Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Inverting Learned Dynamics Models for Aggressive Multirotor Control

Alexander Spitzer and Nathan Michael
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{spitzer, nmichael}@cmu.edu

***Abstract*—We present a control strategy that applies inverse dynamics to a learned acceleration error model for accurate multirotor control input generation. This allows us to retain accurate trajectory and control input generation despite the presence of exogenous disturbances and modeling errors. Although accurate control input generation is traditionally possible when combined with parameter learning-based techniques, we propose a method that can do so while solving the relatively easier non-parametric model learning problem. We show that our technique is able to compensate for a larger class of model disturbances than traditional techniques can and we show reduced tracking error while following trajectories demanding accelerations of more than $7\ \mathrm{m/s^2}$ in multirotor simulation and hardware experiments.**

I. Introduction

A. Motivation

In the last several years, aerial robotics has seen a surge in popularity, largely due to the increasing viability of applications [29, 19, 7]. Multirotors have been particularly well represented, due to their agility and versatility, and have additionally been a fruitful testbed for nonlinear controllers and trajectory generation strategies [25, 21, 33, 20].

Computing precise control inputs for a dynamical system often requires accurate knowledge of its dynamics. Van Nieuwstadt and Murray [38] showed how the concept of differential flatness can be used to generate control inputs that follow a given trajectory for *differentially flat* systems.

For a multirotor, differential flatness can be used to compute the exact inputs required to follow a specified trajectory in x, y, z, and yaw (See Mellinger and Kumar [28]). The computed control inputs are only accurate if the fixed dynamic model and its associated parameters, e.g. mass, inertia, etc., are correct. Often, this fixed dynamic model assumption fails and the estimated parameters are inaccurate. This results in suboptimal trajectory tracking performance.

One possible approach to alleviate this problem is to estimate the model parameters from vehicle trajectory data. This however, can be difficult, and is still suboptimal when the chosen parameterization cannot realize the true vehicle model. On the other hand, non-parametric error models are commonly used and relatively easy to learn but are not readily used in the differential flatness framework. In this work, we show how a non-parametric error model can be used to generate control inputs that follow a specified trajectory. We additionally provide an extension to the proposed approach that can deal with *input-dependent* model errors via numerical optimization. We validate the control input generation strategy both in simulation and through experiments on a quadrotor.

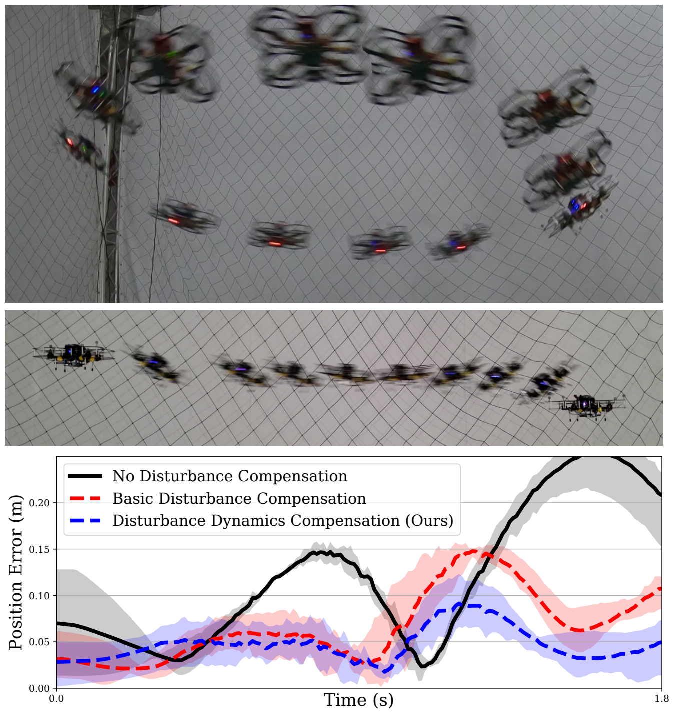


Fig. 1. Our experimental platform while executing an aggressive circle trajectory (top) and an aggressive line trajectory (middle) using the proposed control input generation strategy that is capable of compensating for dynamic and input-dependent acceleration disturbances (**FF5**). Our method substantially reduces tracking error along the aggressive line trajectory (bottom).

B. Related Works

Accurate and aggressive multirotor flight has been explored in [37, 30, 10, 28] among others. As for many other robotic platforms, accurate modeling has been shown to improve flight performance [36, 4]. Traditional non-learning based modeling can be achieved via hand crafted experiments, calibration procedures, and computer-aided design [27]. Since this requires significant manual effort and engineering hours, there have been many works exploring automatic parameter estimation methods [6, 5] and non-parametric model learning methods

[22, 8] for multirotor control. In this work, we focus on non-parametric model learning methods, since parameter learning methods can be limited in their accuracy by the choice of parameterization [31]. There has also been work on learning control input corrections for aggressive flight without learning a dynamical model [24]. These methods are not a focus of this paper since they can typically only be applied while executing the trained trajectories or reference quantities. A learned dynamical model can be applied to any trajectory or reference.

Non-parametric model learning methods for robot control have been employed in [13, 35, 1, 26]. Model learning performed in real-time incrementally has been studied in [16, 14, 3]. Florez et al. [13] use Locally Weighted Projection Regression (LWPR [39]) while Gijsberts and Metta [14] use Random Fourier Features [32], which was extended to Incremental Sparse Spectrum Gaussian Process Regression (ISSGPR), a Bayesian regression formulation, in Gijsberts and Metta [15]. Droniou et al. [9] evaluated LWPR and ISSGPR for the purposes of robot control and found ISSGPR to perform better. In this work, we use linear regression, but our approach can use any model learning strategy.

Once an accurate dynamical system model is known, a Model Predictive Control (MPC) strategy can be used to optimize a desired cost function, subject to the dynamics [8, 25, 23, 2]. These approaches often make approximations to ensure real time feasibility [8, 2]. Furthermore, Desaraju [8] does not perform full inverse dynamics on the disturbance, which can lead to suboptimal performance while tracking aggressive trajectories.

The differential flatness property of multirotors has been widely exploited for accurate trajectory tracking [11, 17, 28, 12, 34, 30]. Differential flatness of the multirotor subject to linear drag was shown in Faessler et al. [11]. This extends the applicability of the approach to a limited family of disturbances. Faessler et al. [11] do not address the issue of nonlinear disturbances as a function of state and/or control input in the flatness computations. Issues arising from singularities, commonly encountered during aggressive flight, were discussed and mitigated in Morrell et al. [30], increasing the robustness of the differential flatness approach.

Although control inputs computed using the differential flatness framework will automatically take into account dynamical model parameter changes, such as mass, inertia, etc., it is not clear how to incorporate non-parametric model corrections. In this work, we build on the differential flatness formulation by extending it to compensate for learned non-parametric dynamic model disturbances. Our approach can compensate for arbitrary disturbances that are a function of vehicle position and velocity, as well as control input dependent disturbances that are a function of vehicle orientation and thrust. This increases the applicability of the approach to a much wider range of realistic flight conditions.

C. Notation

Lowercase letters such as u and z are scalars in $\mathbb{R}$. Boldface lowercase letters such as $\boldsymbol{u}$ and $\boldsymbol{z}$ are vectors. $\boldsymbol{I}_n$ is the n by n identity matrix. $\dot{x}$ denotes the total time derivative of x. m is mass, g is the gravitational constant and I is inertia. All functions in this paper are assumed to have continuous second derivatives everywhere and thus all second partial derivatives are symmetric, i.e. $\frac{\partial f}{\partial x \partial y} = \frac{\partial f}{\partial y \partial x}$. Unless otherwise indicated, all vector quantities are expressed in a fixed reference frame.

II. METHOD

In this section, we first introduce the problem statement in Part *A*. Part *B* details our approach for compensating for dynamic disturbances that can be a function of vehicle position, vehicle velocity, or other quantities that are independent of the applied control inputs. Part *C* extends the approach to compensate for disturbances that are *input-dependent* and can be a function of e.g. the applied vehicle thrust or vehicle orientation. Finally, Part *D* describes the model learning approach.

A. Problem Statement

Assume we are given a desired position over time, $\boldsymbol{x}_d(t) \in \mathbb{R}^3$, along with its first four time derivatives, the velocity, acceleration, jerk, and snap: $\boldsymbol{v}_d(t)$, $\boldsymbol{a}_d(t)$, $\boldsymbol{j}_d(t)$, $\boldsymbol{s}_d(t)$.

Equation (1) shows a typical acceleration model of a multirotor, where the commanded acceleration is aligned with the body z-axis.

$$\boldsymbol{a} = u\boldsymbol{z} + \boldsymbol{g} + \boldsymbol{f}_e(\boldsymbol{\eta}, u) \tag{1}$$

Here $u \in \mathbb{R}$ is the commanded body acceleration, $\boldsymbol{z} \in \mathbb{R}^3$ is the body z-axis ($||\boldsymbol{z}|| = 1$), $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^\top$ is the gravity vector, and $\boldsymbol{f}_e \in \mathbb{R}^3$ is an additive acceleration error model that can, in general, be a function of both vehicle state $\boldsymbol{\eta}$ and control input u.

The objective is to compute the body acceleration u, body z-axis $\boldsymbol{z}$, angular velocity $\boldsymbol{\omega}$, and angular acceleration $\dot{\boldsymbol{\omega}}$ such that integrating $\dot{\boldsymbol{\omega}}$ forwards in time twice results in an orientation with $\boldsymbol{z}$ as the z-axis and that the vehicle acceleration, which is a function of $u\boldsymbol{z}$, equals the desired vehicle acceleration $\boldsymbol{a}_d(t)$. This will ensure that the vehicle follows the specified trajectory $\boldsymbol{x}_d(t)$. Note that while $\boldsymbol{z}$ and $\boldsymbol{\omega}$ are not true control inputs to the system, they are necessary as feedforward references to the attitude feedback controller. Once the body acceleration u and angular acceleration $\dot{\boldsymbol{\omega}}$ are computed, they are multiplied by mass and inertia and used as the feedforward force and torque in the position and attitude feedback controllers respectively.

For simplicity, we will assume that the yaw of the vehicle is always zero, but all of the methods presented are applicable while following yaw trajectories as well.

B. Input-independent error compensation

The simplest version of our control input generation strategy assumes that the disturbance model $\boldsymbol{f}_e$ is a function of the vehicle position and velocity only: $\boldsymbol{f}_e(\boldsymbol{x}, \dot{\boldsymbol{x}})$. In this case, the desired acceleration vector can be computed directly, as shown in (2).

$$u\boldsymbol{z} = \boldsymbol{a}_d - \boldsymbol{g} - \boldsymbol{f}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) \tag{2}$$

Since $\boldsymbol{z}$ must be of unit length, both u and $\boldsymbol{z}$ can be computed from $u\boldsymbol{z}$ by computing the magnitude and normalizing.

The angular velocity and angular acceleration are found by first computing the first and second time derivatives of $\boldsymbol{z}$.

Differentiating (2) in time results in

$$\dot{u}\boldsymbol{z} + u\dot{\boldsymbol{z}} = \boldsymbol{j}_d - \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} - \frac{\partial \boldsymbol{f}_e}{\partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}} = \boldsymbol{j}_d^{\text{eff}} \tag{3}$$

Since $\boldsymbol{z}$ is of unit length, and it must remain so, it must be perpendicular to $\dot{\boldsymbol{z}}$. Thus taking a dot product of (3) with $\boldsymbol{z}$ allows us to find $\dot{u}$.

$$\dot{u} = \boldsymbol{j}_d^{\text{eff}\top}\boldsymbol{z} \tag{4}$$

Inserting $\dot{u}$ into (3) gives us $\dot{\boldsymbol{z}}$.

$$\dot{\boldsymbol{z}} = \frac{1}{u}\left(\boldsymbol{j}_d^{\text{eff}} - \left(\boldsymbol{j}_d^{\text{eff}\top}\boldsymbol{z}\right)\boldsymbol{z}\right) \tag{5}$$

The body angular velocity can be extracted from $\dot{\boldsymbol{z}}$ by first defining the body x and body y-axes using a desired vehicle yaw, then projecting $\dot{\boldsymbol{z}}$ onto those axes. See [28] for the details.

To find $\ddot{\boldsymbol{z}}$, we differentiate (3).

$$\ddot{u}\boldsymbol{z} + 2\dot{u}\dot{\boldsymbol{z}} + u\ddot{\boldsymbol{z}} = \boldsymbol{s}_d - \ddot{\boldsymbol{f}}_e(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{s}_d^{\text{eff}} \tag{6}$$

The second time derivative of the learned disturbance $\boldsymbol{f}_e$ is shown in (7). Note that the second partial derivative of the error model with respect to its vector inputs is a 3rd order tensor.

$$\begin{aligned}\ddot{\boldsymbol{f}}_e = &\left(\frac{\partial^2 \boldsymbol{f}_e}{\partial \boldsymbol{x}^2}\dot{\boldsymbol{x}} + \frac{\partial^2 \boldsymbol{f}_e}{\partial \boldsymbol{x}\dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}}\right)\dot{\boldsymbol{x}} + \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{x}}\ddot{\boldsymbol{x}} + \\ &\left(\frac{\partial^2 \boldsymbol{f}_e}{\partial \dot{\boldsymbol{x}}\boldsymbol{x}}\dot{\boldsymbol{x}} + \frac{\partial^2 \boldsymbol{f}_e}{\partial \dot{\boldsymbol{x}}^2}\ddot{\boldsymbol{x}}\right)\ddot{\boldsymbol{x}} + \frac{\partial \boldsymbol{f}_e}{\partial \dot{\boldsymbol{x}}}\dddot{\boldsymbol{x}}\end{aligned} \tag{7}$$

Noting that differentiating $\boldsymbol{z}^\top\dot{\boldsymbol{z}} = 0$ implies $\boldsymbol{z}^\top\ddot{\boldsymbol{z}} = -\dot{\boldsymbol{z}}^\top\dot{\boldsymbol{z}}$ and again taking a dot product with $\boldsymbol{z}$, we can compute $\ddot{u}$.

$$\ddot{u} = \boldsymbol{s}_d^{\text{eff}\top}\boldsymbol{z} + u\dot{\boldsymbol{z}}^\top\dot{\boldsymbol{z}} \tag{8}$$

Inserting $\ddot{u}$ into (6) gives us $\ddot{\boldsymbol{z}}$.

$$\ddot{\boldsymbol{z}} = \frac{1}{u}\left(\boldsymbol{s}_d^{\text{eff}} - \ddot{u}\boldsymbol{z} - 2\dot{u}\dot{\boldsymbol{z}}\right) \tag{9}$$

To compute the body angular acceleration from $\ddot{\boldsymbol{z}}$, we note that $\ddot{\boldsymbol{z}} = \dot{\boldsymbol{\omega}} \times \boldsymbol{z} + \boldsymbol{\omega} \times \dot{\boldsymbol{z}}$ and proceed as before for the angular velocity, by projecting $\ddot{\boldsymbol{z}} - \boldsymbol{\omega} \times \dot{\boldsymbol{z}}$ onto the body x and y-axes.

Note that the above equations for $\dot{\boldsymbol{z}}$ and $\ddot{\boldsymbol{z}}$ are similar to those derived in [28] with the difference that here, the first and second derivatives of the learned dynamics model are incorporated. In this way, the control inputs generated *anticipate* changes in the disturbance.

One practical issue that arises is that the vehicle acceleration, $\ddot{\boldsymbol{x}}$, and jerk, $\dddot{\boldsymbol{x}}$, are not readily available during operation. Computing them from odometry by taking finite-differences will introduce noise. To alleviate this in our experiments, we use the acceleration and jerk demanded by the trajectory, which are good approximations of the true vehicle acceleration and jerk when tracking error is low.

C. Input-dependent error compensation

In many cases, additive dynamics model errors are a function of the applied control input and vehicle orientation, in addition to the vehicle position and velocity. For example, if the mass of the vehicle is not accurately known (or alternatively, the actuators are not properly modeled), the disturbance will be a linear function of the applied acceleration. The input-dependent acceleration model is shown in (10).

$$u\boldsymbol{z} = \boldsymbol{a}_d - \boldsymbol{g} - \boldsymbol{f}_e(\boldsymbol{\eta}, \boldsymbol{u}) \tag{10}$$

Here, $\boldsymbol{\eta} = \begin{bmatrix}\boldsymbol{x} & \dot{\boldsymbol{x}}\end{bmatrix}^\top$ contains the vehicle position and velocity and $\boldsymbol{u} = u\boldsymbol{z}$.

Without assuming a particular form for the additive error term $\boldsymbol{f}_e$, it is not possible to solve for the required acceleration and orientation analytically. We must resort to solving the problem numerically. Interestingly however, once a solution for the acceleration and orientation is found, the rest of the control inputs can be found analytically in a method similar to the input-independent case described above.

We first rewrite the acceleration model as the functional equation $\boldsymbol{f}(\boldsymbol{u}, t) = 0$ that is only a function of $\boldsymbol{u}$ and time. We compute the time derivative of $\boldsymbol{u}$ by taking a derivative of the above equation and solving the resulting linear system.

$$\dot{\boldsymbol{f}}(\boldsymbol{u}, t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\dot{\boldsymbol{u}} + \frac{\partial \boldsymbol{f}}{\partial t} = 0 \tag{11}$$

$$\dot{\boldsymbol{u}} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^{-1}\frac{\partial \boldsymbol{f}}{\partial t} \tag{12}$$

For our acceleration model, $\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{u} + \boldsymbol{g} + \boldsymbol{f}_e(\boldsymbol{\eta}, \boldsymbol{u}) - \boldsymbol{a}_d$. The necessary derivatives are shown in (13) and (14).

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} = \boldsymbol{I}_3 + \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{u}} \tag{13}$$

$$\frac{\partial \boldsymbol{f}}{\partial t} = \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{\eta}}\dot{\boldsymbol{\eta}} - \boldsymbol{j}_d \tag{14}$$

To find $\ddot{\boldsymbol{u}}$, we take a derivative of (11) and again solve the resulting linear system.

$$\ddot{\boldsymbol{f}}(\boldsymbol{u}, t) = \left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u}^2}\dot{\boldsymbol{u}} + 2\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u}\partial t}\right)\dot{\boldsymbol{u}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\ddot{\boldsymbol{u}} + \frac{\partial^2 \boldsymbol{f}}{\partial t^2} \tag{15}$$

$$\ddot{\boldsymbol{u}} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^{-1}\left(-\left(\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u}^2}\dot{\boldsymbol{u}} + 2\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u}\partial t}\right)\dot{\boldsymbol{u}} - \frac{\partial^2 \boldsymbol{f}}{\partial t^2}\right) \tag{16}$$

The necessary derivatives for our acceleration model are shown in (17), (18), and (19).

$$\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u}^2} = \frac{\partial^2 \boldsymbol{f}_e}{\partial \boldsymbol{u}^2} \tag{17}$$

$$\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{u} \partial t} = \frac{\partial^2 \boldsymbol{f}_e}{\partial \boldsymbol{u} \partial \boldsymbol{\eta}} \dot{\boldsymbol{\eta}} \tag{18}$$

$$\frac{\partial^2 \boldsymbol{f}}{\partial t^2} = \left(\frac{\partial^2 \boldsymbol{f}_e}{\partial \boldsymbol{\eta}^2} \dot{\boldsymbol{\eta}} \right) \dot{\boldsymbol{\eta}} + \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{\eta}} \ddot{\boldsymbol{\eta}} - \boldsymbol{s}_d \tag{19}$$

To compute $\dot{\boldsymbol{z}}$ from $\dot{\boldsymbol{u}}$, we take a derivative of $\boldsymbol{u} = u\boldsymbol{z}$ and proceed as before, by projecting onto $\boldsymbol{z}$ and solving first for $\dot{u}$.

$$\dot{\boldsymbol{u}} = \dot{u}\boldsymbol{z} + u\dot{\boldsymbol{z}} \tag{20}$$

$$\dot{u} = \dot{\boldsymbol{u}}^\top \boldsymbol{z} \tag{21}$$

$$\dot{\boldsymbol{z}} = \frac{1}{u} \left(\dot{\boldsymbol{u}} - \dot{u}\boldsymbol{z} \right) \tag{22}$$

To compute $\ddot{\boldsymbol{z}}$ from $\ddot{\boldsymbol{u}}$, and the angular velocity and angular acceleration, we follow the same approach as for the input-independent case.

It should be noted that this approach requires the existence of a solution to $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \dot{\boldsymbol{u}} = \frac{\partial \boldsymbol{f}}{\partial t}$ and the analogous equation for $\ddot{\boldsymbol{u}}$. Solutions will only fail to exist when the estimated disturbance model is strong enough to completely negate the acceleration imparted by $\boldsymbol{u}$. This may be a concern when learning a model from data, but in practice has not occurred in our experiments.

D. Model Learning

To estimate $\boldsymbol{f}_e$ from vehicle trajectory data, we fit a model to differences between the observed and the predicted acceleration at every time step. The observed acceleration is computed using finite-differences of the estimated vehicle velocity while the predicted acceleration is $u\boldsymbol{z} + \boldsymbol{g}$.

In principle, any regressive model whose derivatives are available can be used.

III. EXPERIMENTS

We first evaluate the proposed approach on a simulated 2D multirotor that is subjected to a series of input-independent and input-dependent disturbances. We then evaluate how the approach reduces tracking error on a quadrotor executing aggressive trajectories.

A. Simulation

The 2D planar multirotor captures many of the important dynamics present in the 3D multirotor. Namely, orientation and acceleration are coupled. In fact, the motion of a 3D multirotor moving in a vertical plane, e.g. in a straight line trajectory, can essentially be described with the 2D multirotor. As such, we believe a planar simulation is an appropriate testbed for our method.

The 2D multirotor force model is shown in Fig. 2. The dynamics are shown in (23) – (25), where F is the applied body force and τ is the applied body acceleration. The mass, m, was set to 4.19 kg, gravity g to 10.18 m/s^2, and inertia I to 0.123 kg-m^2.

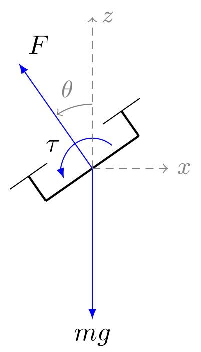


Fig. 2. Force diagram of the 2D multirotor used in the simulation experiment. F and τ are the control inputs.

$$m\ddot{x} = -F \sin(\theta) \tag{23}$$

$$m\ddot{z} = F \cos(\theta) - mg \tag{24}$$

$$I\ddot{\theta} = \tau \tag{25}$$

We subject the simulated multirotor to disturbances selected from Table I. Disturbance 1 is constant and emulates a fixed force field in the x direction, e.g. due to wind. It is not input-dependent and is not dynamic since it does not change along with the vehicle state. Disturbance 2 is velocity dependent and emulates drag in the x direction. Disturbance 3 depends on the vehicle angle and is thus input-dependent. Disturbance 4 is velocity dependent and emulates drag in the z direction. Disturbance 5 is a mass perturbation that adds a disturbance linear in the applied acceleration, which makes it input-dependent.

TABLE I
DISTURBANCES USED IN THE 2D MULTIROTOR SIMULATION EXPERIMENT.

No.	Effect	Input-dependent?	Dynamic?
1	$\ddot{x}$ -= 4.1	no	no
2	$\ddot{x}$ -= $3.1\dot{x}$	no	yes
3	$\ddot{x}$ += $1.4\sin(\theta)$	yes	yes
4	$\ddot{z}$ -= $3.1\dot{z}$	no	yes
5	m += 2	yes	yes

The vehicle is given a desired trajectory that takes it from $x = 0$, $z = 0$, to $x = 1$, $z = 1$ in one second. The trajectories in x and z are both 7th-order polynomials that have the velocity, acceleration, and jerk equal to zero at each of their endpoints. This ensures that the trajectory starts and ends with the vehicle at rest, at an angle of zero, and with an angular velocity of zero. When Disturbance 1 is in effect, the vehicle's angle is initialized such that maintaining zero acceleration in z also maintains zero acceleration in x. This ensures that the trajectory can be perfectly followed with correct control inputs despite the constant acceleration disturbance in x. In all other cases, the vehicle state starts at 0.

We show x and z tracking error for the following feedforward input generation strategies with and without feedback.

FF1) No disturbance learning
FF2) Basic disturbance compensation (no disturbance dynamics)
FF3) Disturbance compensation w/ numerical optimization
FF4) Dist. comp. w/ disturbance dynamics (ours)
FF5) Dist. comp. w/ num. opt. and disturbance dynamics (ours)

FF1 uses the feedforward generation strategy as presented in [28] and does not do any regression for disturbance learning. **FF2** and **FF3** do not consider the dynamics of the disturbance; they compute the angular velocity and angular acceleration feedforward terms as in [28] while incorporating the learned disturbance in the acceleration model, (1). **FF4** is the proposed approach that deals with input-independent disturbances while **FF5** is the proposed approach that deals with input-dependent disturbances.

In this experiment, **FF3** and **FF5** solve (10) numerically using the modified Powell method root finder in SciPy [18]. The initial guess for the optimization is the solution from the previous timestep.

Position and angle feedback is provided by PD controllers with gains of 10 on position and velocity errors, 300 on angle errors, and 30 on angular velocity errors. The position PD controller output is added to the desired acceleration and the angle PD controller output is added to the desired angular acceleration.

In all simulation experiments, the feature vector used for linear regression of model errors is shown in (26). The features were hand selected to appropriately model the disturbances in Table I.

$$\boldsymbol{\phi}(x, z, \theta, \dot{x}, \dot{z}, F) = \left[x, z, \dot{x}, \dot{z}, \sin(\theta), F\sin(\theta), F\cos(\theta), 1\right]^\top \tag{26}$$

The learned model is thus

$$\boldsymbol{f}_e(\boldsymbol{\eta}, \boldsymbol{u}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{\eta}, \boldsymbol{u}) \tag{27}$$

$\boldsymbol{w}$ is the result of regressing the projected input data $\boldsymbol{\phi}$ to the observed acceleration errors and minimizing least squared error. In this experiment, $\boldsymbol{w}$ is recomputed after every trajectory execution using data from all past executions. Results reported are on the 3rd run, since we found that only two regression steps were needed to converge to an accurate enough model. This is not surprising, as in this simulation there is no noise and the features used can appropriately reproduce the applied disturbances.

Each control configuration is subjected to the following set of disturbance combinations.

A) Disturbance 1
B) Disturbances 1, 2, and 4
C) Disturbances 3 and 5
D) Disturbances 1, 2, 3, 4, and 5

Error plots for each of the four disturbance sets without feedback control are shown in Figs. 3a – 3d. Under only a constant disturbance (Fig. 3a), all disturbance compensation strategies work well, since the disturbance is neither input-dependent nor dynamic. When we introduce drag, a dynamic disturbance, in disturbance set **B** (Fig. 3b), only the approaches that compensate for disturbance dynamics, **FF4** and **FF5**, achieve low error. Although basic disturbance compensation as in **FF2** helps considerably, accounting for disturbance dynamics improves performance further. Since in disturbance set **B**, the disturbances are still input-independent, the use of numerical optimization to solve the acceleration model (1) has no effect.

TABLE II
MAXIMUM ABSOLUTE POSITION ERRORS, IN METERS, FOR EACH CONTROL STRATEGY WITHOUT FEEDBACK IN SIMULATION

Dist. Set \ Strategy	FF1	FF2	FF3	FF4	FF5
A	0.829	**0.008**	**0.007**	**0.008**	**0.008**
B	1.275	0.140	0.137	**0.007**	**0.003**
C	1.998	0.417	0.684	1.153	**0.025**
D	2.131	0.866	0.159	1.023	**0.002**

TABLE III
MAXIMUM ABSOLUTE POSITION ERRORS, IN METERS, FOR EACH CONTROL STRATEGY WITH FEEDBACK IN SIMULATION

Dist. Set \ Strategy	FF1	FF2	FF3	FF4	FF5
A	0.256	**0.001**	**0.001**	**0.001**	**0.001**
B	0.412	0.041	0.042	**0.001**	**0.000**
C	0.293	0.069	0.064	0.076	**0.001**
D	0.722	0.189	0.030	0.194	**0.000**

Under input-dependent disturbances, we see that **FF5** is the only approach that achieves low error. This is expected, as for both disturbance sets **C** and **D**, there are dynamic and input-dependent disturbances present.

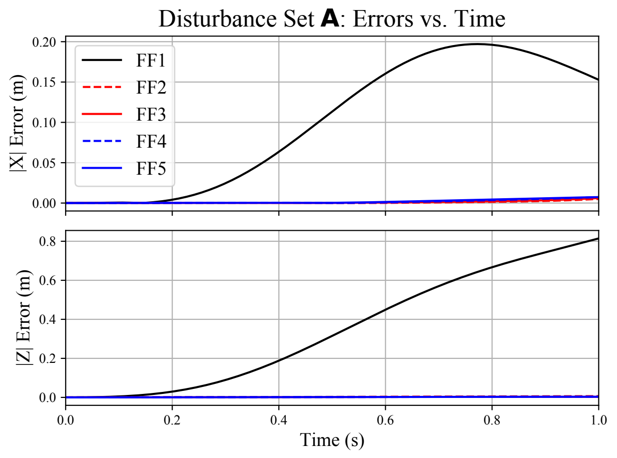
Error plots for disturbance set **D** *with* feedback control are shown in Fig. 4. We see that although feedback can reduce the error, it is not enough to completely eliminate the error. **FF5** still outperforms the other methods, achieving nearly zero error in all trials.

The maximum absolute position errors over the trajectory for all tested configurations are listed in Tables II and III.

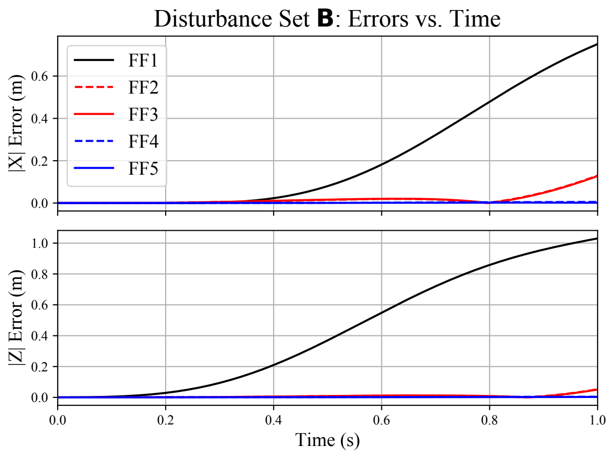
B. Hardware

1) Platform & Setup: To validate the usefulness of dynamic disturbance compensation and input-dependent disturbance compensation, we compare the five aforementioned feedforward generation strategies, **FF1** through **FF5**, on a 750 g quadrotor while following aggressive trajectories. Figure 5 shows the hardware platform and Fig. 1 shows the robot while following aggressive circle and line trajectories.

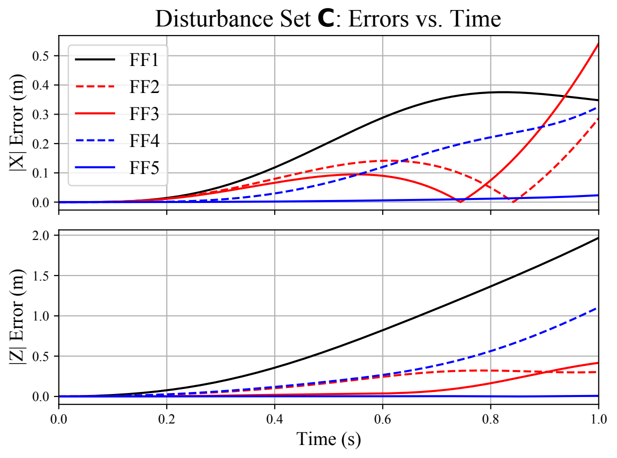
Position, velocity, and yaw feedback is provided by a motion capture arena at 100 Hz, while pitch, roll, and angular velocity feedback is provided by a Pixhawk PX4 at 250 Hz. Feedback control is performed by a cascaded PD system following [28]. The feedforward terms are as computed by **FF1** through **FF5**. **FF3** and **FF5** solve (10) numerically using the Newton-Raphson method. All control computation is



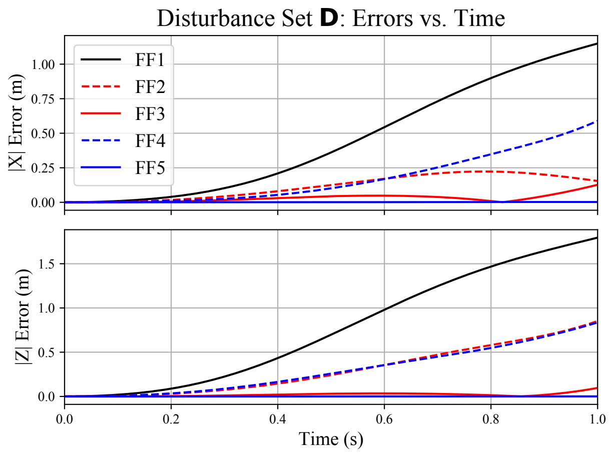
(a) Errors for disturbance set **A**, containing only a constant disturbance. All strategies that compensate for the disturbance perform well.



(b) Errors for disturbance set **B**, containing dynamic, but input-independent disturbances. **FF4** and **FF5**, which compensate for dynamic disturbances, perform the best.



(c) Errors for disturbance set **C**, containing dynamic and input-dependent disturbances. Only **FF5** performs well.



(d) Errors for disturbance set **D**, which contains all considered disturbances. **FF5** performs the best.

Fig. 3. Error plots for all five feedforward strategies without feedback control under each of the four disturbance sets.

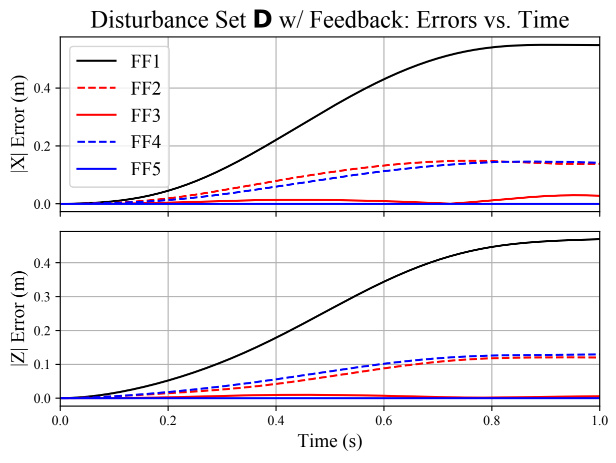


Fig. 4. Error plots for all five feedforward strategies *with* feedback control under disturbance set **D**. **FF5** outperforms all others.

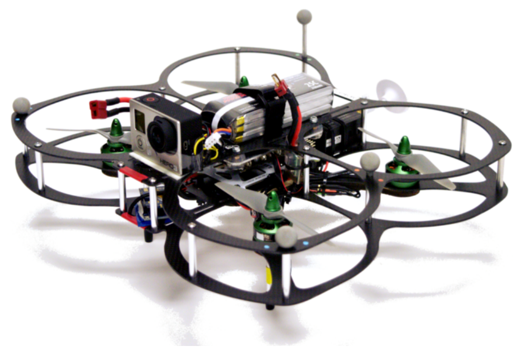


Fig. 5. The 750 g quadrotor used for the hardware experiments. Onboard computation is performed by an Odroid XU4 and the Pixhawk 1 Flight Controller.

performed onboard the vehicle's Odroid XU4 computer. The position control loop runs at 100 Hz and the attitude control loop runs at 200 Hz.

For the hardware experiments, we use three test trajectories: a 1.8 s straight line trajectory, a circle trajectory, and a figure 8 trajectory. The trajectories are designed to be near the limit of what the robot can feasibly track. Table IV lists the three trajectories and their maximum absolute derivatives.

TABLE IV
THE AGGRESSIVE TRAJECTORIES USED TO EVALUATE THE PROPOSED APPROACH AND THEIR MAXIMUM DERIVATIVES.

Traj.	x **(m)**	$\dot{x}$ **(m/s)**	$\ddot{x}$ **(m/s^2)**	$x^{(3)}$ **(m/s^3)**	$x^{(4)}$ **(m/s^4)**
Line	2.7	3.28	6.26	24.31	216
Circle	2.0	2.75	7.56	21.43	64.35
Figure 8	2.0	2.75	7.15	21.43	59.25

2) Model Learning: We use linear regression as the model learning strategy in the hardware experiment. Input data to the regression is a 6 dimensional vector consisting of the vehicle velocity and the commanded acceleration vector $\boldsymbol{u}$. The system starts with an uninitialized model and uses a few test trajectories per trial to regress to the acceleration error. The error model is then held fixed during the remaining trajectories used for error evaluation. Although in principle, the model can be updated incrementally, keeping it fixed allows for a fair comparison between the control strategies.

3) Results: For the line trajectory, each of **FF1**, **FF2**, **FF4**, and **FF5** is evaluated four times. The first four trajectories, run using **FF1**, are used to train the acceleration error model. An overlay of the vehicle executing the 2.7 m line trajectory can be seen in Fig. 1. Absolute errors along the trajectory and errors along the vertical axis for the line trajectory are shown in Fig. 6. **FF1** performs the worst, especially along the vertical axis, indicating that the robot is underestimating the control input required to maintain hover. **FF2** eliminates much of the error in the vertical axis, but still accumulates significant error along the trajectory, rising above 10 cm consistently. **FF4** and **FF5** provide on average a 30% reduction in the average absolute $x - y$ tracking error along the trajectory when compared to **FF2**. This indicates that taking disturbance dynamics into account can significantly improve tracking performance. This trajectory does not provide sufficient clarity to determine the impact of **FF5**, input-dependent disturbance compensation.

For the circle trajectory, all of the feedforward strategies are evaluated once, with **FF3** and **FF5** receiving two and four more trajectories respectively. An overlay of the vehicle executing the circle trajectory can be seen in Fig. 1. Fig. 7 shows the resulting error. As expected **FF1**, with no disturbance compensation, performs the worst. **FF2**, **FF4**, and **FF5** all perform similarly well, with **FF2** achieving slightly lower vertical error than the others. **FF3** performs slightly worse here than **FF2**, suggesting that the numerical routine may be failing to converge or that the input's dependence on the acceleration error has not been properly modeled.

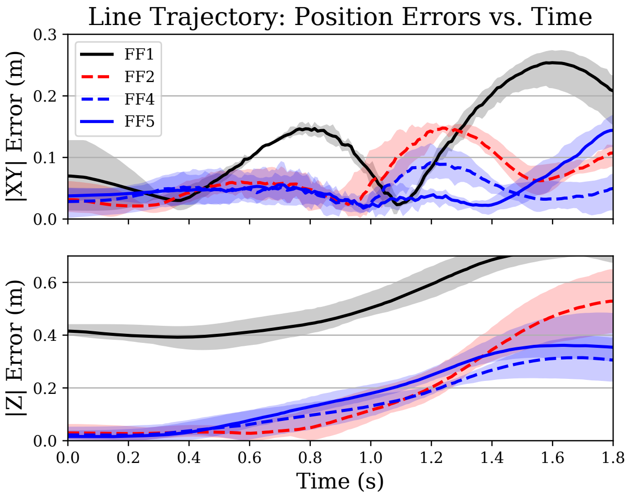


Fig. 6. Average absolute errors during an aggressive straight line trajectory for four of the five control strategies. Shaded regions denote the minimum and maximum errors per timestep over four trials. Means (m) (± std (m)) over the 4 trajectories of the average $|x - y|$ error for **FF1**, **FF2**, **FF4**, and **FF5** respectively are 0.120 ± 0.003, 0.067 ± 0.003, **0.047** ± 0.009, and **0.048** ± 0.10. Those for the average $|z|$ error are 0.525 ± 0.023, 0.173 ± 0.034, **0.142** ± 0.025, and 0.173 ± 0.030.

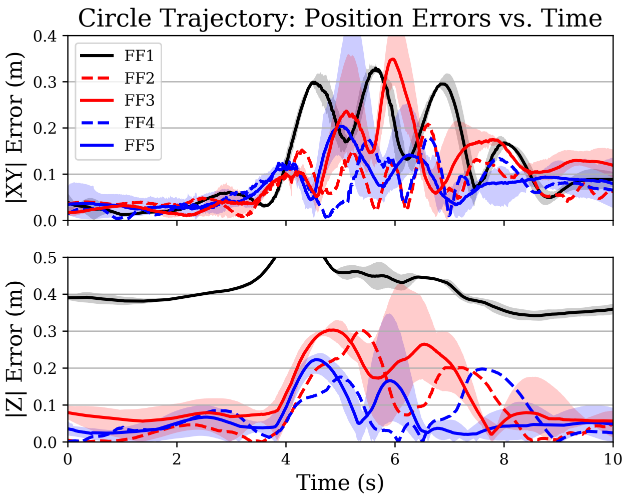


Fig. 7. Average absolute errors during an aggressive circle trajectory for the five control strategies. Shaded regions denote the minimum and maximum errors per timestep. Avg. $|x - y|$ errors (m) for **FF1**, **FF2**, **FF3**, **FF4**, and **FF5** are 0.118, **0.071**, 0.103, **0.069**, and **0.076**, respectively, while avg. $|z|$ errors (m) are 0.41, 0.084, 0.122, **0.069**, and **0.066**, respectively.

Fig. 8 shows the error of **FF1**, **FF2**, **FF4**, and **FF5** along the figure 8 trajectory for one trial each. The improvement of **FF4** over **FF2** here is smaller than in the other trajectories, suggesting that dynamic disturbances have relatively less of an impact when following the figure 8, though more experimental trials are warranted to strengthen this claim.

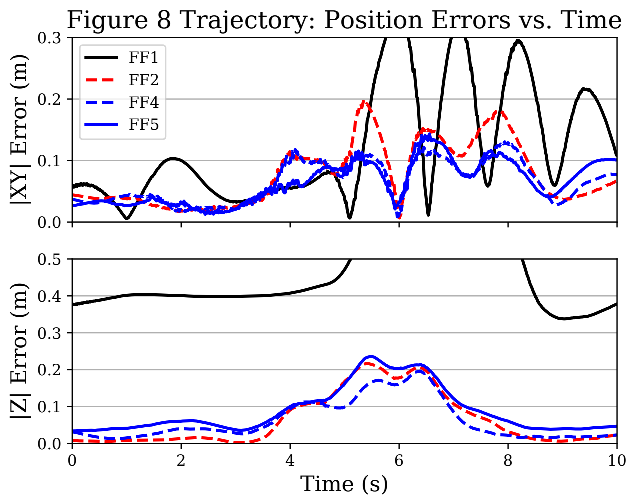


Fig. 8. Errors during an aggressive figure 8 trajectory for four of the five control strategies. Avg. $|x - y|$ errors (m) for **FF1**, **FF2**, **FF4**, and **FF5** are 0.105, 0.076, **0.063**, and **0.059** respectively, while avg. $|z|$ errors (m) are 0.473, **0.064**, **0.064**, and 0.088, respectively.

IV. Conclusion

We have presented a method that allows compensation of dynamic disturbances through evaluation of the derivatives of a learned model. We have shown in both simulation and hardware experiments that our dynamic disturbance compensation method improves performance over traditional disturbance compensation. We have also shown the usefulness of input-dependent disturbance compensation in simulation and preliminary results on hardware. The versatility of the approach in a realistic robotics application has been verified through evaluation on three distinct test trajectories.

Future work will evaluate nonlinear regression techniques, such as ISSGPR, on hardware platforms, as well as consider regression techniques that explicitly optimize model derivative accuracy. An interesting avenue of future study is to analyze theoretically how the error model accuracy affects the performance of each of the feedforward generation strategies. Lastly, we hope to apply this technique to the attitude dynamics of multirotors, in order to fully compensate for vehicle disturbances and modeling errors.

V. Acknowledgments

The authors thank Xuning Yang for helpful feedback on this manuscript.

References

[1] P. Abbeel, V. Ganapathi, and A. Ng. Learning vehicular dynamics, with application to modeling helicopters. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, Cambridge, MA, USA, 2005. MIT Press. URL https://dl.acm.org/citation.cfm?id=2976248.2976249.

[2] A. Aswani, H. Gonzalez, S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013. URL https://www.sciencedirect.com/science/article/pii/S0005109813000678.

[3] S. V. Balakrishnan. Fast incremental adaptation using maximum likelihood regression and stochastic gradient descent. In *INTERSPEECH*, 2003. URL https://pdfs.semanticscholar.org/ffc2/422d108b11cb8f0e947cf0f8b92c6f7607b5.pdf.

[4] M. Bangura and R. Mahony. Nonlinear dynamic modeling for high performance control of a quadrotor. In *Australasian conference on robotics and automation*, pages 1–10, 2012. URL https://www.araa.asn.au/acra/acra2012/papers/pap121.pdf.

[5] M. Burri, M. Bloesch, D. Schindler, I. Gilitschenski, Z. Taylor, and R. Siegwart. Generalized information filtering for MAV parameter estimation. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3124–3130. IEEE, 2016. URL https://ieeexplore.ieee.org/document/7759483/.

[6] M. Burri, J. Nikolic, H. Oleynikova, M. W. Achtelik, and R. Siegwart. Maximum likelihood parameter identification for MAVs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4297–4303. IEEE, 2016. URL https://ieeexplore.ieee.org/document/7487627.

[7] E. Cappo, A. Desai, M. Collins, and N. Michael. Online planning for human–multi-robot interactive theatrical performance. *Autonomous Robots*, 42(8):1771–1786, December 2018. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-018-9755-0. URL https://link.springer.com/10.1007/s10514-018-9755-0.

[8] V. Desaraju. *Safe, efficient, and robust predictive control of constrained nonlinear systems*. Ph.D. Thesis, Carnegie Mellon University., 4 2017. doi: 10.1184/R1/6721379.v1. URL https://kilthub.cmu.edu/articles/Safe_Efficient_and_Robust_Predictive_Control_of_Constrained_Nonlinear_Systems/6721379.

[9] A. Droniou, S. Ivaldi, V. Padois, and O. Sigaud. Autonomous online learning of velocity kinematics on the icub: A comparative study. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3577–3582. IEEE, 2012. URL https://ieeexplore.ieee.org/document/6385674.

[10] M. Faessler, D. Falanga, and D. Scaramuzza. Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 2(2):476–482, 2017. doi: 10.1109/LRA.2016.2640362. URL https://ieeexplore.ieee.org/document/7784809.

[11] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3:620–626, 2018. doi: 10.1109/LRA.2017.2776353. URL https://ieeexplore.

ieee.org/document/8118153.

[12] J. Ferrin, R. Leishman, R. Beard, and T. McLain. Differential flatness based control of a rotorcraft for aggressive maneuvers. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2688–2693, September 2011. doi: 10.1109/IROS.2011.6095098. URL https://ieeexplore.ieee.org/document/6095098.

[13] J. Florez, D. Bellot, and G. Morel. LWPR-model based predictive force control for serial comanipulation in beating heart surgery. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 320–326, July 2011. doi: 10.1109/AIM.2011.6027055. URL https://ieeexplore.ieee.org/document/6027055.

[14] A. Gijsberts and G. Metta. Incremental learning of robot dynamics using random features. In *2011 IEEE International Conference on Robotics and Automation*, pages 951–956. IEEE, May 2011. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5980191. URL https://ieeexplore.ieee.org/document/5980191/.

[15] A. Gijsberts and G. Metta. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks*, 41:59 – 69, 2013. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2012.08.011. URL https://www.sciencedirect.com/science/article/pii/S0893608012002249.

[16] D. Grollman and O. C. Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *2008 IEEE International Conference on Robotics and Automation*, pages 3315–3320, Pasadena, CA, USA, May 2008. IEEE. ISBN 978-1-4244-1646-2. doi: 10.1109/ROBOT.2008.4543716. URL https://ieeexplore.ieee.org/document/4543716/.

[17] M. Hehn and R. D'Andrea. Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*, 31(4):877–892, August 2015. ISSN 1552-3098. doi: 10.1109/TRO.2015.2432611. URL https://ieeexplore.ieee.org/document/7128399.

[18] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL https://www.scipy.org/.

[19] S. Kim, S. Choi, and H. J. Kim. Aerial manipulation using a quadrotor with a two DOF robotic arm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4990–4995, November 2013. doi: 10.1109/IROS.2013.6697077. URL https://ieeexplore.ieee.org/document/6697077.

[20] D. Lee, H. Jin Kim, and S. Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of Control, Automation and Systems*, 7(3):419–428, June 2009. ISSN 1598-6446. doi: 10.1007/s12555-009-0311-8. URL https://link.springer.com/10.1007/s12555-009-0311-8.

[21] T. Lee, M. Leoky, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, December 2010. doi: 10.1109/CDC.2010.5717652. URL https://ieeexplore.ieee.org/document/5717652.

[22] Q. Li, J. Qian, Z. Zhu, X. Bao, M. Helwa, and A. Schoellig. Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5189, May 2017. doi: 10.1109/ICRA.2017.7989607. URL https://ieeexplore.ieee.org/document/7989607.

[23] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004. URL https://homes.cs.washington.edu/~todorov/papers/LiICINCO04.pdf.

[24] S. Lupashin, A. Schoellig, 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*, pages 1642–1648. IEEE, 2010. URL https://ieeexplore.ieee.org/document/5509452/.

[25] Z. Manchester and S. Kuindersma. DIRTREL: Robust trajectory optimization with ellipsoidal disturbances and lqr feedback. In *Robotics: Science and Systems (RSS)*, 2017. URL https://doi.org/10.15607/RSS.2017.XIII.057.

[26] C. McKinnon and A. Schoellig. Learning multimodal models for robot dynamics online with a mixture of gaussian process experts. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 322–328, 2017. URL https://ieeexplore.ieee.org/document/7989041.

[27] D. Mellinger. *Trajectory generation and control for quadrotors*. Publicly Accessible Penn Dissertations. 547., 2012. URL https://repository.upenn.edu/edissertations/547.

[28] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011. URL https://ieeexplore.ieee.org/document/5980409/.

[29] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, September 2012. ISSN 15564959. doi: 10.1002/rob.21436. URL https://doi.wiley.com/10.1002/rob.21436.

[30] B. Morrell, M. Rigter, G. Merewether, R. Reid, R. Thakker, T. Tzanetos, V. Rajur, and G. Chamitoff. Differential flatness transformations for aggressive quadrotor flight. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. doi: 10.1109/ICRA.2018.8460838. URL https://ieeexplore.ieee.org/document/8460838.

[31] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340,

November 2011. ISSN 1612-4782, 1612-4790. doi: 10.1007/s10339-011-0404-1. URL https://link.springer.com/10.1007/s10339-011-0404-1.

[32] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008. URL https://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

[33] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research: The 16th International Symposium ISRR*, volume 114, pages 649–666. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28872-7. doi: 10.1007/978-3-319-28872-7_37. URL https://doi.org/10.1007/978-3-319-28872-7_37.

[34] G. Rivera and O. Sawodny. Flatness-based tracking control and nonlinear observer for a micro aerial quadcopter. *AIP Conference Proceedings*, 1281(1):386, September 2010. ISSN 0094-243X. doi: 10.1063/1.3498483. URL https://aip.scitation.org/doi/10.1063/1.3498483.

[35] S. Schaal, C. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, Jul 2002. ISSN 1573-7497. doi: 10.1023/A:1015727715131. URL https://doi.org/10.1023/A:1015727715131.

[36] J. Svacha, K. Mohta, and V. Kumar. Improving quadrotor trajectory tracking by compensating for aerodynamic effects. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 860–866, June 2017. doi: 10.1109/ICUAS.2017.7991501. URL https://ieeexplore.ieee.org/document/7991501.

[37] E. Tal and S. Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4282–4288, 12 2018. doi: 10.1109/CDC.2018.8619621. URL https://ieeexplore.ieee.org/document/8619621.

[38] M. Van Nieuwstadt and R. Murray. Real-time trajectory generation for differentially flat systems. *Int. J. Robust Nonlinear Control*, 8(11): 995–1020, September 1998. ISSN 1099-1239. doi: 10.1002/(SICI)1099-1239(199809)8:11⟨995::AID-RNC373⟩3.0.CO;2-W. URL https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1099-1239(199809)8:11⟨995::AID-RNC373⟩3.0.CO;2-W/abstract.

[39] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005. doi: 10.1162/089976605774320557. URL https://doi.org/10.1162/089976605774320557.