*) uses angular velocity measurements $\mathbf{z}_o = \boldsymbol{\omega}_m + \boldsymbol{\eta}_m$, assumed corrupted by an additive zero mean Gaussian noise $\boldsymbol{\eta}_m$, whose covariance can be identified from data or adjusted as a tuning parameter.

B. Robust Tube MPC for Trajectory Tracking

In this part, first we present the hover-linearized vehicle model employed for control design (Section III-B.1). We then present the optimization problem solved by a linear RTMPC (Section III-B.2) and the compensation schemes to account for the effects of linearization (Section III-B.3).

1) Linearized Model: We linearize the dynamics (2) around hover using a procedure that largely follows [25], [32]. The key differences are highlighted in the following.

First, for interpretability, we represent the attitude of the MAV via the Euler angles yaw ψ, pitch θ, roll ϕ (*intrinsic* rotations around the z-y-x). The corresponding rotation matrix can be obtained as $\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$, where $\mathbf{R}_j(\alpha)$ denotes a rotation of α around the j-th axis. Additionally, we express the dynamics (2) in a yaw-fixed frame I, so that ${}_I\mathbf{x}$ is aligned with ${}_W\mathbf{x}$. The roll ${}_I\phi$ and pitch ${}_I\theta$ angles (and their first derivative ${}_I\varphi$, ${}_I\vartheta$) expressed in I can be expressed in B via the rotation matrix $\mathbf{R}_{BI}$:

$$\begin{bmatrix}\phi\\ \theta\end{bmatrix} = \mathbf{R}_{BI}\begin{bmatrix}{}_I\phi\\ {}_I\theta\end{bmatrix},\ \mathbf{R}_{BI} = \begin{bmatrix}\cos(\psi) & \sin(\psi)\\ -\sin(\psi) & \cos(\psi)\end{bmatrix}. \tag{6}$$

The state $\mathbf{x}$ of the linearized model is chosen to be:

$$\mathbf{x} = [\mathbf{p}^\top, \mathbf{v}^\top, {}_I\phi, {}_I\theta, {}_I\delta\phi_{\text{cmd}}, {}_I\delta\theta_{\text{cmd}}]^\top, \tag{7}$$

where ${}_I\delta\phi_{\text{cmd}}, {}_I\delta\theta_{\text{cmd}}$ denote the linearized commanded attitude. We chose the control input $\mathbf{u}$ to be:

$$\mathbf{u} = [{}_I\varphi_{\text{cmd}}, {}_I\vartheta_{\text{cmd}}, \delta f_{\text{cmd}}]^\top, \tag{8}$$

where δf_{cmd} denotes the linearized commanded thrust, and ${}_I\varphi_{\text{cmd}}$ and ${}_I\vartheta_{\text{cmd}}$ are the commanded roll and pitch rates. The choice of $\mathbf{x}$ and $\mathbf{u}$ differs from [2], [25] as the control input consist in the roll, pitch rates rather than $\theta_{\text{cmd}}, \phi_{\text{cmd}}$; this is done to avoid discontinuities in $\theta_{\text{cmd}}, \phi_{\text{cmd}}$, and to feedforward angular velocity commands to the attitude controller.

The linear translational dynamics are obtained by linearization of the translational dynamics in (2) around hover. Linearizing the closed-loop rotational dynamics is more challenging, as they should include the linearization of the attitude controller. Following [32], we model the closed-loop

attitude dynamics around hover expressed in I as:

$$
\begin{aligned}
{}_I\dot{\theta} &= \frac{1}{\tau_\theta}(k_\theta\, {}_I\theta_{\text{cmd}} - {}_I\theta), \quad {}_I\dot{\theta}_{\text{cmd}} = {}_I\vartheta_{\text{cmd}}, \\
{}_I\dot{\phi} &= \frac{1}{\tau_\phi}(k_\phi\, {}_I\phi_{\text{cmd}} - {}_I\phi), \quad {}_I\dot{\phi}_{\text{cmd}} = {}_I\varphi_{\text{cmd}},
\end{aligned} \tag{9}
$$

where k_ϕ, k_θ are gains of the commanded roll and pitch angles, while τ_ϕ, τ_θ are the respective time constants. These parameters can be obtained via system identification.

Last, we model the unknown external force disturbance $\mathbf{f}_{\text{ext}}$ in (2) as a source of bounded uncertainty, assumed to be $\|\mathbf{f}_{\text{ext}}\|_\infty < \bar{f}_{\text{ext}}$. This introduces an additive bounded uncertainty $\mathbf{w} \in \mathbb{W}$, with:

$$
\mathbb{W} := \{\mathbf{w} = [\mathbf{0}_3^\top, \mathbf{f}_{\text{ext}}^\top, \mathbf{0}_4^\top]^\top \mid \|\mathbf{f}_{\text{ext}}\|_\infty < \bar{f}_{\text{ext}}\}. \tag{10}
$$

Via discretization with sampling period T_c, we obtain the following linear, uncertain state space model:

$$
\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_k, \tag{11}
$$

subject to actuation and state constrains $\mathbb{U} = \{\mathbf{u} \in \mathbb{R}^3 | \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}\}$, and $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^{10} | \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max}\}$.

2) Controller Formulation: The trajectory tracking RTMPC is based on [28], with the objective function modified to track a desired trajectory.

Optimization problem. At every timestep t, RTMPC takes as input the current state of the robot $\mathbf{x}_t$ and a $N+1$-step desired trajectory $\mathbf{X}_t^{\text{des}} = \{\mathbf{x}_{0|t}^{\text{des}}, \ldots, \mathbf{x}_{N|t}^{\text{des}}\}$, where $\mathbf{x}_{i|t}^{\text{des}}$ indicates the desired state at the future time $t+i$, as given at the current time t. It then computes a sequence of reference states $\bar{\mathbf{X}}_t = \{\bar{\mathbf{x}}_{0|t}, \ldots, \bar{\mathbf{x}}_{N|t}\}$ and actions $\bar{\mathbf{U}}_t = \{\bar{\mathbf{u}}_{0|t}, \ldots, \bar{\mathbf{u}}_{N-1|t}\}$, that will not cause constrain violations ("safe") regardless of the realization of $\mathbf{w} \in \mathbb{W}$. This is achieved by solving:

$$
\begin{aligned}
\bar{\mathbf{U}}_t^*, \bar{\mathbf{X}}_t^* \;=\; & \underset{\bar{\mathbf{U}}_t, \bar{\mathbf{X}}_t}{\operatorname{argmin}} \|\mathbf{e}_{N|t}\|_{\mathbf{P}_x}^2 + \sum_{i=0}^{N-1} \|\mathbf{e}_{i|t}\|_{\mathbf{Q}_x}^2 + \|\mathbf{u}_{i|t}\|_{\mathbf{R}_u}^2 \\
\text{subject to } & \bar{\mathbf{x}}_{i+1|t} = \mathbf{A}\bar{\mathbf{x}}_{i|t} + \mathbf{B}\bar{\mathbf{u}}_{i|t}, \\
& \bar{\mathbf{x}}_{i|t} \in \mathbb{X} \ominus \mathbb{Z}, \;\; \bar{\mathbf{u}}_{i|t} \in \mathbb{U} \ominus \mathbf{K}\mathbb{Z}, \\
& \mathbf{x}_t \in \mathbb{Z} \oplus \bar{\mathbf{x}}_{0|t}, \; i = 0, \ldots, N-1
\end{aligned} \tag{12}
$$

where $\mathbf{e}_{i|t} = \bar{\mathbf{x}}_{i|t} - \mathbf{x}_{i|t}^{\text{des}}$ is the tracking error. The positive definite matrices $\mathbf{Q}_x$, $\mathbf{R}_u$ define the trade-off between deviations from the desired trajectory and actuation usage, while $\|\mathbf{e}_{N|t}\|_{\mathbf{P}_x}^2$ is the terminal cost. $\mathbf{P}_x$ and $\mathbf{K}$ are obtained by formulating an infinite horizon optimal control LQR problem using $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}_x$ and $\mathbf{R}_u$ and by solving the associated algebraic Riccati equation [33]. $\oplus$, $\ominus$ denote set addition (*Minkowski sum*) and subtraction (*Pontryagin difference*). As often done in practice [32], we do not include a terminal set constraints, but we make sure to select a sufficiently long prediction horizon to achieve recursive feasibility.

Tube and ancillary controller. A control input for the real system is generated by RTMPC via an *ancillary controller*:

$$
\mathbf{u}_t = \bar{\mathbf{u}}_t^* + \mathbf{K}(\mathbf{x}_t - \bar{\mathbf{x}}_t^*), \tag{13}
$$

where $\bar{\mathbf{u}}_{0|t}^* = \bar{\mathbf{u}}_t^*$ and $\bar{\mathbf{x}}_{0|t}^* = \bar{\mathbf{x}}_t^*$. This controller ensures that the system remains inside a *tube* (with "cross-section" $\mathbb{Z}$) centered around $\bar{\mathbf{x}}_t^*$ regardless of the realization of the disturbances in $\mathbb{W}$, provided that the state of the system starts in such tube (constraint $\mathbf{x}_t \in \mathbb{Z} \oplus \bar{\mathbf{x}}_{0|t}$). The set $\mathbb{Z}$ is a disturbance invariant set for the closed-loop system $\mathbf{A}_K := \mathbf{A} + \mathbf{B}\mathbf{K}$, satisfying the property that $\forall \mathbf{x}_t \in \mathbb{Z}$, $\forall \mathbf{w}_t \in \mathbb{W}$, $\forall t \in \mathbb{N}^+$, $\mathbf{x}_{t+1} = \mathbf{A}_K\mathbf{x}_t + \mathbf{w}_t \in \mathbb{Z}$. In our work, we compute an approximation of $\mathbb{Z}$ by computing the largest deviations from the origin of $\mathbf{A}_K$ by performing Monte-Carlo simulations, with disturbances uniformly sampled from $\mathbb{W}$.

3) Compensation schemes and attitude setpoints: Following [32], we apply a compensation scheme to the commands:

$$
f_{\text{cmd}} = \frac{\delta f_{\text{cmd}} + g}{\cos(\phi)\cos(\theta)}, \quad \begin{bmatrix}\phi_{\text{cmd}} \\ \theta_{\text{cmd}}\end{bmatrix} = \frac{g}{f_{\text{cmd}}} \begin{bmatrix}\delta\phi_{\text{cmd}} \\ \delta\theta_{\text{cmd}}\end{bmatrix}. \tag{14}
$$

This desired orientation θ_{cmd}, ϕ_{cmd} is converted to a desired rotation matrix $\mathbf{R}_d$, setpoint for the attitude controller, setting the desired yaw angle to the current yaw angle $\psi_{\text{cmd}} = \psi$. Last, the desired angular velocity $\boldsymbol{\omega}_d$ is computed from the desired yaw, pitch, roll rates $\dot{\mathbf{q}} = [\dot{\psi}_{\text{cmd}}, \vartheta_{\text{cmd}}, \varphi_{\text{cmd}}]^\top$ via [34] $\boldsymbol{\omega}_d = \mathbf{E}_{\psi,\theta,\phi}\dot{\mathbf{q}}$, with

$$
\mathbf{E}_{\psi,\theta,\phi} = \begin{bmatrix} 0 & -\sin(\psi) & \cos(\psi)\sin(\theta) \\ 0 & \cos(\psi) & \sin(\psi)\cos(\theta) \\ 1 & 0 & -\sin(\theta) \end{bmatrix}, \tag{15}
$$

and assuming the desired yaw rate $\dot{\psi}_{\text{cmd}} = 0$.

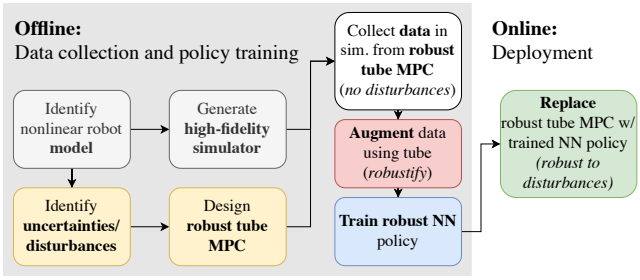

Fig. 4: Imitation learning strategy employed to learn a robust neural network (NN) from robust tube model predictive controller (RTMPC). Key idea is to collect RTMPC demonstrations in simulation and to leverage properties of RTMPC to augment the collected demonstrations with data that improve the robustness of the trained policy.

C. Robust Tracking NN Policy

The procedure to generate a computationally efficient NN policy capable of reproducing the response of the trajectory tracking RTMPC in Section III-B is summarized in Figure 4.

Policy input-output. The deep NN policy that we intend to train is denoted as π_θ, with parameters θ. Its input-outputs are the same as the ones of RTMPC in Section III-B:

$$
\mathbf{u}_t = \pi_\theta(\mathbf{x}_t, \mathbf{X}_t^{\text{des}}). \tag{16}
$$

Policy training. Our approach, based on our previous work [2], consists in the following steps:

1) We design a high-fidelity simulator implementing a discretized model (discretization period T_s) of the nonlinear dynamics (2) and the control architecture consisting of the attitude controller (Section III-A) and RTMPC (Section III-B). In the simulator, we assume that the MAV is not subject to disturbances, setting $\mathbf{f}_{\text{ext}} = \mathbf{0}_3$ and $\boldsymbol{\tau}_{\text{ext}} = \mathbf{0}_3$, and therefore we do not simulate the torque observer.

2) Given a desired trajectory, we collect a $T+1$-step *demonstration* $\mathcal{T}$ by simulating the entire system con-

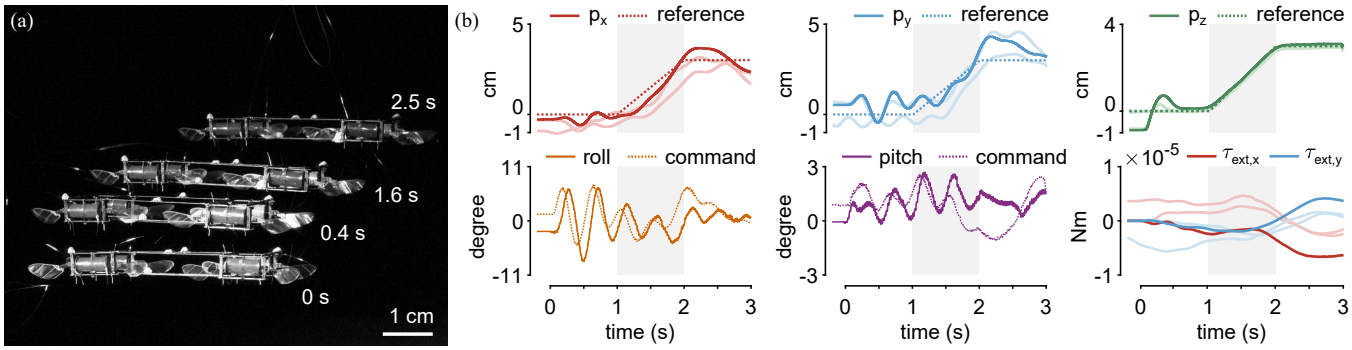

Fig. 5: Performance of the proposed flight control strategy in Task 1 (T1), where the robot follows a 3.5 s ramp trajectory on x-y-z. The experiment is repeated three times, and the results are included in the shaded lines, with the exception of the roll and pitch angles, for clarity. These results highlight the accuracy of the proposed trajectory tracking error, which achieves a 0.6 cm position tracking RMSE on x-y, and 0.2 cm on z.

TABLE I: Position Root Mean Squared Errors (RMSE) and Maximum Absolute Errors (MAE) when tracking a ramp (T1), a ramp with disturbances (T2) and a circle (T3). All the values are computed after takeoff ($t > 0.5$ s).

	T1: Ramp (3 runs, 2.5 s)						**T2 (1 run, 7.0 s)**		**T3: Circle (3 runs, 7.5 s)**					
	RMSE (cm, ↓)			MAE (cm, ↓)			RMSE	MAE	RMSE (cm, ↓)			MAE (cm, ↓)		
Axis	AVG	MIN	MAX	AVG	MIN	MAX	(cm, ↓)	(cm, ↓)	AVG	MIN	MAX	AVG	MIN	MAX
x	0.6	0.4	0.9	1.1	0.7	1.5	0.7	2.5	1.0	0.8	1.4	2.0	1.7	2.6
y	0.8	0.7	0.9	1.5	1.3	1.6	1.0	2.1	1.5	1.3	1.8	2.8	2.5	3.1
z	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.6	0.5	0.8

trolled by RTMPC. At every timestep t of the demonstration, we store the inputs, output of RTMPC, with the addition of the safe plans $\bar{\mathbf{u}}_t^*, \bar{\mathbf{x}}_t^*$, obtaining $\mathcal{T} = \{(\mathbf{x}_0, \mathbf{u}_0, \bar{\mathbf{u}}_0^*, \bar{\mathbf{x}}_0^*, \mathbf{X}_0^{\text{des}}), \dots, (\mathbf{x}_T, \mathbf{u}_T, \bar{\mathbf{u}}_T^*, \bar{\mathbf{x}}_T^*, \mathbf{X}_T^{\text{des}})\}$.

3) We generate a dataset of (inputs, output) of the controller, using the data obtained from the collected demonstration $\mathcal{T}$. The obtained dataset $\mathcal{D}$ has the form $\mathcal{D} = \{(\{\mathbf{x}_0, \mathbf{X}_0^{\text{des}}\}, \mathbf{u}_0), \dots, (\{\mathbf{x}_T, \mathbf{X}_T^{\text{des}}\}, \mathbf{u}_T)\}$.

4) For every timestep t in $\mathcal{T}$, we augment the dataset $\mathcal{D}$ by generating N_{augm} extra (state, action) pairs $(\mathbf{x}_{i,t}^+, \mathbf{u}_{i,t}^+)$ by uniformly sampling extra states from the tube ($\mathbf{x}_{i,t}^+ \in \bar{\mathbf{x}}_t^* \oplus \mathbb{Z}$), and by computing the corresponding actions $\mathbf{u}_{i,t}^+$ with the ancillary controller:

$$\mathbf{u}_{i,t}^+ = \bar{\mathbf{u}}_t^* + \mathbf{K}(\mathbf{x}_{i,t}^+ - \bar{\mathbf{x}}_t^*). \tag{17}$$

The extra datasets $\mathcal{D}_t^+ = \{(\{\mathbf{x}_{i,t}^+, \mathbf{X}_{i,t}^{\text{des}}\}, \mathbf{u}_{i,t}^+)_{i=1}^{N_{\text{augm}}}\}$, $t = 0, \dots, T$ are then combined with $\mathcal{D}$, obtaining the training dataset. This data augmentation procedure [2] generates data that can compensate for the effects of uncertainties in $\mathbb{W}$. This procedure has also the potential to reduce the time needed for the data collection phase over other existing IL methods, as (17) can be computed efficiently. We note that step $2-4$ should be repeated for every possible trajectory the policy needs to learn to follow, using IL methods such as Behavior Cloning (BC) or Dataset-Aggregation (DAgger) [2], [35]. However, as shown in our previous work [2], if a set of sufficiently representative trajectories is collected during training, then the policy is capable to execute new slightly different trajectories, achieving generalization capabilities.

5) The optimal parameters θ^* for the policy (16) are then found by training the policy on the collected and augmented dataset, minimizing the Mean Squared Error (MSE) loss.

IV. Experimental Evaluation

The robustness and performance of the described flight controllers are experimentally evaluated on the soft-actuated MIT SoftFly [3]. We consider two trajectories of increasing difficulty, a position ramp, where we additionally perturb the MAV with external disturbances, and a circular trajectory.

Experimental setup. The flight experiments are performed in an environment equipped with 6 motion-capturing cameras (Vantage V5, Vicon). This system provides positions and orientations of the robot, and velocities are obtained via numerical differentiation. The controller runs at 2 kHz on the Baseline Target Machine (Speedgoat) using the Simulink Real-Time operating system; its commands are converted to sinusoidal signals for flapping motion at 10 kHz. Voltage amplifiers ($677B$, Trek) are connected to the controller and produce amplified control voltages to the robot.

Controller and training parameters. The parameters for the controllers are obtained via system identification, also leveraging [36], [37]. The RTMPC has a 1-second long prediction horizon, with $N = 50$ and $T_c = 0.02$; we set $\bar{f}_{\text{ext}}$ to correspond to 15% of the weight force acting on the robot. State and actuation constraints capture safety and actuation limits of the robot (e.g., max actual/commanded roll/pitch $< 25\,\text{deg}$, $\|\delta f_{\text{cmd}}\| < 80\% mg$). The employed policy is a feed-forward, fully connected NN with 2-hidden layers, 32 neurons per layer (as in our previous work [2], where this size has been shown to be capable to learn multiple trajectories). Its input size is 310 (current state, and reference trajectory containing desired position, velocity across the prediction horizon N), and its output size is 3. During the experiments, we slightly tune the parameters $\mathbf{Q}_x, \mathbf{R}_u, \mathbf{K}_R, \mathbf{K}_\omega$ to study how sensitive is our approach to tuning changes, without observing large performance variations. We train a policy for each type of trajectory (ramp, circle), using the ADAM optimizer, with a learning rate $\eta = 0.001$, for 15 epochs. Data augmentation is performed by using $N_{\text{augm}} = 200$. Training each policy takes about 1 minute (for $T = 350$ steps) on a Intel i9-10920 (12 cores) with two Nvidia RTX 3090 GPUs.

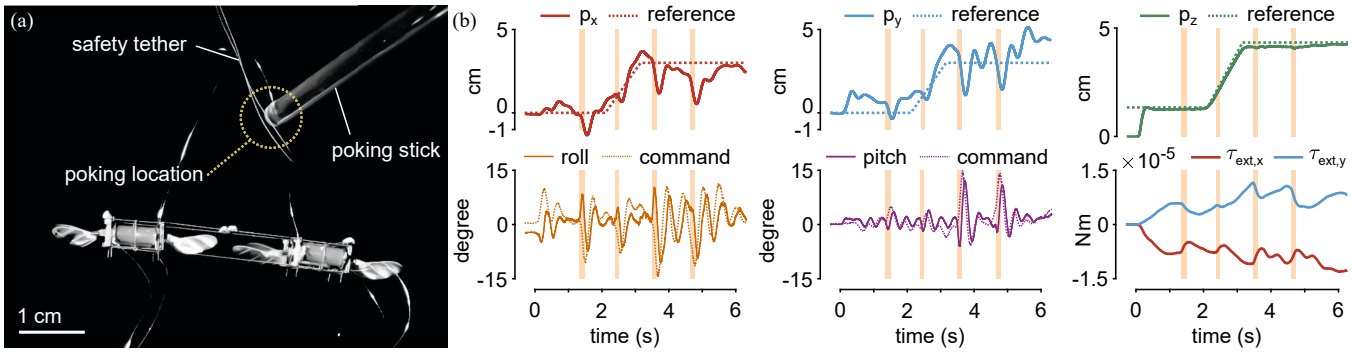

Fig. 6: Robustness of the proposed flight control strategy when applying large force/torque disturbances to the MAV while tracking a 6.0 s ramp trajectory on x-y-z, our Task 2 (T2). These results highlight that the robot is not destabilized by the large disturbances, achieving an impressive 0.3 cm MAE on the z axis. The fast attitude dynamics additionally enable the robot to recover in only 200 ms.

A. Task 1: Position Ramps

First, we consider a position ramp of 3 cm along the three axes of the W, with a duration of 1 s. The total flight time in the experiment is of 3 s, and we repeat the experiment three times. This is a task of medium difficulty, as the robot needs to simultaneously roll, pitch and accelerate along z, but the maneuver covers a small distance (5.2 cm). Figure 5 (a) shows a time-lapse of the maneuver. Table I reports the position tracking RMSE (computed after take-off, starting at $t_0 = 0.5$), and shows that we can consistently achieve sub-centimeter RMSE on all the axes, with a maximum absolute error (MAE) smaller than 1.6 cm. This is a 60% reduction over the 4.0 cm MAE on x-y reported in [3] for a hover task. Remarkably, the altitude MAE is only 0.2 cm, with a similar reduction (60%) over [3] (0.5 cm). Figure 5 shows the robot's desired and actual position and attitude across multiple runs (shaded lines), demonstrating repeatability. Figure 5 additionally highlights the role of the torque observer, which estimates a position-dependent disturbance, possibly caused by the forces applied by the power cables, or by the safety tether.

B. Task 2: Rejection of Large External Disturbances

Next, we increase the complexity of the task by intentionally applying strong disturbances with a stick to the safety tether (Figure 6 (a)), while tracking the same ramp trajectory in Task 1. This causes accelerations > 0.25 g. Figure 6 reports position, attitude, and estimated disturbances, where the contact periods have been highlighted in orange. Table I reports RMSE and MAE. From Figure 6 (b) we highlight that a) the position of the robot remains close to the reference (< 2.5 cm MAE on x-y) and, surprisingly, the altitude is almost unperturbed (0.3 cm MAE). b) the robot, thanks to its small inertia, recovers quickly from the large disturbances, being capable to permanently reduce (after impact, until the next impact) its acceleration by 50% in less than 200 ms; c) despite the impulse-like nature of the applied disturbance, the torque observer detects some of its effects.

C. Task 3: Circular Trajectory

Last, we track a circular trajectory along the x-z axis of W for three times. The trajectory has a duration of 7.5 s (including takeoff), with a desired velocity of 5.2 cm/s, and the circle has a radius of 5.0 cm. A composite image of the experiment is shown in Figure 1. Figure 7 provides a qualitative evaluation of the performance of our approach, while Table I reports the RMSE of position tracking across the three runs. These results highlight that: a) in line with Task 1, our approach consistently achieves mm-level accuracy in altitude tracking (3 mm average RMSE on z), and cm-level accuracy in x-y position tracking (RMSE < 1.8 cm, MAE < 3.1 cm); b) the external torque observer plays an important role in detecting and compensating external torque disturbances, which corresponds to approximately 30% of the maximum torque control authority around ${}_B x$.

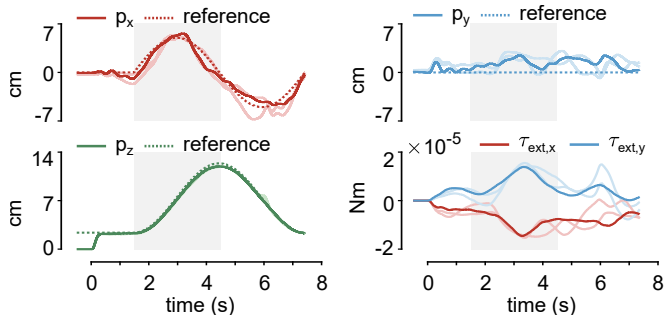

Fig. 7: Performance in Task 3 (T3), where the robot tracks a long (7.5 s) circular trajectory, with a wide (5.0 cm) radius. The experiment has been repeated three times, and the results are included in the shaded lines. These results highlight the high and repeatable accuracy during agile flight of the RTMPC-like NN policy controlling the robot, as well as the role of the torque observer, which is capable to estimate and compensate for the effects of large rotational uncertainties. A composite image of the trajectory is shown in Figure 1.

V. Conclusions

This work has presented the first robust MPC-like neural network policy capable of experimentally controlling a sub-gram MAV [3]. In our experimental evaluations, the proposed policy achieved high control rates (2 kHz) on a small offboard computer, while demonstrating small (< 1.8 cm) RMS tracking errors, and the ability to withstand large external disturbances. These results open up novel and exciting opportunities for agile control of sub-gram MAVs. First, the demonstrated robustness and computational efficiency paves the way for onboard deployment under real-world uncertainties. Second, the newly-developed trajectory tracking capabilities enable data collection at different flight regimes, for model discovery and identification [36], [37]. Last, as the computational cost of a learned neural-network policy grows favorably with respect to state size, we can use larger, more sophisticated models for control design, further exploiting the nimble characteristics of sub-gram MAVs.

REFERENCES

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

[2] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How, "Demonstration-efficient guided policy search via imitation of robust tube mpc," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 462–468.

[3] Y. Chen, S. Xu, Z. Ren, and P. Chirarattananon, "Collision resilient insect-scale soft-actuated aerial robots with high agility," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1752–1764, 2021.

[4] P. Liu, S. P. Sane, J.-M. Mongeau, J. Zhao, and B. Cheng, "Flies land upside down on a ceiling using rapid visually mediated rotational maneuvers," *Science advances*, vol. 5, no. 10, p. eaax1877, 2019.

[5] S. B. Fuller, A. D. Straw, M. Y. Peek, R. M. Murray, and M. H. Dickinson, "Flying drosophila stabilize their vision-based velocity controller by sensing wind with their antennae," *Proceedings of the National Academy of Sciences*, vol. 111, no. 13, pp. E1182–E1191, 2014.

[6] V. M. Ortega-Jimenez, R. Mittal, and T. L. Hedrick, "Hawkmoth flight performance in tornado-like whirlwind vortices," *Bioinspiration & biomimetics*, vol. 9, no. 2, p. 025003, 2014.

[7] L. Ristroph, A. J. Bergou, G. Ristroph, K. Coumes, G. J. Berman, J. Guckenheimer, Z. J. Wang, and I. Cohen, "Discovering the flight autostabilizer of fruit flies by inducing aerial stumbles," *Proceedings of the National Academy of Sciences*, vol. 107, no. 11, pp. 4820–4824, 2010.

[8] A. K. Dickerson, P. G. Shankles, N. M. Madhavan, and D. L. Hu, "Mosquitoes survive raindrop collisions by virtue of their low mass," *Proceedings of the National Academy of Sciences*, vol. 109, no. 25, pp. 9822–9827, 2012.

[9] A. M. Mountcastle and S. A. Combes, "Biomechanical strategies for mitigating collision damage in insect wings: structural design versus embedded elastic materials," *Journal of Experimental Biology*, vol. 217, no. 7, pp. 1108–1115, 2014.

[10] 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.

[11] X. Yang, Y. Chen, L. Chang, A. A. Calderón, and N. O. Pérez-Arancibia, "Bee+: A 95-mg four-winged insect-scale flying robot driven by twinned unimorph actuators," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4270–4277, 2019.

[12] Y. M. Chukewad, J. James, A. Singh, and S. Fuller, "Robofly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2025–2040, 2021.

[13] "Drones by the Numbers," Sep. 2022, [Online; accessed 11. Sep. 2022]. [Online]. Available: https://www.faa.gov/uas/resources/by_the_numbers

[14] "Flyability — Drones for indoor inspection and confined space," Sep. 2022, [Online; accessed 11. Sep. 2022]. [Online]. Available: https://www.flyability.com

[15] "Skydio 2+™ and X2™ – Skydio Inc." Sep. 2022, [Online; accessed 11. Sep. 2022]. [Online]. Available: https://www.skydio.com

[16] R. D'Andrea, "Guest editorial can drones deliver?" *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 647–648, 2014.

[17] "Voliro Airborne Robotics," Sep. 2020, [Online; accessed 11. Sep. 2022]. [Online]. Available: https://voliro.com

[18] Z. Ren, S. Kim, X. Ji, W. Zhu, F. Niroui, J. Kong, and Y. Chen, "A high-lift micro-aerial-robot powered by low-voltage and long-endurance dielectric elastomer actuators," *Advanced Materials*, vol. 34, no. 7, p. 2106757, 2022.

[19] N. Pérez-Arancibia, "Challenging path to creating autonomous and controllable microrobots," Apr. 2021, [Online; accessed 13. Sep. 2022]. [Online]. Available: https://youtu.be/4cdz2OPo9SI

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

[21] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[22] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic tube mpc for nonlinear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1655–1662.

[23] B. T. Lopez, "Adaptive robust model predictive control for nonlinear systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[24] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[25] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[26] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body mpc for a dynamically stable mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[27] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[28] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[29] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[30] Y.-H. Hsiao, S. Kim, Z. Ren, and Y. Chen, "Heading control of a long-endurance insect-scale aerial robot powered by soft artificial muscles," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*.

[31] M. H. Dickinson, F.-O. Lehmann, and S. P. Sane, "Wing rotation and the aerodynamic basis of insect flight," *Science*, vol. 284, no. 5422, pp. 1954–1960, 1999.

[32] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot operating system (ROS)*. Springer, 2017, pp. 3–39.

[33] K. J. Åström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[34] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[35] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[36] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[37] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, "Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control," *Proceedings of the Royal Society A*, vol. 478, no. 2260, p. 20210904, 2022.