Fly Out The Window: Exploiting Discrete-Time Flatness for Fast Vision-Based Multirotor Flight

Melissa Greeff[1], SiQi Zhou[1], and Angela P. Schoellig[1]

***Abstract*—Recent work has demonstrated fast, agile flight using only vision as a position sensor and no GPS. Current feedback controllers for fast vision-based flight typically rely on a full-state estimate, including position, velocity and acceleration. An accurate full-state estimate is often challenging to obtain due to noisy IMU measurements, infrequent position updates from the vision system, and an imperfect motion model used to obtain high-rate state estimates required by the controller. In this work, we present an alternative control design that bypasses the need for a full-state estimate by exploiting discrete-time flatness, a structural property of the underlying vehicle dynamics. First, we show that the Euler discretization of the multirotor dynamics is discrete-time flat. This allows us to design a predictive controller using only a window of inputs and outputs, the latter consisting of position and yaw estimates. We highlight in simulation that our approach outperforms controllers that rely on an incorrect full-state estimate. We perform extensive outdoor multirotor flight experiments and demonstrate reliable vision-based navigation. In these experiments, our discrete-time flatness-based controller achieves speeds up to 10 m/s and significantly outperforms similar controllers that hinge on full-state estimation, achieving up to 80% path error reduction.**

***Index Terms*—Aerial Systems: Mechanics and Control; Aerial Systems: Perception and Autonomy; Vision-Based Navigation**

I. Introduction

MULTIROTOR unmanned aerial vehicles (UAVs), see Fig. 1, are mechanically simple and highly maneuverable, which makes them suitable to a wide range of applications such as infrastructure inspection [1], transportation, search-and-rescue missions, and mapping operations. This has challenged researchers to develop multirotor systems that can move beyond lab demonstrations to real-world scenarios where GPS may not always be reliable due to poor satellite coverage, multipath propagation or jamming. As such, in recent years significant advancements have been made in enabling high-performance flight using vision-based sensing as a lightweight and versatile alternative [2]-[4].

Control design within the visual navigation pipeline tends to rely on high-rate state estimates (generally 50-200 Hz for position control). In contrast, the visual system often estimates lower-rate (currently 10-35 Hz) and noisy position (and orientation) information compared to GPS or Vicon. The mismatch between the control requirements and visual measurement is addressed with an additional state estimator that uses a prediction model and/or integrates IMU measurements, to determine high-rate state estimates. Traditionally, the controllers and planners are designed independently of the state estimator [2]-[5]. Control design tends to assume perfect state estimation. Historically, this has been motivated by the separation principle, which theoretically guarantees that optimality is retained for certain linear stochastic systems when control and state estimation are decoupled. However, the required assumptions for these guarantees, that is, a linear model and zero-mean Gaussian output noise, are practically never satisfied by vision-based multirotor systems in real-world operations.

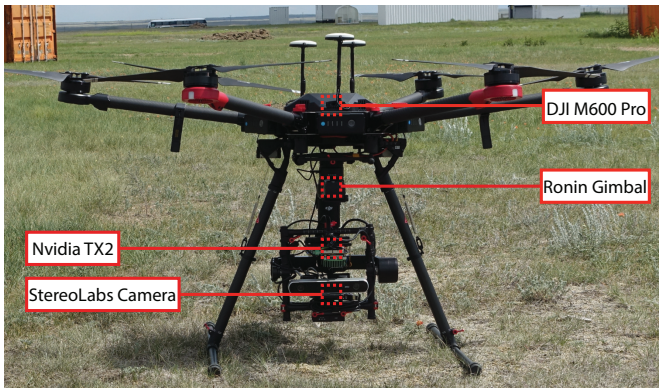

Fig. 1. Multirotor system used for autonomous vision-based flight experiments. We use a DJI M600 Pro with Ronin-MX gimbal and a StereoLabs camera. All computation is performed onboard on the Nvidia TX2.

Moreover, obtaining an accurate full-state estimate using visual navigation for multirotor control is challenging due to typically noisy IMU measurements, an infrequent position update from the vision system due to the required computational time and an imperfect motion model used to obtain high-rate state estimates required by control.

Related work tends to address the challenge of high-performance control under imperfect state estimation by (i) improving the state estimate by incorporating additional sensors, for example, IMU, laser range-finders, event cameras, e.t.c. [3], [6]; (ii) designing robust control to account for the worst-case state estimation error learnt offline from data [9], [8]; or (iii) performing output feedback control by coupling control and state estimation [10]-[12].

In this work, we present an alternative approach that by-

Manuscript received: September, 9, 2021; Revised December, 10, 2021; Accepted January, 31, 2022.

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by (organizations/grants which supported the work.)

[1]The authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS) and the Vector Institute for Artificial Intelligence, Toronto. This work was supported by Drone Delivery Canada, Defence Research and Development Canada, and the Natural Sciences and Engineering Research Council of Canada. Email: melissa.greeff@mail.utoronto.ca, siqi.zhou@robotics.utias.utoronto.ca, schoellig@utias.utoronto.ca

Digital Object Identifier (DOI): see top of this page.

passes the full state estimation requirement by exploiting a property known as discrete-time flatness. This property allows us to design a controller for vision-based flight that uses a window of inputs and special flat outputs, specifically position and yaw, and avoids inaccurate velocity and acceleration estimates.

II. Related Work and Contributions

1) Improve state estimation: Autonomous vision-based flight typically relies on visual or visual-inertial odometry (VIO), when an IMU is added, within a simultaneous localization and mapping (SLAM) [3] or visual teach and repeat (VT&R) framework [2], [4]. For example, vision-based drone racing relies on SLAM [3], [6]. In drone racing the multirotor must fly through static gates. The drift in VIO is compensated for by using a deep neural network (DNN) to learn a robust gate model. The identified gate locations are accounted for in an extended Kalman filter (EKF) to improve the multirotor state estimate. The state is then used in time-optimal trajectory planners [5] and simple lower-level controllers, for example, PD control or model predictive control (MPC). This estimation approach is task-specific and relies on static gates. Another approach, improves state estimates by fusing Ultra-wideband (UWB) range measurements with VIO in an EKF [7].

2) Design robust control for worst-case state estimation: In [8], a bound on the state estimation error is learned offline using data. A robust controller is synthesized that ensures a bounded system response despite estimation errors. This has been applied to vision-based flight for multirotors in [9]. This results in conservative performance and can sometimes lead to an infeasible problem. Moreover, the current theory applies to linear models and, therefore, has only been applied to flight near hover.

3) Couple control and state estimation: While commonly MPC considers state feedback, output feedback MPC approaches exist and rely on only measurements (or outputs). Robust output feedback MPC explicitly considers bounded measurement noise and incorporates a state estimator within the optimization problem in MPC [10]. Robust output feedback MPC can be a tube-based method (which guarantees constraint satisfaction despite measurement noise) [10]-[12] or a minmax method (which optimizes the worst-case performance) [13]. Current theory applies to linear models only [10]-[12]. Moreover, the main practical limitation is that these methods are slow to compute. Even for low-dimensional linear systems, the controller can often only be implemented at around 5 Hz [10]. For vision-based navigation, the measurement noise may vary and can be difficult to evaluate prior to flight. Furthermore, overestimating the measurement noise can lead to conservative performance.

In this work, we present a predictive controller that does not rely on the full-state estimate for vision-based multirotor flight. The contributions of this work are three-fold:

- To the best of our knowledge, this is the first work to demonstrate that the property known as discrete-time flatness holds for the Euler discretization of multirotors. This means that only a window of input (thrust and torques) and output (position and yaw) samples is required for control design.
- We highlight in simulation how the approach outperforms controllers that rely on a poor full-state estimate as a result of noisy position measurements (and higher-order derivative estimation) or large initial state uncertainty.
- In outdoor experiments, we show the application of our proposed discrete-time flatness-based controller to vision-based flight at speeds up to 10 m/s and how it outperforms controllers that hinge on accurate full-state estimation.

III. Background

A. Differential Flatness

We recall the formal definition of differential flatness [14].

Definition 1 (Differential Flatness): A nonlinear system model,

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{1}$$

with $t \in \mathbb{R}^+$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ is differentially flat if there exists an output $\mathbf{y}(t) \in \mathbb{R}^m$, whose components are differentially independent (i.e., the components are not related to each other through a differential equation), such that the following holds:

$$\begin{aligned} \mathbf{y} &= \tilde{H}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, ..., \mathbf{u}^{\delta}), \\ \mathbf{x} &= \tilde{\Lambda}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{r-1}), \\ \mathbf{u} &= \tilde{F}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{r}), \end{aligned} \tag{2}$$

where $\tilde{H}, \tilde{\Lambda}, \tilde{F}$ are smooth functions, δ and r are the maximum orders of the derivatives of $\mathbf{u}$ and $\mathbf{y}$ needed to describe the system and $\mathbf{y} = [y_1, y_2, ...y_m]^T$ is called the flat output. □

More intuitively, the system (1) is flat if there exists a one-to-one correspondence (or mapping) between its solutions $(\mathbf{x}(t), \mathbf{u}(t))$ and solutions $\mathbf{y}(t)$ of a trivial systems (with the same number of inputs, i.e., $\mathbf{y}^r = \tilde{\mathbf{v}}$ where $\tilde{\mathbf{v}} \in \mathbb{R}^m$ is a new fictitious input). This means that both the state $\mathbf{x}(t)$ and the input $\mathbf{u}(t)$ at time t can be determined from the (flat) output $\mathbf{y}(t)$ and a finite number of its derivatives.

The property of differential flatness (with position and yaw as output) is well-established for multirotors [15] and has been exploited for efficient trajectory generation in [15] and model predictive control in [16].

B. Discrete-Time Flatness

The discrete-time counterpart of differential flatness, commonly referred to as forward-difference flatness (or difference flatness), has recently been introduced in [17], [18]. The main difference is that time derivatives in Definition 1 are replaced by forward-shifts of the output [19].

Definition 2 (Discrete-Time Flatness): A nonlinear system model,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{3}$$

with $k \geq 0$, $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{u}_k \in \mathbb{R}^m$ is difference flat if there exists an output $\mathbf{y}_k \in \mathbb{R}^m$ whose components are independent

(i.e., the components are not related to each other through a difference equation), such that the following holds:

$$\mathbf{y}_k = H(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_{k+1}, ..., \mathbf{u}_{k+\delta}),$$

$$\mathbf{x}_k = \Lambda\left(\mathbf{y}_k, \mathbf{y}_{k+1}, ..., \mathbf{y}_{k+r-1}\right), \tag{4}$$

$$\mathbf{u}_k = F\left(\mathbf{y}_k, \mathbf{y}_{k+1}, ..., \mathbf{y}_{k+r}\right), \tag{5}$$

where H, Λ, F are smooth functions, δ and r are the maximum number of forward shifts of $\mathbf{u}$ and $\mathbf{y}$ needed to describe the system and $\mathbf{y}_k = [y_{1,k}, y_{2,k}, ...y_{m,k}]^T$ is called the flat output. □

More intuitively, the system (3) is flat if there exists a one-to-one correspondence (or mapping) between its solutions $(\mathbf{x}_k, \mathbf{u}_k)$ and solutions $\mathbf{y}_k$ of a trivial system (i.e., they do not satisfy a difference equation, but rather $\mathbf{y}_{k+r} = \mathbf{v}_k$ where $\mathbf{v}_k \in \mathbb{R}^m$ is a new fictitious input). This means that both the state $\mathbf{x}_k$ and input $\mathbf{u}_k$ at time step k can be determined from a finite number of future values of the output $\mathbf{y}_k$ [19]. The key concept of discrete-time flatness is that *the trajectory of the (flat) output in a certain finite window uniquely determines the state and input at any time step* [20].

Unfortunately, differential flatness of (1) does not necessarily imply difference flatness of a discretization (either exactly if possible or using Euler discretization) of (1). A counterexample is provided in [19]. However, in this paper, we show that difference flatness still applies to an Euler discretization of the multirotor dynamics and how we can exploit this property in *control design using only input and output information* (i.e., bypassing the more traditional requirement for full-state information).

IV. Methodology

We show that discrete-time flatness, as described in Definition 2, holds for the Euler discretization of the full multirotor dynamics model (from [14]). We also, briefly, show that this property holds for a simpler 2-D motion model, with underlying first-order pitch and roll dynamics (as a result of lower-level attitude controllers), which is used in our simulation and experimental results in Sec. V and VI. In Sec. IV-C, we illustrate how discrete-time flatness can be used in a predictive controller that uses only input and (flat) output information.

A. Discrete Flatness of Euler-Discretized Multirotor Dynamics Model

1) Dynamics Model: We consider the following discrete-time model (using the Euler discretization approximation of the model in [14]):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta t \dot{\mathbf{x}}_k,$$

where δt is the time step of the discretization and the state at time step k, $\mathbf{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k, z_k, \dot{z}_k, \theta_k, \phi_k, \psi_k, p_k, q_k, r_k]^T$ comprises of the 3-D position x_k, y_k, z_k, the 3-D velocity $\dot{x}_k, \dot{y}_k, \dot{z}_k$, the vehicle roll ϕ_k, pitch θ_k and yaw ψ_k, and the angular velocities (in the vehicle frame) p_k, q_k, r_k at time step k. The translational dynamics are:

$$\begin{bmatrix} \ddot{x}_k \\ \ddot{y}_k \\ \ddot{z}_k + g \end{bmatrix} = \frac{T_k}{m} \begin{bmatrix} R_k^{13} \\ R_k^{23} \\ R_k^{33} \end{bmatrix}, \tag{6}$$

where $\ddot{x}_k, \ddot{y}_k, \ddot{z}_k$ are the 3-D accelerations at time step k, T_k is the commanded thrust at time step k, g is the gravitational constant, m is the vehicle mass, and R is the rotation matrix from the body to inertial frames.

The notation $R_k^{\alpha\beta}$ denotes the α row and β column entry of rotation matrix R at time step k. The rotational dynamics are:

$$\begin{bmatrix} \dot{p}_k \\ \dot{q}_k \\ \dot{r}_k \end{bmatrix} = \begin{bmatrix} -\frac{(I_{zz}-I_{yy})}{I_{xx}} q_k r_k + \frac{1}{I_{xx}} \tau_k^x \\ -\frac{(I_{xx}-I_{zz})}{I_{yy}} p_k r_k + \frac{1}{I_{yy}} \tau_k^y \\ -\frac{(I_{yy}-I_{xx})}{I_{zz}} p_k q_k + \frac{1}{I_{zz}} \tau_k^z \end{bmatrix}, \tag{7}$$

where $\dot{p}_k, \dot{q}_k, \dot{r}_k$ are the angular accelerations and $\tau_k^x, \tau_k^y, \tau_k^z$ are the commanded torques, about the respective axes, at time step k. We have assumed a diagonal inertial matrix with diagonal components I_{xx}, I_{yy}, I_{zz}.

2) Discrete-Time Flatness Derivation: We consider the output comprising of the 3-D position and yaw at time step k:

$$\mathbf{y}_k = [x_k, y_k, z_k, \psi_k]^T.$$

We show that both state $\mathbf{x}_k$, (4) in Definition 2, and input $\mathbf{u}_k = [T_k, \tau_k^x, \tau_k^y, \tau_k^z]^T$, (5) in Definition 2, can be determined from forward shifts of the output $\mathbf{y}_k$.

a) State from Forward Shifts of Output: From the discretized position dynamics we obtain the velocity at time step k in terms of forward shifts of the output, i.e., $\dot{x}_k = \frac{x_{k+1}-x_k}{\delta t}$ and $\dot{y}_k = \frac{y_{k+1}-y_k}{\delta t}$, and $\dot{z}_k = \frac{z_{k+1}-z_k}{\delta t}$. Similarly, velocity at time step $k+1$ can be described in terms of forward shifts of the output, i.e., $\dot{x}_{k+1} = \frac{x_{k+2}-x_{k+1}}{\delta t}$, $\dot{y}_{k+1} = \frac{y_{k+2}-y_{k+1}}{\delta t}$, $\dot{z}_{k+1} = \frac{z_{k+2}-z_{k+1}}{\delta t}$. Given that $[R_k^{13}, R_k^{23}, R_k^{33}]^T$ is a unit vector in (6), we can obtain this vector from forward shifts of the output as $[R_k^{13}, R_k^{23}, R_k^{33}]^T = \frac{\mathbf{t}_k}{|\mathbf{t}_k|}$ where:

$$\mathbf{t}_k = \begin{bmatrix} \frac{x_{k+2}-2x_{k+1}+x_k}{\delta t^2} \\ \frac{y_{k+2}-2y_{k+1}+y_k}{\delta t^2} \\ \frac{z_{k+2}-2z_{k+1}+z_k}{\delta t^2} + g \end{bmatrix}. \tag{8}$$

We obtain the pitch θ_k and roll ϕ_k at time step k in terms of 2 forward shifts of the output by plugging in the above expressions for $R_k^{13}, R_k^{23}, R_k^{33}$ into:

$$\theta_k = atan\left(\frac{R_k^{13}}{R_k^{33}} C_{\psi_k} + \frac{R_k^{23}}{R_k^{33}} S_{\psi_k}\right), \tag{9}$$

$$\phi_k = atan\left(\left(\frac{R_k^{13}}{R_k^{33}} S_{\psi_k} - \frac{R_k^{23}}{R_k^{33}} C_{\psi_k}\right) C_{\theta_k}\right), \tag{10}$$

where we have assumed $\theta_k \in (-\pi/2, \pi/2)$ and $\phi_k \in (-\pi/2, \pi/2)$. From these Euler angles, we can describe the rotation matrix R_k at time step k in terms of forward shifts of the output.

We can obtain similar expressions for $R_{k+1}^{13}, R_{k+1}^{23}, R_{k+1}^{33}$ at time step $k+1$ by computing $\mathbf{t}_{k+1}$ in (8). We determine the pitch θ_{k+1} and roll ϕ_{k+1} at time step $k+1$ by using $R_{k+1}^{13}, R_{k+1}^{23}, R_{k+1}^{33}$ and ψ_{k+1} in (9) - (10).

We can obtain similar expressions for the pitch θ_{k+2} and roll ϕ_{k+2} at time step $k+2$. The angular velocity of the vehicle in the inertial frame ω_k at time step k is (see [14] for details):

$$\omega_k = \begin{bmatrix} C_{\psi_k} & R_k^{12} & 0 \\ S_{\psi_k} & R_k^{22} & 0 \\ 0 & R_k^{32} & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi}_k \\ \dot{\theta}_k \\ \dot{\psi}_k \end{bmatrix} = R_k \begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix}.$$

Exploiting the Euler discretization, we obtain:

$$\begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix} = R_k^{-1} \begin{bmatrix} C_{\psi_k} & R_k^{12} & 0 \\ S_{\psi_k} & R_k^{22} & 0 \\ 0 & R_k^{32} & 1 \end{bmatrix} \begin{bmatrix} \frac{\phi_{k+1}-\phi_k}{dt} \\ \frac{\theta_{k+1}-\theta_k}{dt} \\ \frac{\psi_{k+1}-\psi_k}{dt} \end{bmatrix}. \tag{11}$$

Using the relationship between forward shifts of the output $\mathbf{y}_k$ and the Euler angles θ_k, ϕ_k, ψ_k at time step k, see (9) - (10), and $\theta_{k+1}, \phi_{k+1}, \psi_{k+1}$ at time step $k+1$ in (11), we can determine the angular velocities p_k, q_k, r_k at time step k in terms of forward shifts of the output. Similarly we can obtain expressions for $p_{k+1}, q_{k+1}, r_{k+1}$ by plugging in the expressions for $\phi_{k+1}, \theta_{k+1}, \psi_{k+1}$ and $\phi_{k+2}, \theta_{k+2}, \psi_{k+2}$ in terms of the forward shifts of the output.

b) Input from Forward Shifts of Output: We can obtain the commanded thrust at time step k in terms of forward shifts of the output from (6) as $T_k = m|\mathbf{t}_k|$, where $|\mathbf{t}_k|$ is the magnitude of $\mathbf{t}_k$ in (8). Using the Euler discretization of the rotational dynamics (7), the torques at time step k are:

$$\begin{bmatrix} \tau_k^x \\ \tau_k^y \\ \tau_k^z \end{bmatrix} = \begin{bmatrix} I_{xx}\frac{p_{k+1}-p_k}{dt} + (I_{zz} - I_{yy})\, q_k r_k \\ I_{yy}\frac{q_{k+1}-q_k}{dt} + (I_{xx} - I_{zz})\, p_k r_k \\ I_{zz}\frac{r_{k+1}-r_k}{dt} + (I_{yy} - I_{xx})\, p_k q_k \end{bmatrix}. \tag{12}$$

Using the expressions for angular velocities p_k, q_k, r_k at time step k and $p_{k+1}, q_{k+1}, r_{k+1}$ at $k+1$ in terms of the forward shifts of the output allows us to compute the torques in (12) from 4 forward shifts of the output (i.e., we require $\mathbf{y}_k, \mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \mathbf{y}_{k+3}, \mathbf{y}_{k+4}$). From (5) in Definition 2, we call this a window of $r = 4$.

B. Discrete Flatness of Multirotor Model in 2-D with First-Order Pitch-Roll Dynamics

We consider a slightly simpler model in the simulations, Sec. V, and experiments, Sec. VI, performed in this letter. The multirotor performs only 2-D motion in the $x-y$ plane and we have an underlying attitude controller. The underlying attitude controller generates a first-order pitch and roll response. This is a common (and useful) assumption for many commercial platforms (such as the DJI M600 in Fig. 1) where a black-box attitude controller is implemented onboard [16].

1) Dynamics Model: We consider the following discrete-time model (using Euler discretization approximation):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta t \dot{\mathbf{x}}_k,$$

where the state at time step k, $\mathbf{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k, \theta_k, \phi_k]^T$ comprises of the 2-D position x_k, y_k, the 2-D velocity $\dot{x}_k$, $\dot{y}_k$, and the pitch θ_k and roll ϕ_k angles. We consider the dynamics:

$$\dot{\mathbf{x}}_k = [\dot{x}_k, g\frac{R_k^{13}}{R_k^{33}}, \dot{y}_k, g\frac{R_k^{23}}{R_k^{33}}, \frac{\tilde{k}}{\tau}\theta_k^{cmd} - \frac{\theta_k}{\tau}, \frac{\tilde{k}}{\tau}\phi_k^{cmd} - \frac{\phi_k}{\tau}]^T, \tag{13}$$

where R is the rotation matrix from the body to inertial frames, $R_k^{\alpha\beta}$ is the α row and β column of rotation matrix R at time step k, τ is the first-order time constant and $\tilde{k}$ is the first-order gain for the roll and pitch dynamics. The control input $\mathbf{u}_k = [\theta_k^{cmd}, \phi_k^{cmd}]^T$ includes the commanded pitch and roll. We assume the yaw is fixed, i.e., $\psi_k = \psi_\star \forall k$.

2) Discrete-Time Flatness Derivation: We consider the output comprising of the position at time step k:

$$\mathbf{y}_k = [x_k, y_k]^T.$$

We show that both state $\mathbf{x}_k$ and input $\mathbf{u}_k$ can be determined from forward shifts of the output $\mathbf{y}_k$.

a) State from Forward Shifts of Output: Similar to Sec. IV-A, we can obtain the 2-D velocity at time step k and $k+1$ in terms of forward shifts of the output. Similarly, by using the 2-D velocity at time step k and $k+1$ in (13), we obtain the rotation at time step k in terms of forward-shifts of the output. That is, $\frac{R_k^{13}}{R_k^{33}} = \frac{x_{k+2}-2x_{k+1}+x_k}{g\delta t^2}$ and $\frac{R_k^{23}}{R_k^{33}} = \frac{y_{k+2}-2y_{k+1}+y_k}{g\delta t^2}$. The pitch θ_k and roll ϕ_k at time step k in terms of forward shifts of the output is obtain from (9) - (10) with $\psi_k = \psi_\star$.

b) Input from Forward Shifts of Output: Similar to Sec. IV-A, the rotation at time step $k+1$, i.e., $\frac{R_{k+1}^{13}}{R_{k+1}^{33}} = \frac{x_{k+3}-2x_{k+2}+x_{k+1}}{gdt^2}$ and $\frac{R_{k+1}^{23}}{R_{k+1}^{33}} = \frac{y_{k+3}-2y_{k+2}+y_{k+1}}{gdt^2}$, is used to determine the pitch θ_{k+1} and roll ϕ_{k+1} at $k+1$ in terms of forward shifts of the output. From the first-order pitch and roll dynamics in (13) we can determine the pitch and roll commands θ_k^{cmd} and ϕ_k^{cmd} from the pitch and roll at time steps k and $k+1$. The pitch and roll at time steps k and $k+1$ can be determined from 3 forward shifts of the output. Consequently, we have determined the input at time step k as a function of 3 forward shifts of the output, i.e.,

$$\mathbf{u}_k = F\left(\mathbf{y}_k, \mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \mathbf{y}_{k+3}\right). \tag{14}$$

We coin this function (14) the *Output-To-Input Map* with a window $r = 3$ in (5). The *Input-To-Output Map* is described by its inverse, i.e.,

$$\mathbf{y}_{k+3} = F^{-1}\left(\mathbf{y}_k, \mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \mathbf{u}_k\right). \tag{15}$$

C. Predictive Control Exploiting Discrete Flatness

We present a predictive control design in Fig. 2 that exploits discrete-time flatness, from Definition 2, to use a window of previous inputs and measured (flat) outputs to compute the control input. To do this, we propose three core components: *(i) Output-to-Input Map* – maps a window r of the future optimized trajectory to the input using (16); *(ii) Input-To-Output Map* – maps a window of past inputs and output measurements to constraint the future trajectory using (17); and *(iii) Output Trajectory Optimization* – optimizes the output trajectory using (18).

1) Output-To-Input Map: At time step k we require a window of future time steps of the output trajectory $\mathbf{y}_k, \mathbf{y}_{k+1}, ...\mathbf{y}_{k+r}$ to determine the input $\mathbf{u}_k$. We propose using predictive control to predict the optimized output trajectory window $\mathbf{y}_k^*, \mathbf{y}_{k+1}^*, ...\mathbf{y}_{k+r}^*$ which can be used to determined the input from (5) as:

$$\mathbf{u}_k = F\left(\mathbf{y}_k^*, \mathbf{y}_{k+1}^*, ...\mathbf{y}_{k+r}^*\right). \tag{16}$$

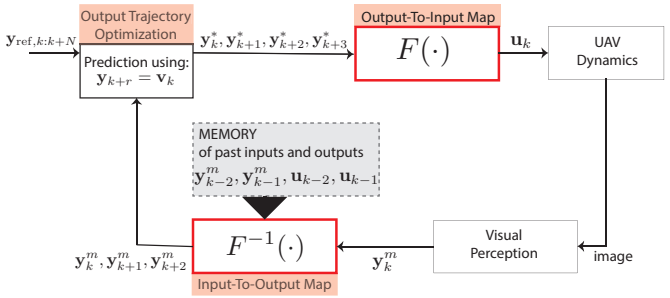

Fig. 2. Overview of predictive control (horizon N) design using a window $r = 3$ of input and (flat) output data by exploiting discrete-time flatness.

For example, we use the optimized output trajectory window $\mathbf{y}^*_k, \mathbf{y}^*_{k+1}, ...\mathbf{y}^*_{k+3}$ in the *Output-to-Input Map* in (14) for our 2-D multirotor model (Sec. IV-B) with a window size $r = 3$.

2) Input-To-Output Map: At time step k we require a window of past time step measurements of the output trajectory $\mathbf{y}^m_k, \mathbf{y}^m_{k-1}, \mathbf{y}^m_{k-r+1}$ and previously sent inputs $\mathbf{u}_{k-1}, ..., \mathbf{u}_{k-r+1}$ to determine the effect of past inputs on the future output trajectory at $\mathbf{y}^m_{k+1}, ...\mathbf{y}^m_{k+r-1}$. More precisely, we compute $\mathbf{y}^m_{k+1}, ..., \mathbf{y}^m_{k+r-1}$ as:

$$\mathbf{y}^m_{k+(j-1)} = F^{-1}\left(\mathbf{y}^m_{k-(r-j+1)}, ..., \mathbf{y}^m_{k+(j-2)}, \mathbf{u}_{k-(r-j+1)}\right), \tag{17}$$

where $j = 2, ..., r$ and r is the window size. For example, the *Input-to-Output Map* in (15) for our 2-D multirotor model (Sec. IV-B) determined a window size $r = 3$. In this case, we compute the effect of u_{k-1} and u_{k-2} on y^m_{k+1} and y^m_{k+2} as:

$$\mathbf{y}^m_{k+1} = F^{-1}\left(\mathbf{y}^m_{k-2}, \mathbf{y}^m_{k-1}, \mathbf{y}^m_k, \mathbf{u}_{k-2}\right),$$
$$\mathbf{y}^m_{k+2} = F^{-1}\left(\mathbf{y}^m_{k-1}, \mathbf{y}^m_k, \mathbf{y}^m_{k+1}, \mathbf{u}_{k-1}\right).$$

This is used to constrain the output trajectory optimization to account for the effects of previously sent inputs.

3) Output Trajectory Optimization: At each time step, we solve the following optimization problem:

$$\begin{aligned} \mathbf{y}^*_{k:k+N} = \underset{\mathbf{y}_{k:k+N}}{\text{argmin}} \quad & J\left(\mathbf{y}_{k:k+N}\right) \\ \text{s.t.} \quad & \mathbf{y}_{k+j-1} = \mathbf{y}^m_{k+j-1}, \forall j = 1, .., r, \end{aligned} \tag{18}$$

where N is the prediction horizon, $J(\cdot)$ is a selected trajectory cost function (often selected as a quadratic with a tracking error term and a regularization term regulating how quickly the output changes). Additional constraints on the output trajectory can also be enforced if necessary.

V. Simulations

We consider the dynamics in (13) from Sec. IV-B and consider a 1-D motion in the x-direction with $\psi^* = 0$, $y = 0$, $\dot{y} = 0$ and $\phi = 0$. The dynamics (13) are executed at 200 Hz. The controllers are run at 50 Hz. We consider the proposed predictive control in *Discrete-Flatness* exploiting discrete-time flatness with the cost in (18):

$$J(\cdot) = \sum_{k=0}^{N} \left(\mathbf{y}_k - \mathbf{y}_{ref,k}\right)^T \mathbf{Q} \left(\mathbf{y}_k - \mathbf{y}_{ref,k}\right) + \bar{\mathbf{v}}_k^T \mathbf{R} \bar{\mathbf{v}}_k$$

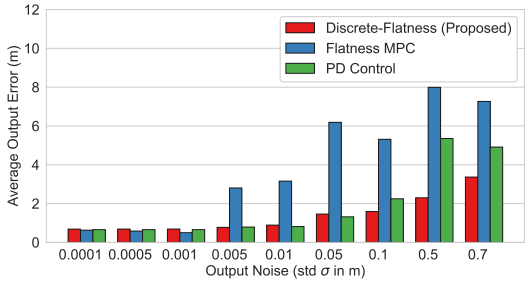

Fig. 3. Simulation comparison of the effect of position noise on the tracking performance of *PD Control* in green, *Flatness MPC* (predictive control using continuous-time flatness) in blue, and our discrete-time flatness or *Discrete-Flatness* in red. Measurements of the x-position have zero-mean Gaussian noise $\mathbf{y}^m = \mathbf{y} + [\mathcal{N}(0, \sigma^2), 0]^T$, where we vary the standard deviation σ (Output Noise). For each σ, we compute the average position or output tracking error over 10 trials. The proposed *Discrete-Flatness* (red) is more robust to position noise compared to the traditional *Flatness MPC* (blue), which relies on higher-order position derivative estimates for predictive control.

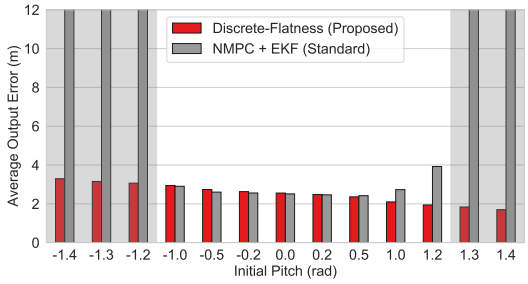

Fig. 4. Simulation comparison of the effect of initial state uncertainty on the performance of the proposed *Discrete-Flatness* in red and *NMPC+EKF*, i.e., using a standard nonlinear MPC with an extended Kalman filter (EKF) for state estimation, in grey. Our proposed *Discrete-Flatness* has robustness to significant initial uncertainty in the pitch, because unlike the EKF we do not linearize about the the current estimate and instead implicitly use the relationship between the state at time-step k and the position (or output) trajectory from k to $k + r$ as described by the discrete-time flatness property in (4) to discard biased state estimates after r time steps. The shaded grey region shows where the *NMPC+EKF* approach is unstable.

where $\bar{\mathbf{v}}_k = \mathbf{y}_{k+3} - 3\mathbf{y}_{k+2} + 3\mathbf{y}_{k+1} - \mathbf{y}_k$, $N = 200$ is the prediction horizon, $\mathbf{Q}$ weights the error with the reference point and $\mathbf{R}$ weights a regularisation term.

a) Robustness to noisy position and higher-order derivative estimates: We start from no motion at the origin, i.e., $\mathbf{x}_k = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$ at $k = 0$. We compare the effect of noise on the performance of *Flatness MPC*, that uses continuous-time differential flatness as in [16] and relies on position and first-order finite-difference velocity and acceleration estimates in the state feedback, *PD Control*, which relies on position and finite-difference velocity estimates, and the proposed *Discrete-Flatness* control. At each noise level or σ, we compute the average position tracking error for the feasible reference trajectory $\mathbf{y}_{ref} = Atsin(\omega t)$ where $A = 1.4, \omega = 1.0$ over 10 trials for each controller. In Fig. 3, we observe that at low position noise (i.e. $\sigma < 0.0005$) all controllers are tuned to have similar performance. However, as the position noise increases the performance of *Flatness MPC* (blue in Fig. 3) significantly worsens as a result of heightened noise in the higher-order derivatives used in the state estimate. PD

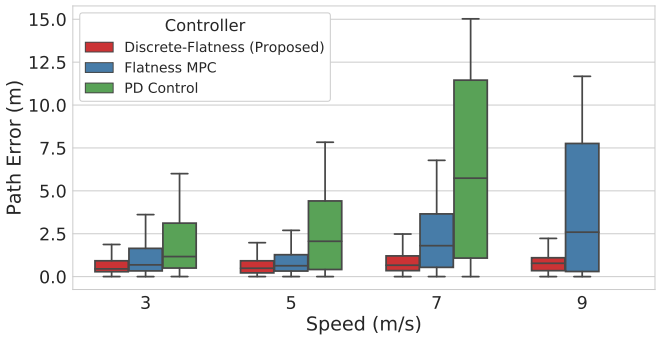

Fig. 5. Comparison of path error at increasing desired speeds for autonomous vision-based flight of multirotor in Fig. 1 using *PD Control* (green), *Flatness MPC* (blue) and our proposed *Discrete-Flatness* controller (red). The box plots are computed over 3 trials (for each controller at each speed) repeating the L-shaped path in Fig. 7. Our proposed approach *Discrete-Flatness* (red) outperforms related controllers by performing prediction and not relying on noisy and delayed state estimation. The vehicle under *PD Control* (green) goes unstable at 9 m/s. The performance under *Flatness MPC* (blue) degrades as the speed increases. Our proposed *Discrete-Flatness* (red) maintains low path error for all desired speeds and achieves an average path error reduction of $20-40\%$ for low speeds $\leq 5m/s$ and $65-80\%$ for high speeds $> 5m/s$ over *Flatness MPC*.

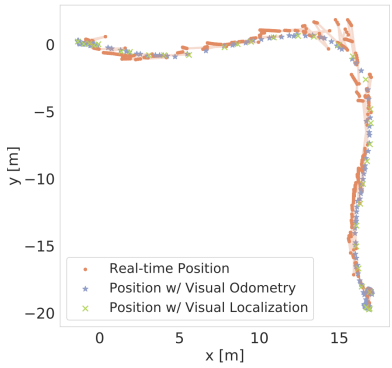

Fig. 6. Comparison of the estimated noisy real-time position at 50 Hz (red), the estimated position from VO and bundle adjustment (blue) and including visual localization (green) for a path flown at 9 m/s.

Control outperforms *Flatness MPC* for higher position noise (i.e. $\sigma > 0.005$) because we consider a feasible trajectory and it does not rely on a state estimate to make future model predictions. Our proposed approach is a predictive controller that achieves better performance despite high position noise (i.e. $\sigma > 0.005$) by avoiding higher-derivatives of the position estimates.

b) Robustness to initial state uncertainty: In simulation, we consider a multirotor with motion in 1-D in the x-direction starting at an initial position and velocity of zero. The objective is to move to a reference point 10 m away. We assume that measurements are obtained at the same rate of 50 Hz as the control input. In Fig. 4, we compare the average error for our proposed *Discrete-Flatness* and *NMPC+EKF*, which uses a standard extended Kalman filter (EKF) and nonlinear model predictive controller (NMPC). Both controllers assume the quadrotor starts at rest (i.e., the state is zero). The EKF considers an initial state uncertainty with standard deviation of 0.5 rad on the pitch angle. We compare the performance of each of the approaches for different true initial pitch values. We assume no measurement or process noise.

VI. Experiments

A. Hardware Setup

As shown in Fig. 1, we use a DJI Matrice 600 Pro with attached Ronin-MX gimbal. All processing, including visual navigation, planning and control happens onboard the UAV on the primary computer, NVIDIA Tegra TX2 module (6 ARM cores + 256 core Pascal GPU). The primary computer uses a serial Transistor-Transistor Logic (TTL) connection to connect with the on-board M600 autopilot which provides vehicle data (e.g., the gimbal state) and the interface for sending control commands to the vehicle. There is a StereoLabs ZED stereo camera mounted to the Ronin-MX gimbal to provide greyscale imagery with resolution 672 x 376 at 15 Hz. The Tegra TX2 runs NVIDIA L4T v28.2 and an XBee 900 Mhz wireless link is used to communicate with a ground station where we can monitor any status changes and send high-level commands (i.e., changing autonomy states).

B. Visual Teach and Repeat Navigation Overview

This section summarizes the visual teach and repeat (VT&R) navigation system used in the experiments [2]. In VT&R, we consider two phases. During the *teach* phase, the UAV flies using autonomous GPS waypoint following. The purpose of the teach path is to create a map of visual landmarks along the path. This taught path is stored as a set of vertices and edges which include the landmarks observed at each vertex. During the *repeat* phase, GPS is disabled and the vehicle needs to perform autonomous navigation (of the teach path) while the VT&R algorithm performs visual localization using a local segment from the taught path. In these experiments, we consider the *controller implemented during repeat*. During *repeat* visual odometry (VO) is performed in the same manner as during teach with an additional thread running visual localization, i.e., visual matching to the map created during *teach*, see [2] for more details. The high-rate real-time pose (or higher-dimensional state) used by the controllers is determined by extrapolating the last VO pose forward using Simultaneous Trajectory Estimation And Mapping (STEAM). STEAM was first introduced in [21] and details on the implementation used in this paper can be found in [22]-[23]. STEAM uses low-rate position estimates from vision to estimate a continuous-time state-trajectory, that can be queried by the controller at the current time, but relies on an accurate motion model or prior. Fig. 6 shows the position estimate (red) queried by the controller from STEAM at each control time step for a 9 m/s trajectory. Associated with each of these position estimates is also a velocity and acceleration estimate for example. As demonstrated in Fig. 6 for sharp turns at higher speeds STEAM does not accurately estimate the trajectory. *Flatness MPC* relies on position, velocity and acceleration estimates from STEAM. *PD Control* relies on position and velocity estimates from STEAM. Our proposed *Discrete-Flatness* approach relies on only the position estimate as feedback.

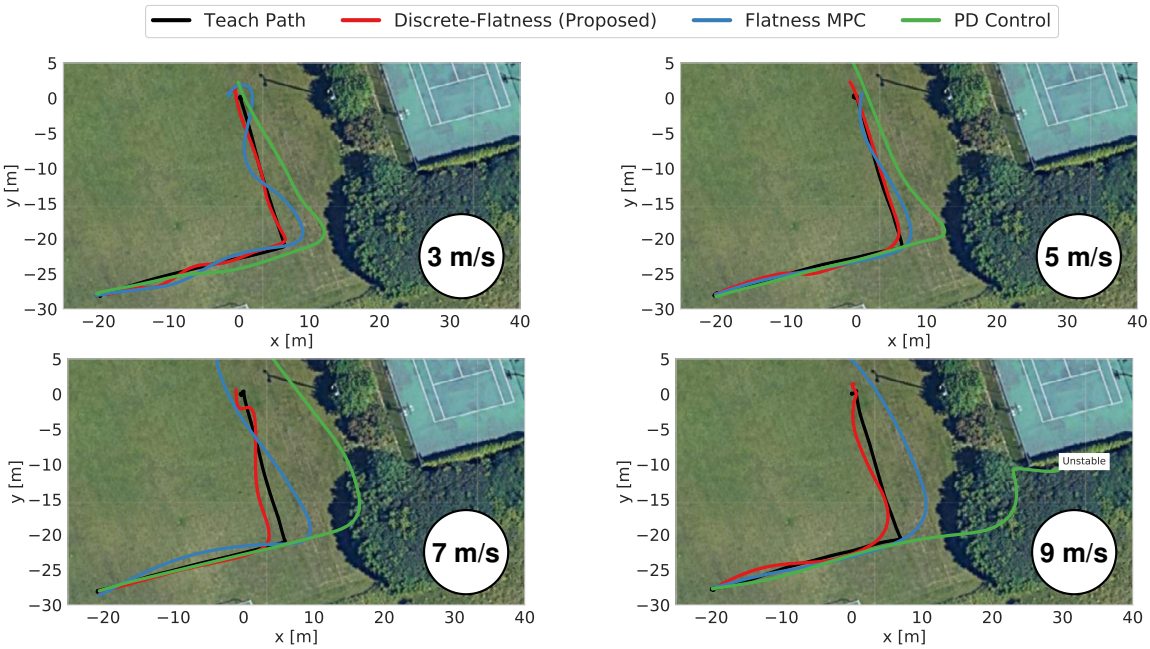

Fig. 7. Visualization of path flown when repeating the *teach* path (black) under vision-based navigation using different controllers with a desired speed of (a) 3 m/s (b) 5 m/s (c) 7 m/s and (d) 9 m/s. The L-path starts at the lower left corner (at approximately $(-20, -30)$) and ends at the origin. We show one trial (out of three used in Fig. 5) for each controller. We compare the paths flown using *PD Control* (green), *Flatness MPC* (blue) and our proposed *Discrete-Flatness* (red). Our proposed *Discrete-Flatness* (red) outperforms the alternative controllers that rely on noisy/delayed state estimates. Unlike the alternative controllers our proposed approach achieves high performance (see Fig. 5) by turning before the corner even at high speeds.

C. Comparison with Related State-Feedback Controllers

We compare three controllers (*PD Control* [2], *Flatness MPC* [16] [24], and our proposed *Discrete-Flatness* controller) implemented at 50 Hz. We consider only flight in 2-D (fixed yaw and no motion in the z-direction). All controllers are used to determine roll and pitch commands as described in Sec. IV-B.

a) Reference Generation: We use a simple reference generation approach. The geometric teach path is created by connecting keyframes from teach with straight lines segments. The reference position and velocity at time step k for the non-predictive controller (*PD Control*) is determined by finding the closest point on the path, i.e., $\mathbf{y}_{ref,k} = \mathbf{y}_{closest}$, and computing the reference velocity $\dot{\mathbf{y}}_{ref,k}$ as the desired speed in the direction of the next keyframe. *PD Control* [2] determines the commands at time step k by weighting the position error $\mathbf{y}_k - \mathbf{y}_{ref,k}$ with the velocity error $\dot{\mathbf{y}}_k - \dot{\mathbf{y}}_{ref,k}$. In the predictive controllers (*Flatness MPC* and *Discrete-Flatness*) we compute the reference on the path with a fixed desired speed. At time step k we compute the reference by finding the closest point $\mathbf{y}_{closest}$ on the path, i.e., $\mathbf{y}_{ref,k} = \mathbf{y}_{closest}$. We compute the reference at the next time step by moving in the direction of the reference velocity (desired speed in the direction of the next keyframe) $\dot{\mathbf{y}}_{ref,k}$ as $\mathbf{y}_{ref,k+1} = \mathbf{y}_{ref,k} + \delta t \dot{\mathbf{y}}_{ref,k}$, for the prediction horizon $k+1, k+2, ...k+N$. The reference velocity $\dot{\mathbf{y}}_{ref,k}$ changes as the output reference moves to the next straight line segment of the path.

b) Parameters: We consider a lookahead of $2s$ for *Discrete-Flatness* and $1.5s$ for *Flatness MPC*. These were the maximum horizons for each controller that we could reliably compute at a rate of 50 Hz (20 ms). The lookahead difference is a result of *Flatness MPC* requiring a few additional matrix multiplications at each time step. We tune the weights for all controllers to maximize stable performance.

c) Results: We consider a simple L-shape teach path, black in Fig. 7, at a fixed altitude of 10 m above ground. In Fig. 5, we present a box plot of the path error for each controller with desired speeds of 3 m/s, 5m/s, 7m/s and 9 m/s. These results are obtained from 3 repeated trials for each controller at each speed. As expected, the *PD Control* (green) has the worst performance as it is unable to predict and react to the sharp turn in the path. At 9 m/s the *PD Control* is not reliable and causes the vehicle to go unstable. *Flatness MPC* (blue) has good performance for 3 m/s and 5 m/s, however, the performance dramatically degrades at 7 m/s and 9 m/s. Similar to the simulation in Sec. V, *Flatness MPC* (blue) relies on higher-order derivatives of position (velocity and acceleration). STEAM filters some of the noise in these estimates but introduces a delay. Consequently, inaccurate real-time estimation of these quantities prevents the *Flatness MPC* from making an accurate forward model prediction. As such the vehicle does not turn before the corner (bottom right of black teach path in Fig. 7) for 7 m/s and 9 m/s. Our proposed *Discrete-Flatness* (red) approach maintains a low tracking error at all speeds by turning before the corner as highlighted in Fig. 7.

d) Demonstration of Discrete-Flatness up to 10 m/s: We demonstrate the efficacy of our proposed *Discrete-Flatness* controller for VT&R at speeds up to 10 m/s by flying three paths (at fixed 20 m altitude above ground) - a D-shaped path (at 10 m/s), an S-shaped path (at 3 m/s) and an L-shaped path (at 8 m/s). The DSL flights are shown in Fig. 8 with

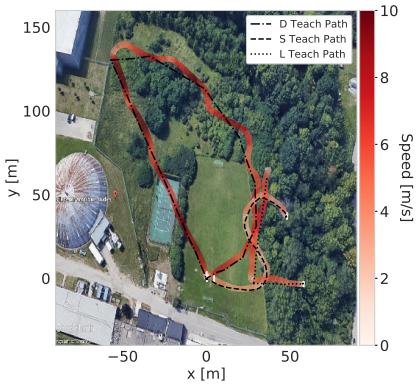

Fig. 8. Visualization of 3 additional paths flown using vision-based navigation and the proposed *Discrete-Flatness* control achieving speeds up to 10 m/s. We fly three paths: 1) a D-shaped path (at 10 m/s), 2) an S-shaped path (at 3 m/s) and 3) an L-shaped path (at 8 m/s). This spells out the abbreviation of our lab "Dynamic System Lab" or DSL.

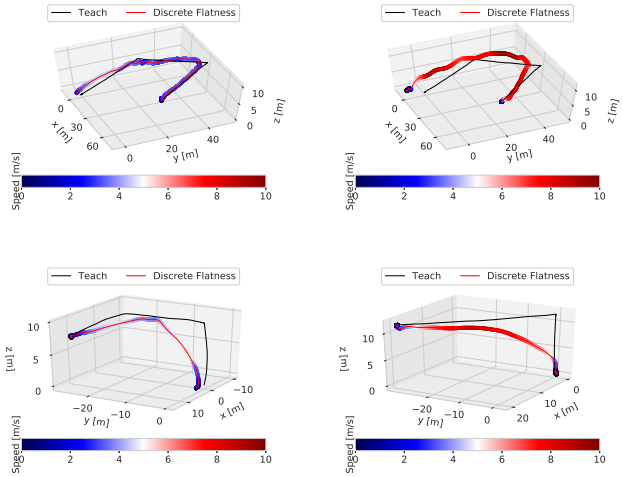

Fig. 9. Visualization of various 3D paths flown using vision-based navigation and the proposed *Discrete-Flatness* control.

the speed profile overlay. The wobbles in the D-shaped path are a result of the heightened real-time position noise when turning as illustrated in Fig. 6. We have extended the proposed *Discrete-Flatness* control to 3D and show outdoor trajectories at various speeds using vision-in-the-loop in Fig. 9.

VII. Conclusion and Future Work

Exploiting discrete-time flatness for outdoor high-speed vision-based navigation, with potentially noisy high-rate real-time output (position) measurements, is a promising approach as it allows closed-loop visual autonomy without state estimation. This is particularly relevant to future work on learning-based control that could improve performance by simultaneously learning $F(\cdot)$ in (14) and $F^{-1}(\cdot)$ in (15) using *only input and output data*. Future work will also investigate the effect of window size r on learning performance and extend our approach to different input and output rates. Video: https://tinyurl.com/flyoutthewindow

References

[1] D. Thakur, G. Loianno, L. Jarin-Lipschitz, A. Zhou, V. Kumar, "Autonomous inspection of a containment vessel using a micro aerial vehicle", in *Proc. Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1-7, 2019.

[2] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig and T. D. Barfoot, "There's no place like home: visual teach and repeat for emergency return of multirotor UAVs during GPS failure," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 161-168, 2019.

[3] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "AlphaPilot: autonomous drone racing," in *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[4] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen. "Teach-repeat-replan: a complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526-1545, 2020.

[5] Y. Song, M. Steinweg, E. Kaufmann and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021, *Accepted*.

[6] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: optimal methods meet learning for drone racing," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, pp. 690-696, 2019.

[7] T. Nguyen, A. H. Zaini, C. Wang, K. Guo and L. Xie, "Robust target-relative localization with ultra-wideband ranging and communication," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, pp. 2312-2319, 2018.

[8] S. Dean, N. Matni, B. Recht and V. Ye, "Robust guarantees for perception-based control," in *Proc. Conference on Learning for Dynamics and Control (L4DC)*, vol. 120, pp. 350-360, 2020.

[9] L. Jarin-Lipschitz, R. Li, T. Nguyen, V. Kumar and N. Matni, "Robust, perception-based control with quadrotors," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pp 7737-7743, 2020.

[10] L. Brunke, S. Zhou, and A. P. Schoellig, "RLO-MPC: robust learning-based output feedback MPC for improving the performance of uncertain systems in iterative tasks," in *Proc. Int. Conf. on Decision and Control (CDC)*, 2021, *Accepted*.

[11] J Lorenzetti and M. Pavone, "A simple and efficient tube-based robust output feedback model preditive control scheme," in *Proc. European Control Conference (ECC)*, pp. 1775-1782, 2020.

[12] M. Kogel and R. Findeisen, "Robust output feedback mpc for uncertain linear systems with reduced conservatism," in *Proc. IFAC World Congress*, vol. 50, no. 1, pp. 10685-10690, 2017.

[13] D. A. Copp and J. P. Hespanha, "Simultaneous nonlinear model predictive control and state estimation," *Automatica*, vol. 77, pp. 143-154, 2017.

[14] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *Int. Journal of Control*, vol. 61, no. 6, pp. 1327-1361, 1995.

[15] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, pp. 2520-2525, 2011.

[16] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 6740-6745, 2018.

[17] P. Guillot and G. Millerioux, "Flatness and submersivity of discrete-time dynamical systems," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 337-342, 2020.

[18] B. Kolar, J. Diwold and M. M. Schöberl, "Necessary and sufficient conditions for difference flatness," arXiv:1909.02868, 2019.

[19] J. Diwold, B. Kolar and M. Schöberl, "A trajectory-based approach to discrete-time flatness," *Control Systems Letters*, vol. 6, pp. 289-294, 2020.

[20] M. Alsalti, J. Berberich, V. G. Lopez, F. Allgower, M. A. Müller, "Data-based system analysis and control of flat nonlinear systems," arXiv:2103.02892, 2021.

[21] S. Anderson and T. D. Barfoot, "Full STEAM ahead: exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3)," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 157-164, 2015.

[22] J. N. Wong, D. J. Yoon, A. P. Schoellig and T. D. Barfoot, "Variational inference with parameter learning applied to vehicle trajectory estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5291-5298, 2020.

[23] J. N. Wong, D. J. Yoon, A. P. Schoellig and T. D. Barfoot, "A data-driven motion prior for continuous-time trajectory estimation on SE(3)," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1429-1436, 2020.

[24] M. Greeff, T. D. Barfoot and A. P. Schoellig, "A perception-aware flatness-based model predictive control for fast vision-based multirotor flight," in *Proc. IFAC World Congress*, vol. 53, no. 2, 9412-9419, 2020.