# Flying Flat Out: Fast Multirotor Flight Using Vision-Based Navigation in Real-World Environments

by

Melissa Greeff

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto

Flying Flat Out: Fast Multirotor Flight Using Vision-Based Navigation in Real-World Environments

Melissa Greeff
Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto
2022

# Abstract

Multirotor unmanned aerial vehicles (UAVs) are mechanically simple and highly maneuverable robots that are suitable to a wide range of applications such as infrastructure inspection, transportation, search-and-rescue missions, and mapping operations. Reliable autonomous multirotor flight needs to expand beyond lab demonstrations to real-world environments. To enable this, in the first part of this thesis, we present computationally efficient control algorithms, by exploiting a property of the dynamics known as differential flatness. We exploit this property to enable efficient prediction and safe learning from online data. As a result, we develop safe high-performance control by accounting for nonlinear and unknown dynamics in a computationally tractable way. In the second part of this thesis, we explore some of the challenges to high-speed autonomous vision-based flight. Real-world environments may be GPS-denied and vision-based navigation, relying predominantly on an onboard camera, is a lightweight and cost-effective alternative. Most standard controllers are perception-agnostic and tend to assume *(i)* the action computed by the controller has no effect on the ability of vision-based navigation to determine location and *(ii)* perfect state estimation is obtained. These assumptions often limit the reliability and performance of perception-agnostic controllers for autonomous vision-based flight. In contrast, we present perception-aware control algorithms that account for partial visual knowledge of the environment and plan despite imperfect state estimation. These approaches are validated through outdoor experiments on a DJI M600 multirotor where we demonstrate autonomous vision-based flight at speeds up to 10 m/s.

# Acknowledgements

Firstly, I would like to thank my family in South Africa for their support and believing in me. Thank you, mom, for reminding me how insanely proud of me you are. Thank you, dad, for being an amazing role model throughout my life on how to solve problems and for reading my accepted papers. You showed me how fun it is to be an engineer! Thank you to my sister for our weekly chats and for being my cheerleader.

I would like to thank my advisor, Prof. Angela Schoellig, for guiding me through these years and for supporting my ideas. I am grateful for your mentorship. Thank you to my thesis committee for the their feedback. It lead to new directions and made me a better researcher. I would like to thank all my lab mates at the Dynamic Systems Lab. I enjoyed our coffee chats, lunches at Vector, and conference adventures. It was amazing to learn from so many talented people. In particular, I would like to thank Karime and SiQi for their friendship. I would like to thank everyone that I worked with on the DRDC outdoor vision-based flight project.

Thank you to the Mazzuca family - Rose, Serge and Chris. You made sure I was well fed and looked after. Family dinners and special occasions were a much needed break. I appreciate all your support.

And lastly, this thesis is dedicated to my partner, Matthew Mazzuca. Thank you for believing in me and for being my rock. I could not have done this without you.

# Contents

# List of Symbols

| | |
|---|---|
| [...] | Row vector |
| $[...]^T$ | Column vector |
| $\alpha(\cdot), \beta(\cdot), a(\cdot), b(\cdot)$ | Nonlinear functions (assumed to be unknown) for control-affine single-input single-output systems |
| $\dot{\mathbf{x}}$ | First-order time derivative of $\mathbf{x}$ |
| $\dot{\psi}_{\text{cmd}}$ | Commanded yaw rate of multirotor |
| $\dot{z}_{\text{cmd}}$ | Commanded vertical velocity of multirotor |
| $\hat{f}_1(\cdot), \hat{f}_2(\cdot)$ | Known functions in the nominal model dynamics for control-affine single-input single-output systems |
| $\lambda_{\min}(\cdot)$, $\lambda_{\max}(\cdot)$ | Minimum and maximum eigenvalues of a matrix |
| $\mathbb{E}(\cdot)$ | Expectation of a random variable |
| $\mathbb{V}(\cdot)$ | Variance of a random variable |
| $\hat{\mathbf{v}}$ | Predicted sequence of flat inputs |
| $\hat{\mathbf{z}}$ | Predicted sequence of flat states |
| $\tilde{\mathbf{K}}$ | Linear quadratic regulator gain |
| $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$ | Optimal control cost function weights |
| $\mathbf{a}$ | Input vector to Gaussian process |
| $\mathbf{a}^*$ | Query input vector to Gaussian process |
| $\mathbf{C}_{\mathfrak{l}\mathfrak{t}}$ | Rotation matrix from frame $\mathfrak{t}$ to frame $\mathfrak{l}$ |

| | |
|---|---|
| $\mathbf{e}$ | Trajectory tracking error |
| $\mathbf{K}$ | Covariance matrix |
| $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{l}\mathfrak{t}}$ | Vector from frame $\mathfrak{t}$ to frame $\mathfrak{l}$ expressed in frame $\mathfrak{o}$ |
| $\mathbf{R}$ | Rotation matrix of the multirotor body frame with respect to an inertial frame |
| $\mathbf{T}_{\mathfrak{l}\mathfrak{t}}$ | Transform from frame $\mathfrak{t}$ to frame $\mathfrak{l}$ |
| $\mathbf{u}$ | Input vector of the system |
| $\mathbf{u}_k$ | Vector $\mathbf{u}$ at time step $k$ |
| $\mathbf{v}$ | Flat input |
| $\mathbf{v}_d$ | Desired flat input |
| $\mathbf{x}$ | State vector of the system |
| $\mathbf{x}^{(r)}$ | $r^{th}$ time derivative of $\mathbf{x}$ |
| $\mathbf{y}$ | Output vector of the system |
| $\mathbf{y}_{\text{ref}}$ | Reference output of the system |
| $\mathbf{z}$ | Flat state |
| $\mathbf{z}_d$ | Desired flat state |
| $\mathcal{D}$ | Observed data used in Gaussian process |
| $\mathcal{N}(\cdot,\cdot)$ | Normal or Gaussian distribution |
| $\mathcal{N}(\mu_Y, \mu_X, \sigma_Y^2, \sigma_X^2, \rho)$ | Bivariate correlated normal distribution with means $\mu_Y, \mu_X$, variances $\sigma_Y^2, \sigma_X^2$ and a coefficient of correlation $\rho$ |
| $\mathfrak{w}, \mathfrak{b}, \mathfrak{l}, \mathfrak{t}, \mathfrak{a}, \mathfrak{n}$ | Vertices in the visual teach and repeat localization chain |
| $\mu(\cdot)$ | Gaussian mean |
| $\phi$ | Roll angle of multirotor |
| $\phi_{\text{cmd}}$ | Commanded roll angle of multirotor |
| $\psi$ | Yaw angle of multirotor |

| | |
|---|---|
| $\psi(\mathbf{z}, u)$ | Nonlinear function of flat state $\mathbf{z}$ and input $u$ for single-input single-output systems |
| Pr | Probability |
| $\sigma^2(\cdot)$ | Gaussian variance |
| $\tau$ | Identified time constant |
| $\theta$ | Pitch angle of multirotor |
| $\theta_{\text{cmd}}$ | Commanded pitch angle of multirotor |
| $\tilde{k}$ | Identified gain |
| $f(\cdot, \cdot)$ | System dynamics |
| $f_1(\cdot), f_2(\cdot)$ | Unknown functions in the system dynamics for control-affine single-input single-output systems |
| $h(\cdot)$ | Observation model |
| $h_L(\cdot)$ | Landmark probability model |
| $k(\cdot, \cdot)$ | Kernel function |
| $m$ | Dimension of the input and output |
| $N$ | Prediction horizon in model predictive control |
| $n$ | Dimension of the state |
| $N_d$ | Number of data points used in Gaussian process |
| $u$ | Input of single-input single-output systems |
| $v$ | Flat input for single-input single-output systems |
| $V(\cdot)$ | Lyapunov function |
| $v_d$ | Desired flat input for single-input single-output systems |
| $v_{\text{cmd}}$ | Commanded flat input for single-input single-output systems |
| $x, y, z$ | Position of the multirotor with respect to an inertial frame |

|  |  |
|---|---|
| $y$ | Output of single-input single-output systems |
| $y_i$ | $i^{th}$ component of $\mathbf{y}$ |

# Chapter 1

# Introduction

In recent years, there has been a push toward developing robust and reliable autonomous robot navigation to facilitate more efficient transportation, exploration of dangerous environments, and assist people in performing difficult tasks. This requires expanding the limited operating regime of current robot navigation to enable autonomy in less structured and changing environments and in safety-critical interactions.

Multirotor unmanned aerial vehicles (UAVs) are mechanically simple and highly maneuverable robots that are suitable to a wide range of applications such as infrastructure inspection, see [Bircher et al., 2016] or [Thakur et al., 2019], transportation, search-and-rescue missions, see [Rudol et al., 2008], and mapping operations, see [Han et al., 2013]. To support these applications, reliable autonomous multirotor navigation needs to expand beyond lab demonstrations to real-world environments.

Even in lab demonstrations, exploiting the ability of multirotor UAVs to achieve fast, agile flight requires accounting for their nonlinear dynamics in *controllers* that can operate in real-time. Moreover, real-world environments have the additional challenges of achieving this despite unknown disturbances and changing dynamics. For example, during outdoor flight, multirotor UAVs experience unknown disturbances as a result of wind coming from varying directions and at varying speeds. Changing multirotor dynamics may also be a result of changing intrinsic properties, such as varying payload (e.g., different onboard sensors or packages for 'drone' delivery). Furthermore, to avoid communication delays with a ground station and/or relying on additional infrastructure it is beneficial for computation to take place onboard the UAV on a lightweight computer.

The first challenge that we seek to address in this thesis is that safe high-performance fast multirotor flight in real-world environments requires planning and accounting for nonlinear and unknown dynamics with *computationally tractable con-*

*trol* algorithms that can be used in real-time operation, on-board, in a high-frequency feedback loop.

In this thesis, we address this challenge by exploiting a structural property of many nonlinear models known as differential flatness, see [Fliess et al., 1995]. Many physical systems, including cranes, cars with trailers and *multirotors*, see [Mellinger et al., 2011], can be described by nonlinear models exhibiting the differential flatness property. Intuitively, differential flatness allows us to separate the nonlinear model into a linear dynamics component and a nonlinear transformation. A common control approach for differentially flat systems is to try to cancel the nonlinear term and design a controller using the remaining linear dynamics. This can be computationally efficient to design. However, in reality, performance and safety are limited by the mismatch between the nominal model (for example, used to cancel the nonlinear term) and the actual system dynamics (which may include unknown disturbances and changing dynamics). In **Part I: Exploiting Flatness Structure** of this thesis, see Fig. 1.1, we explore control design that exploits differential flatness but is able to achieve high performance and safety despite the mismatch between the nominal model and actual system. In the first part of this thesis, we develop controllers and theory for systems that are differentially flat and use multirotor flight as an example.

The second challenge is that real-world environments may be GPS-denied (i.e., we cannot rely on GPS for accurate global positioning). Using GPS is limited to environments with sufficient satellite coverage and is susceptible to multipath propagation and intentional jamming. For safety, government regulations have often restricted UAVs to Visual Line of Sight (VLOS) to allow manual piloting of the UAV in the event of GPS loss. To facilitate safe beyond VLOS UAV operations, alternative autonomous navigation solutions are required. These solutions should be able to act as a backup during GPS loss or facilitate standalone use. This has motivated developing *vision-based navigation* that relies primarily on lightweight, inexpensive onboard camera sensors.

In recent years, significant advancements have been made in enabling high-performance autonomous flight using vision-based sensing as a lightweight and versatile alternative, see, for example, [Warren et al., 2019], [Foehn et al., 2020] or [Gao et al., 2020]. Inspired in part by the DARPA Fast Lightweight Autonomy (FLA) challenge, see [DARPA], and the potential for autonomous drone racing, see [Kaufmann et al., 2019], there has been a significant push toward faster vision-based multirotor flight, see, for example, [Mohta et al., 2018] or [Beul et al., 2018]. One

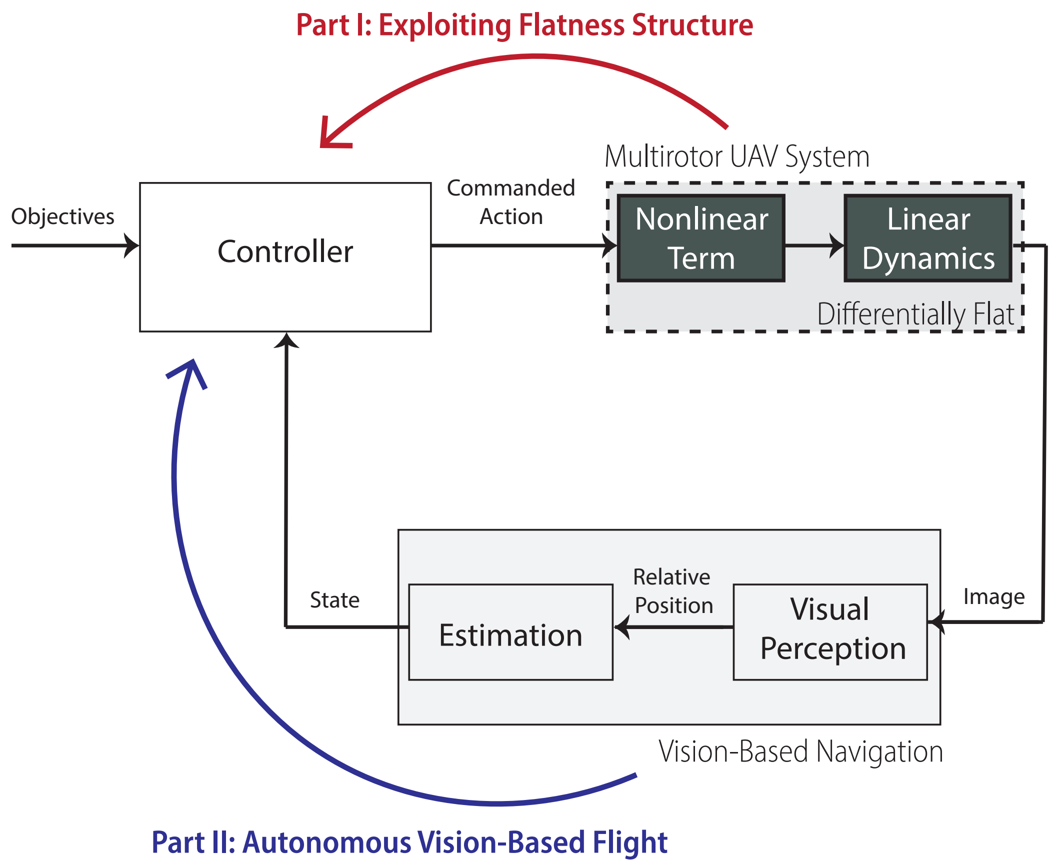


Figure 1.1: In this thesis we focus on controller design and algorithms useful to (i) multirotor UAVs and for (ii) autonomous vision-based flight. In **Part I: Exploiting Flatness Structure** we explore the role of differential flatness to achieve safety and high-performance despite unknown dynamics in a computationally tractable way. In **Part II: Autonomous Vision-Based Flight**, we address specific challenges to high-performance vision-based flight that are often neglected by traditional perception-agnostic control design.

such vision-based approach uses a Visual Teach and Repeat (VT&R) framework, see [Furgale et al., 2010], that allows the UAV to repeat a previously taught path by matching current visual features to those in a locally metric map created during a teach phase, see [Gao et al., 2020] or [Warren et al., 2019].

The work in this thesis relies on a classical autonomy pipeline, including subtasks for sensing, localization, estimation, planning, and control. There have been promising results in an end-to-end approach that can achieve high-speed vision-based flight, see [Loquercio et al., 2021]. Such an end-to-end approach does not have the same processing latency that can compound through the classical pipeline. However, there are still open questions on the quantity and diversity of environment data required to ensure generalization across various outdoor environments. Furthermore, current state-of-the-art end-to-end approaches do not have the interpretability of the classical autonomy pipeline, which makes it challenging to debug or diagnose when the system does not perform well. Within this classical autonomy pipeline, in this thesis we explore tighter integration between localization and estimation modules and control. The interpretability and modularity of our control methods allow them to be easily adapted to new UAV systems (for example, when using a new camera or changing the UAV payload).

This thesis considers the Visual Teach and Repeat (VT&R) framework and focuses on the *controller* required to autonomously repeat a previously taught path. The controller's role is to compute a commanded action to achieve desired objectives (for example, to repeat the previously taught path at some desired speed) using feedback from vision-based navigation. The feedback is often in the form of a state estimate that can include the UAV's location relative to the path, orientation, velocity and acceleration information. Many state-of-the-art controllers tend to make two flawed assumptions for vision-based navigation:

- **Assumption 1:** The commanded action computed by the controller has no effect on the ability of vision-based navigation to determine the UAV's location (or localization).
- **Assumption 2:** The commanded action computed by the controller relies on perfect state estimation from vision-based navigation.

**Assumption 1:** Relying on vision may require operating despite *partial knowledge of the environment*. For example, in many vision-based navigation systems knowledge of the world may be limited to a visual-map. The onboard controller relies on feedback from the vision-based navigation system to determine the actions of the multirotor. If the controller is agnostic to the vision-based navigation system, it can determine a commanded action that prioritizes an objective (e.g., flying at some desired speed) but leads the UAV to a location where the vision system may not be able to localize itself with respect to the map or may only be able to obtain a poor estimate of the UAV location. Moreover, poor performance of either subsystem (controller or vision-navigation) is reinforced through this feedback loop.

**Assumption 2:** Controllers for multirotor UAVs tend to rely on high-rate state estimates (generally 50-200 Hz for position control to compensate for their fast, agile dynamics). In contrast, due to computational requirements, visual perception often estimates lower-rate relative position information (currently 10-35 Hz). The mismatch between the control requirements and visual measurement is addressed with an additional state estimator that uses a prediction motion model and/or integrates IMU measurements, to determine high-rate state estimates. An accurate full state estimate is often challenging to obtain due to typically noisy IMU measurements, an infrequent position update from the vision system and an imperfect motion model used to obtain high-rate state estimates required by the controller. Assuming perfect state estimation may, in practice, significantly limit the performance and robustness of the controller.

Therefore, the second challenge that we seek to address is that reliable high-performance flight in real-world environments using *vision-based navigation* requires developing control algorithms that can account for partial knowledge of the environment and plan despite imperfect state estimation. We present methods toward addressing this challenge in a VT&R framework in **Part II: Autonomous Vision-Based Flight**, see Fig. 1.1.

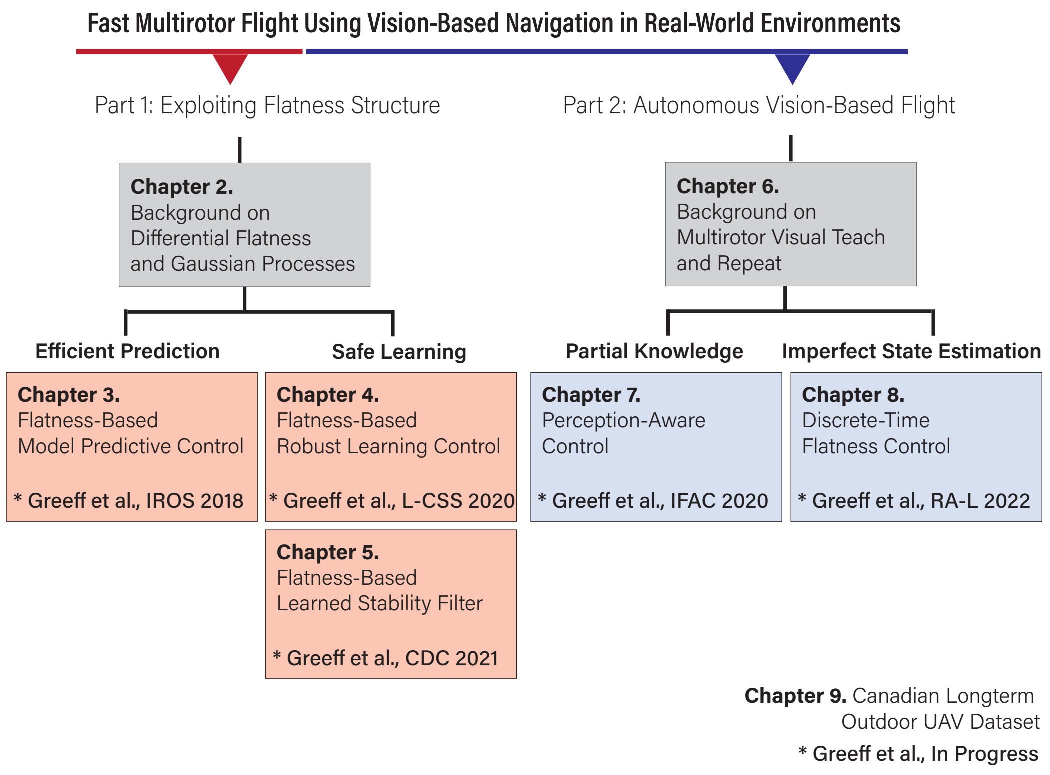

Fast Multirotor Flight Using Vision-Based Navigation in Real-World Environments
Part 1: Exploiting Flatness Structure
Part 2: Autonomous Vision-Based Flight
Chapter 2. Background on Differential Flatness and Gaussian Processes
Chapter 6. Background on Multirotor Visual Teach and Repeat
Efficient Prediction
Safe Learning
Partial Knowledge
Imperfect State Estimation
Chapter 3. Flatness-Based Model Predictive Control
* Greeff et al., IROS 2018
Chapter 4. Flatness-Based Robust Learning Control
* Greeff et al., L-CSS 2020
Chapter 7. Perception-Aware Control
* Greeff et al., IFAC 2020
Chapter 8. Discrete-Time Flatness Control
* Greeff et al., RA-L 2022
Chapter 5. Flatness-Based Learned Stability Filter
* Greeff et al., CDC 2021
Chapter 9. Canadian Longterm Outdoor UAV Dataset
* Greeff et al., In Progress

Figure 1.2: This diagram shows the structure and conceptual relationship between the chapters and contributions of this thesis. Items marked with a * refer to a peer-reviewed conference or journal paper.

## Contributions

Fig. 1.2 highlights the structure and conceptual relationship between the contents of this thesis. For more detail, we provide an overview of the chapters of this thesis below. The work in these chapters have resulted in the following publications:

- M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 6740-6745, 2018.
- M. Greeff and A. P. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using Gaussian Processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121-1126, 2020.
- M. Greeff, A. W. Hall, and A. P. Schoellig, "Learning a stability filter for uncertain differentially flat systems using Gaussian Processes," in *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 789-794, 2021.
- 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, pp. 9412-9419, 2020.
- M. Greeff, S. Zhou, and A. P. Schoellig, "Fly out the window: Exploiting discrete-time flatness for fast vision-based multirotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5023-5030, 2022.

Additional contributions include:

- 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.
- L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, 2021, *Accepted.*
- M. Greeff, B. Patel, M. Warren, M. Bianchi, J. Wong, J. Collier, T. D. Barfoot, and A. P. Schoellig, "Work in the CLOUD: Canadian Longterm Outdoor UAV Dataset," *In Progress.*

## Overview of Part I: Exploiting Flatness Structure

As illustrated in Fig. 1.2, Part I of this thesis contains four chapters.

**Chapter 2 – Background on Differential Flatness and Gaussian Processes:** In this chapter, we provide the formal mathematical definition of differential flatness and key theorems that are required for our work in Chapters 3 - 5. Many physical system models exhibit a structural property known as differential flatness. Intuitively, differential flatness allows us to separate the system's nonlinear dynamics into a linear dynamics component and a nonlinear term. A Gaussian Process (GP) is a nonparametric learning approach that uses data to provide both a most likely prediction and an uncertainty estimate. In this chapter, we provide background on GPs for our work in Chapters 4 - 5.

**Chapter 3 – Flatness-Based Model Predictive Control:** The use of model predictive control for multirotor applications requires balancing trajectory tracking performance and constraint satisfaction with fast computation. This chapter presents a *model-based* technique, Flatness-based Model Predictive Control (FMPC), that can be applied to multirotors, and more generally, differentially flat nonlinear systems. Our proposed FMPC couples feedback model predictive control with feedforward (FF) linearization. The proposed approach has the computational advantage that, similar to linear model predictive control, it only requires solving a convex quadratic program instead of a nonlinear program. However, unlike linear model predictive control, we still account for the nonlinearity in the model through the use of an inverse nonlinear term. In simulation, we demonstrate improved robustness over approaches that couple model predictive control with the more common feedback (FB) linearization. In experiments using multirotors, we also demonstrate improved trajectory tracking compared to classical linear and nonlinear model predictive control approaches. The work in this chapter is model-based and closed-loop performance is limited to the accuracy of the prior model and does not improve with online-data. We investigate learning-based approaches, that can improve performance using online system data, in Chapters 4 - 5.

**Chapter 4 – Flatness-Based Robust Learning Control:** Learning-based control has shown to outperform conventional model-based techniques in the presence of model uncertainties and systematic disturbances. However, most state-of-the-art learning-based nonlinear trajectory tracking controllers still lack any formal guarantees. In this chapter, we exploit the property of differential flatness to design an online, robust learning-based controller to achieve both high tracking performance and safety, through probabilistically guaranteeing a uniform ultimate bound on the tracking error. A common control approach for differentially flat systems is to try to linearize the system by using a feedback (FB) linearization controller designed based on a nominal system model. Performance and safety are limited by the mismatch between the nominal model and the actual system. Our proposed approach uses a nonparametric Gaussian process (GP) to both improve FB linearization and quantify, probabilistically, the uncertainty in our FB linearization. We use this probabilistic bound in a robust linear quadratic regulator (LQR) framework. Through simulation, we highlight that our proposed approach significantly outperforms alternative learning-based strategies that use differential flatness. Our method can safely improve performance with online data but is limited to a linear quadratic regulator (LQR) framework, and can not easily integrate with Flatness-based Model Predictive Control in Chapter 3. Moreover, the computation of the bound for the robust linear controller is a nonlinear program that is not efficient to compute at a rate of 50-100 Hz commonly used by multirotor position control.

**Chapter 5 – Flatness-Based Learned Stability Filter:** In this chapter, we learn a filter that can augment any controller for differentially flat systems to efficiently certify robot tracking stability and input constraints in the presence of model uncertainty. More specifically, we exploit the differential flatness structure and propose using a nonparametric Gaussian process (GP) to learn the unknown nonlinear term. We use this GP in an optimization problem to optimize for an input that is most likely to feedback (FB) or feedforward (FF) linearize the system (i.e., cancel the nonlinear term). This optimization is subject to input constraints and a stability filter, described by an uncertain Control Lyapunov Function (CLF), which probabilistically guarantees exponential trajectory tracking when possible. Furthermore, for systems that are control-affine, we choose to express this structure in the selection of the kernel for the GP. By exploiting this selection, we show that the optimization problem is not only convex but can be efficiently solved as a second-order cone program. We compare our approach to related works in simulation and show that we can achieve similar performance at much lower computational cost.

## Overview of Part II: Autonomous Vision-Based Flight

As illustrated in Fig. 1.2, Part II of this thesis contains four chapters.

**Chapter 6 – Background on Multirotor Visual Teach and Repeat:** This chapter provides background on our vision-based route-following system for the autonomous, safe return of UAVs. Redundant navigation systems are critical for safe operation of UAVs in high-risk environments. We provide detail on our DJI Matrice 600 hardware set-up, the VT&R software, preliminary field testing and critical challenges to high-speed autonomous vision-based flight that are addressed in Chapter 7 and Chapter 8.

**Chapter 7 – Perception-Aware Control:** Despite the push toward fast, reliable vision-based multirotor flight, most vision-based navigation systems still rely on controllers that are perception-agnostic. Given that these controllers ignore their effect on the system's localization capabilities, they can produce an action that allows vision-based localization (and consequently navigation) to fail. In this chapter, we present a perception-aware flatness-based model predictive controller (MPC) that accounts for its effect on visual localization in VT&R. To achieve perception awareness, we first develop a simple geometric model that uses over 12 km of flight data from two different environments (urban and rural) to associate visual landmarks with a probability of being successfully matched. In order to ensure localization, we integrate this model as a chance constraint in our MPC such that we are probabilistically guaranteed that the number of successfully matched visual landmarks exceeds a minimum threshold. We show how to simplify the chance constraint to a nonlinear, deterministic constraint on the position of the multirotor. With desired speeds of 10 m/s, we demonstrate in simulation (based on real-world perception data) how our proposed perception-aware MPC is able to achieve faster flight while guaranteeing localization compared to similar perception-agnostic controllers. We illustrate how our perception-aware MPC adapts the path constraint along the path based on the perception model by accounting for camera orientation, path error and location of the visual landmarks. The result is that repeating the same geometric path but with the camera facing in opposite directions can lead to different optimal paths.

**Chapter 8 – Discrete-Time Flatness Control:** In this chapter, 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. 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. 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.

**Chapter 9 – Canadian Longterm Outdoor UAV Dataset:** In this chapter, we present the Canadian Longterm Outdoor UAV Dataset (CLOUD) with over 30 km of visual-inertial flight data across three different locations across Canada. To our knowledge, this dataset is currently the largest UAV visual outdoor dataset. Our dataset is collected with our DJI Matrice 600 with a gimballed camera. We also include satellite images, rendered using Google Earth, for the various paths flown. Our dataset is well-suited for future research in UAV visual localization including testing robustness to perspective, lighting and seasonal changes. Other potential use cases include UAV navigation using satellite images, experience-based localization and simultaneous localization and mapping.

# Part I

# Exploiting Flatness Structure

# Chapter 2

# Background on Differential Flatness and Gaussian Processes

## 2.1 Differential Flatness

The property of differential flatness has been demonstrated for a variety of robotic dynamic models, including quadrotors, see [Mellinger et al., 2011], differential drive robots, see [G. Campion et al., 1996], car-like robots, see [R. M. Murray et al., 1996], omni-directional robots, see [S. Jiang et al., 2013], protocentric aerial manipulators, see [B. Yüksel et al., 2016] (where the first joint of the manipulator is attached at the center of mass of the aerial vehicle), etc. For this reason, differential flatness is an important concept for both fully-actuated and under-actuated robotics systems. However, some physical nonlinear models of systems are not differentially flat, for example, the "ball and beam", "double inverted pendulum", and certain chemical reactors, see Chapter 12 in [Sira-Ramirez et al., 2018] for further examples.

We recall the formal definition of differential flatness.

**Definition 1 (Differential Flatness)** *Consider a system with a continuous-time, nonlinear model of the form:*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x_0}, \tag{2.1}$$

*with $t \in \mathbb{R}^+$, $\mathbf{x}(t) \in X \subseteq \mathbb{R}^n$, $\mathbf{u}(t) \in U \subseteq \mathbb{R}^m$ and $f$ being a smooth function. This nonlinear system model (2.1) is differentially flat if there exists $\mathbf{y}(t) \in \mathbb{R}^m$, whose components are differentially independent (the components are not related to each other* *through a differential equation), such that the following holds [Fliess et al., 1995]:*

$$\mathbf{y} = \Lambda(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \dots, \mathbf{u}^{(\delta)}), \tag{2.2}$$

$$\mathbf{x} = \Phi(\mathbf{y}, \dot{\mathbf{y}}, \dots, \mathbf{y}^{(r-1)}), \tag{2.3}$$

$$\mathbf{u} = \Psi^{-1}(\mathbf{y}, \dot{\mathbf{y}}, \dots, \mathbf{y}^{(r)}), \tag{2.4}$$

*where $\Lambda$, $\Phi$ and $\Psi^{-1}$ are smooth functions, $\delta$ and $r$ are the maximum orders of the derivatives of $\mathbf{u}$ and $\mathbf{y}$ needed to describe the system and $\mathbf{y} = [y_1, \dots, y_m]^T$ is called the flat output.*

More intuitively, the system (2.1) 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 system (with the same number of inputs, i.e., $\mathbf{y}^{(r)} = \mathbf{v}$ where $\mathbf{v}(t) \in \mathbb{R}^m$ 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.

As explained in [Hagenmeyer et al., 2003 (a)], every *differentially flat* system (2.1) can be represented using a Brunovský state (or *flat state*):

$$\mathbf{z} := \left[y_1, \dot{y}_1, \dots, y_1^{(\rho_1 - 1)}, \dots, y_m, \dot{y}_m, \dots, y_m^{(\rho_m - 1)}\right]^T. \tag{2.5}$$

Note that $\rho_i$ is the maximum derivative of $y_i$ found in (2.4). Using the state transformation between the flat state $\mathbf{z}$ and state $\mathbf{x}$, obtained by differentiation of (2.2) and using (2.3) from Definition 1, we can transform (2.1) into the normal form:

$$y_i^{(\rho_i)} = \gamma_i(\mathbf{y}, \dot{\mathbf{y}}, \dots, \mathbf{y}^{(\rho-1)}, \mathbf{u}, \dot{\mathbf{u}}, \dots, \mathbf{u}^{(\sigma_i)}) := v_i, \tag{2.6}$$

where $\gamma_i$, $i = 1 \dots m$, is a smooth function obtained as a result of the transformation. Note $\sigma_i$ is the maximum derivative of $\mathbf{u}$ after $\rho_i$ times differentiating $y_i$ in (2.2). We define the *flat input* $\mathbf{v}$ as:

$$\mathbf{v} := \left[v_1, v_2, \dots, v_m\right]^T. \tag{2.7}$$

Using the definitions in (2.5) and (2.7), we rewrite (2.6) as:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v}, \tag{2.8a}$$

$$\mathbf{v} = \Psi(\mathbf{z}, \mathbf{u}, \dot{\mathbf{u}}, \dots, \mathbf{u}^{(\sigma)}), \tag{2.8b}$$

where $\sigma = \max \sigma_i$. We name (2.8a) the *linear flat model*. By substituting the definitions in (2.5) and (2.7), we can rewrite (2.4) as

$$\mathbf{u} = \Psi^{-1}(\mathbf{z}, \mathbf{v}). \tag{2.9}$$

The property of differential flatness (with position and yaw as output) is well-established for multirotors and has been exploited for efficient trajectory generation using (2.8a) in [Mellinger et al., 2011].

**Differential Flatness for SISO Control-Affine Models** In the following Definition 2 and Lemma 1, we consider a special case of differential flatness for single-input single-output (SISO) control-affine system models.

Remark: The assumption of control-affine is often true for robotic nonlinear dynamic models which are often affine in the commanded thrust and torque.

**Definition 2 (Differential Flatness SISO Models)** *Consider a single-input, single-output (SISO) system with state* $\mathbf{x} \in \mathbb{R}^n$, $t \in \mathbb{R}^+$ *and input* $u \in \mathbb{R}$*:*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), u(t)). \tag{2.10}$$

*A SISO nonlinear system (2.10) is differentially flat in output* $y(t) \in \mathbb{R}$ *if there exist smooth, invertible functions such that:* $\mathbf{x} = \phi(\mathbf{z})$, $u = \psi^{-1}(\mathbf{z}, v)$, *where* $\mathbf{z} = [y, \dot{y}, ..., y^{(n-1)}]^T$, $v = y^{(n)}$. *Furthermore, if (2.10) is control-affine, see (2.11) below:*

$$\dot{\mathbf{x}}(t) = f_1(\mathbf{x}(t)) + f_2(\mathbf{x}(t))u(t), \tag{2.11}$$

*then* $\psi^{-1}(\mathbf{z}, v)$ *is also control-affine, i.e., we can write* $\psi^{-1}(\mathbf{z}, v) = \alpha(\mathbf{z}) + \beta(\mathbf{z})v$ *[Fliess et al., 1995].*

**Lemma 1** *If system (2.10) is differentially flat in output* $y(t) \in \mathbb{R}$*, then it is equivalent to:*

$$v = \psi(\mathbf{z}, u), \tag{2.12}$$

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}v, \tag{2.13}$$

*where the linear dynamics (2.13) are an integrator chain of degree* $n$ *and the nonlinear term is given by (2.12). Moreover, if (2.10) is control-affine, see (2.11), the nonlinear term (2.12) is also control-affine, that is, (2.12) can be written as:*

$$v = \frac{u - \alpha(\mathbf{z})}{\beta(\mathbf{z})} = a(\mathbf{z}) + b(\mathbf{z})u, \tag{2.14}$$

*where* $a(\mathbf{z}) = -\frac{\alpha(\mathbf{z})}{\beta(\mathbf{z})}$ *and* $b(\mathbf{z}) = \frac{1}{\beta(\mathbf{z})}$.

## 2.2 Gaussian Processes (GPs)

GP regression is a nonparametric approach that is used to approximate a nonlinear map, $\psi(\mathbf{a}) : \mathbb{R}^{\dim(\mathbf{a})} \to \mathbb{R}$, from the input $\mathbf{a}$ to the function value $\psi(\mathbf{a})$. It does this by assuming that the function values $\psi(\mathbf{a})$, associated with different inputs $\mathbf{a}$, are random variables and that any finite number of these random variables have a joint Gaussian distribution. This nonparametric approach still requires us to define two priors: a prior mean function of $\psi(\mathbf{a})$, generally set to zero, and a covariance or kernel function $k(\cdot,\cdot)$ which encodes, for two input points, how similar their respective function values are. For example, a common kernel function is the squared-exponential (SE) function:

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \text{exp}\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{L}^{-2}(\mathbf{a}_i - \mathbf{a}_j)\right) + \delta_{ij}\sigma_\eta^2,$$

which is characterized by three types of hyperparameters: the prior variance $\sigma_f^2$, observation noise $\sigma_\eta^2$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, and the length scales, or the diagonal elements of the diagonal matrix $\mathbf{L}$, which encode a measure of how quickly the function $\psi(\mathbf{a})$ changes with respect to $\mathbf{a}$. These hyperparameters can be optimized by solving a maximum log-likelihood problem, see [Rasmussen et al., 2006].

This GP framework can be used to predict the function value at any query point $\mathbf{a}$ based on $N_d$ noisy observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{\psi}(\mathbf{a}_i)\}_{i=1}^{N_d}$. The predicted mean and variance at the query point $\mathbf{a}$ conditioned on the observed data $\mathcal{D}$ are, see [Rasmussen et al., 2006]:

$$\mu(\mathbf{a}) = \mathbf{k}(\mathbf{a})\mathbf{K}^{-1}\hat{\mathbf{\Psi}}, \tag{2.15}$$

$$\sigma^2(\mathbf{a}) = k(\mathbf{a}, \mathbf{a}) - \mathbf{k}(\mathbf{a})\mathbf{K}^{-1}\mathbf{k}^T(\mathbf{a}), \tag{2.16}$$

where $\hat{\mathbf{\Psi}} = [\hat{\psi}(\mathbf{a}_1), ..., \hat{\psi}(\mathbf{a}_{N_d})]^T$ is the vector of observed function values, the covariance matrix has entries $\mathbf{K}_{ij} = k(\mathbf{a}_i, \mathbf{a}_j), \quad i, j \in 1, ..., N,$ and $\mathbf{k}(\mathbf{a}) = [k(\mathbf{a}, \mathbf{a}_1), ..., k(\mathbf{a}, \mathbf{a}_{N_d})]$ is the vector of the covariances between the query point $\mathbf{a}$ and the observed data points in $\mathcal{D}$.

# Chapter 3

# Flatness-Based Model Predictive Control

**Exploiting Flatness Structure for Efficient Prediction**

## 3.1 Overview and Related Work

The growing interest in unmanned aerial vehicles (UAVs) for applications such as infrastructure inspection, see [Bircher et al., 2016], search-and-rescue missions, see [Rudol et al., 2008], and mapping operations, see [Han et al., 2013], has challenged researchers to develop controllers that can move beyond lab demonstrations to real-world scenarios. Successful controllers therefore must meet the following three criteria:

- exhibit high-trajectory tracking performance;
- explicitly account for input and state constraints;
- demonstrate robustness to unmodelled dynamics, disturbances and time delays.

Furthermore, the inherently fast dynamics of UAVs require *real-time operation, on-board, in a high-frequency feedback loop.*

### 3.1.1 Model Predictive Control

Model predictive control (MPC) is a popular approach to meet the first two criteria by optimizing over a prediction horizon while still explicitly adhering to constraints on the states and inputs of the system, see, for example, [Bangura et al., 2014] and

[Kamel et al., 2017]. However, the practical challenge is to balance this with the real-time computation requirement.

The most common real-time MPC approach is a *direct* method, which transforms the open-loop optimal control problem into a finite-dimensional problem by first discretizing the model, see [Diehl et al., 2009]. This model-based controller then generally considers one of three model classes.

Nonlinear Model Predictive Control (NMPC) uses a nonlinear system model which, coupled with *direct* methods, results in solving a non-convex nonlinear program (NLP) at each time step. Current real-time NMPC tends to find a suboptimal solution of the NLP by performing often only one iteration of a sequential quadratic program (SQP) with Gauss-Newton approximation, see [Wang et al., 2010] or [Houska et al., 2011]. This is often combined with *warm-starting*, i.e., initializing using the estimate from the previous time step, see [Diehl et al., 2009]. For longer time horizons, efficiency can be improved by using a *multiple shooting* method, which considers both the system state and input as optimization variables and adds the system dynamics as equality constraints. When solving the NLP, the SQP solver can then exploit the resulting sparse structure of the problem, see [Wang et al., 2010].

Alternatively, Linear Model Predictive Control (LMPC) uses a linearized model (often about hover for multirotors, see, for example, [Bangura et al., 2014]), which, coupled with *direct* methods, results in a convex quadratic program (QP) that can be efficiently solved at each time step.

### 3.1.2 Exploiting Differential Flatness in Model Predictive Control

The final approach, model predictive control combined with feedback (FB) linearization (shortened to MPC + FB linearization), combines feedback linearization to cancel nonlinear terms with MPC that considers a linear model, see [Primbs et al., 1997] or [Nevistic et al., 1997]. This idea was first presented in the mid 1990s. It was shown that using a representative, but modified cost function, can result in a convex QP as in LMPC. MPC in our proposed approach in Fig. 3.1 similarly solves such a convex QP. Initial work looked promising as simulations showed comparable performance to NMPC but with decreased computational cost, see [Primbs et al., 1997]. However, the practical implementation of MPC + FB linearization appears to be stunted by both robustness issues, see [Khotare et al., 1995], and the required input constraint conversion.

This chapter tackles the following question: *Can we make the idea of MPC + FB linearization, where a linearization term is coupled with linear MPC, practical and implementable?*

To answer this question, we first consider when and how feedback linearization can be applied. Many physical systems, including cranes, cars with trailers and multirotors, can be described by nonlinear models exhibiting a property known as differential flatness, see [Fliess et al., 1995] and [Mellinger et al., 2011]. Intuitively, differential flatness allows us to separate the nonlinear model into a linear dynamics component and a nonlinear transformation. This property can be utilized in both feedback and feedforward linearization, see [Hagenmeyer et al., 2003 (a)].

Our proposed approach founds itself on an intuitive idea presented in early work using MPC and feedback linearization (MPC + FB linearization). In [Primbs et al., 1997], the coupling of MPC and inner loop feedback linearization, $\mathbf{u} = \Psi^{-1}(\mathbf{z}, \mathbf{v}_d)$, is proposed. The idea is simply to cancel the nonlinear terms in (2.8b) and be left with linear prediction dynamics (2.8a), where $\mathbf{v} = \mathbf{v}_d$. The resulting MPC considers (2.8a) instead of the nonlinear model (3.1) with *flat* output $\mathbf{y}(t)$. Despite this computational advantage, there are a few factors that must be considered:

**Factor 1 – Development of a new cost:** The required convexity of the resulting optimal control problem may force us to develop a new cost function. Consider that even if the original cost function used for the nonlinear system is convex, the nonlinear transformation of the state $\mathbf{x}$ and input $\mathbf{u}$ into flat state $\mathbf{z}$ and flat input $\mathbf{v}$ can result in a nonlinear, non-convex cost for the new flat variables, see [Primbs et al., 1997].

**Factor 2 – Conversion of input constraints:** Another critical consideration is that due to the nonlinear parametrization in (2.4) convex constraints on the system input $\mathbf{u}$ may not map to convex constraints on the flat input $\mathbf{v}$. In general, there are two approaches in the literature to obtain linear input constraints on $\mathbf{v}$. The first approach calculates the exact input constraint on $\mathbf{v}$ at the current time step and then applies this as a constant constraint for the entire prediction horizon, see [Deng et al., 2009]. The second approach uses the previously predicted solution sequence for the flat state to construct a linear approximation of the constraints on $\mathbf{v}$, see [Margellos et al., 2010] or [Kurtz et al., 2010]. Our proposed FMPC also suffers from this limitation. In practice, we apply conservative box constraints on the multirotor flat states and flat inputs as is done in [Mueller et al., 2013] for trajectory generation. Determining less conservative constraints in a comparably efficient way is left for future work.

**Factor 3 – Robustness:** In [Khotare et al., 1995], the authors conclude that coupling linear MPC and feedback linearization relies on cancellation of nonlinear terms, which makes its robustness to noise, parameter uncertainty and disturbances difficult to quantify. They suggest trying to incorporate plant uncertainty into MPC + FB linearization. Another robustness issue that has not been addressed for MPC + FB linearization is how to account for known input time delays. Our proposed FMPC instead combines linear MPC with feedforward linearization. Furthermore, we extend their initial results beyond single-input single-output (SISO) simulations.

### 3.1.3 Contributions

Feedforward linearization aims to overcome the robustness issues of feedback linearization, which may be the result of parametric model uncertainty leading to inexact pole-zero cancellation, see [Hagenmeyer et al., 2003 (a)]. In [Hagenmeyer et al, 2004], feedforward linearization achieved improved tracking performance over feedback linearization for a ball-plate experiment. Given its robustness advantages, our proposed approach, shown in Fig. 3.1, couples feedforward linearization with MPC. The contributions of this chapter are three-fold.

- Firstly, we propose a novel *Flatness-based Model Predictive Control* (FMPC) architecture that couples feedback MPC with feedforward linearization. Practical advantages (in particular, an ability to account for known input delays and improved robustness to model parameter uncertainty) over MPC + FB linearization are demonstrated in simulation in Section 3.4.

- Secondly, we implement our FMPC architecture on a multirotor UAV, accounting for inner-loop dynamics and known input time delays.

- Finally, in Section 3.5.3 we demonstrate promising results for FMPC as an outer-loop controller of a multirotor UAV with improved trajectory tracking performance over NMPC and LMPC.

## 3.2 Problem Statement

Consider a system with a continuous-time, nonlinear model of the form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x_0}, \\ \mathbf{y}(t) &= h(\mathbf{x}(t)) \end{aligned} \tag{3.1}$$

with $t \in \mathbb{R}^+$, $\mathbf{x}(t) \in X \subseteq \mathbb{R}^n$, $\mathbf{u}(t) \in U \subseteq \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^m$, and $f, h$ being smooth functions.

Given a reference trajectory $\mathbf{y}_{\text{ref}}(t)$, determine an optimal control problem (OCP) for real-time MPC that can be used to compute an input $\mathbf{u}(t)$ such that high-performance tracking is achieved, i.e. $||\mathbf{y}(t) - \mathbf{y}_{\text{ref}}(t)||$ remains small.

To achieve this, we assume that (3.1) is *differentially flat* in output $\mathbf{y}(t)$, see Definition 1 in Section 2.1, and propose a *Flatness-based Model Predictive Control* (FMPC) that utilizes this property.

## 3.3 Methodology

Our proposed FMPC in Fig. 3.1 still requires careful consideration of *Factor 1* and *Factor 2* of MPC + FB linearization. However, it attempts to address some of the issues related to *Factor 3*. We select output $\mathbf{y}(t)$ to be a flat output. Similarly, the reference trajectory $\mathbf{y}_{\text{ref}}(t)$ is defined in the flat output space. The implementation steps of our proposed FMPC are:

### 3.3.1 Feedforward Linearization

The proposed coupling of *feedforward linearization* and MPC, as seen in Fig. 3.1, allows us to use the linear flat model (2.8a) in a feedback MPC.

We briefly highlight a key result in feedforward linearization that demonstrates how we can rewrite the nonlinear model (3.1) as an equivalent linear one.

**Theorem 1** *(obtained from [Hagenmeyer et al., 2003 (a)]) Consider a desired trajectory in the flat output $\mathbf{y}_d$, including a corresponding desired flat state $\mathbf{z}_d$ (obtained by substituting $\mathbf{y}_d$ for $\mathbf{y}$ in (2.5)) and desired flat input $\mathbf{v}_d$ (obtained by substituting $\mathbf{y}_d$ for $\mathbf{y}$ in (2.6) and (2.7)). Given $\mathbf{y}_d$, if we apply the nominal control,*

$$\mathbf{u} = \Psi^{-1}(\mathbf{z}_d, \mathbf{v}_d), \tag{3.2}$$

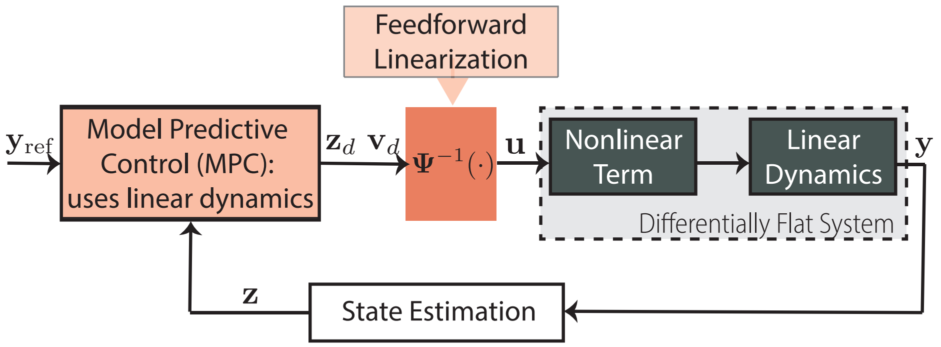

Feedforward Linearization
$\mathbf{y}_{\text{ref}}$
Model Predictive Control (MPC): uses linear dynamics
$\mathbf{z}_d$ $\mathbf{v}_d$
$\mathbf{\Psi}^{-1}(\cdot)$
$\mathbf{u}$
Nonlinear Term
Linear Dynamics
$\mathbf{y}$
Differentially Flat System
$\mathbf{z}$
State Estimation

Figure 3.1: Overall architecture diagram of the proposed Flatness-based Model Predictive Control (FMPC) approach.

*to a differentially flat system (3.1), provided that* $\mathbf{z}(0) = \mathbf{z}_d(0)$*, this results in an equivalent, by change of coordinates, linear system as given in (2.8a).*

Theorem 1 allows trajectory generators or controllers, as in our proposed approach in Fig. 3.1, to only consider the equivalent linear flat model. The output of the trajectory generator or controller, i.e., the desired flat state and flat input, can then be fed through the inverse transformation (3.2) to correct for the nonlinear part (2.8b) in the system. Feedforward linearization differs fundamentally from feedback linearization in that the desired flat state, see (3.2), as opposed to the measured flat state is used in the inverse term. Feedforward linearization has been shown to be more robust towards modelling errors than feedback linearization, see [Hagenmeyer et al, 2004]. Feedback linearization relies on cancelling the respective nonlinearities, which can result in robustness issues. A robustness analysis for feedforward linearization under parametric uncertainty was done in [Hagenmeyer et al., 2003 (b)].

In our approach, the MPC outputs $\mathbf{z}_d$ and $\mathbf{v}_d$, which are then fed through the inverse term (3.2). We take advantage of the robustness of feedforward linearization to parameter uncertainties, where unlike feedback linearization the inverse term does not try to explicitly cancel nonlinear terms. Further, unlike feedback linearization, we can only use feedforward linearization to reduce the nonlinear model (3.1) to an equivalent linear flat model (2.8a) because we satisfy the initial condition requirement in Theorem 1. We continuously ensure adherence to the initial condition requirement by feeding back our measured flat state $\mathbf{z}$ into the MPC where we re-optimize for our updated desired trajectory, $\mathbf{z}_d$ and $\mathbf{v}_d$.

*This means that feedback MPC and feedforward linearization have a symbiotic relationship: feedforward linearization allows us to use a simplified model in MPC, while using MPC as our feedback controller allows us to satisfy the conditions for* *feedforward linearization.*

### 3.3.2 Model Predictive Control

A standard *direct* method MPC strategy is considered. At each sampling time, we solve an open-loop OCP by minimizing a convex quadratic cost function $J(\cdot)$, which is dependent on the sequence of predicted flat states $\hat{\mathbf{z}}$ and flat inputs $\hat{\mathbf{v}}$. This is subject to both the discretized linear flat model of the system (2.8a) and linear constraints on the flat state and input, which approximate state $\mathbf{x}(t) \in X$ and input $\mathbf{u}(t) \in U$ constraints. The resulting OCP is a convex QP which can be efficiently solved for a global minimum.

**Time Delays:** Our proposed FMPC, which couples MPC and feedforward linearization, can easily be extended to systems with known input time delays. Consider the nonlinear system model (2.1) but now the input has a known time delay $t_d$, i.e. $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t - t_d))$. In our differential flatness definition, (2.4) now becomes $\mathbf{u}(t - t_d) = \Psi^{-1}(\mathbf{y}(t), \dot{\mathbf{y}}(t), \ldots, \mathbf{y}^{(\rho)}(t))$ or more compactly, using (2.5) and (2.7), $\mathbf{u}(t - t_d) = \Psi^{-1}(\mathbf{z}(t), \mathbf{v}(t))$. This gives an inverse term:

$$\mathbf{u}(t) = \Psi^{-1}(\mathbf{z}(t + t_d), \mathbf{v}(t + t_d)).$$

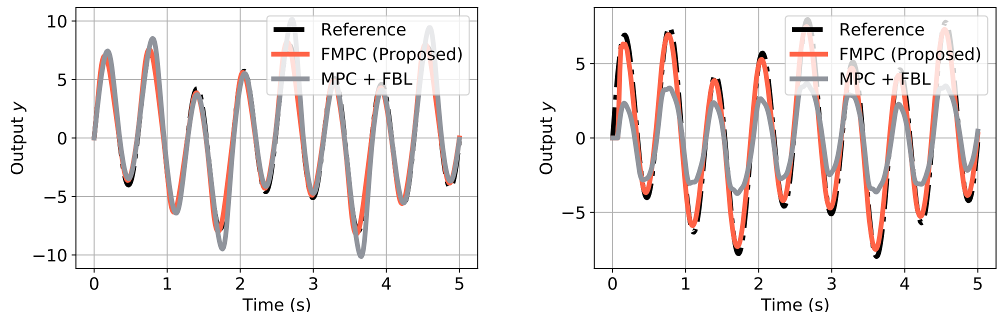
Notice that this relies on forward predicted states and so feedback linearization using the current flat state $\mathbf{z}(t)$ would not cancel the nonlinear terms. Our proposed FMPC instead feeds forward $\mathbf{z}_d(t + t_d)$ and $\mathbf{v}_d(t + t_d)$.

## 3.4 Simulations

We compare our proposed FMPC, coupling MPC with feedforward linearization, with MPC + FB linearization in simulation for a SISO system. We consider a nonlinear system with the following nominal SISO model (taken from [Hagenmeyer et al., 2003 (a)]):

$$\dot{x} = -x - x^3 + u, \tag{3.3}$$

where, utilizing its differential flatness property, we can define flat state and flat output $z = y = x$ and flat input $v = \dot{x}$. Equivalently, we rewrite the nonlinear model as a linear flat model $\dot{z} = v$ and a nonlinear term $v = -z - z^3 + u$. In the simulation,



(a) Model Parameter Sensitivity (b) Input Time Delay

Figure 3.2: Tracking of reference $y_{\text{ref}}(t) = 2\sin(3t) + 6\sin(10t)$ using FMPC and MPC + FB linearization for: (a) *Model Parameter Sensitivity*: with model parameter mismatch; (b) *Input Time Delay*: with known input time delay of 5 time steps.

we consider the following MPC formulation:

$$\min_{y_{1\ldots N}, v_{0\ldots N-1}} \frac{1}{2}\sum_{k=1}^{N} \tilde{Q}(y_k - y_{\text{ref},k})^2 + \frac{1}{2}\sum_{k=0}^{N-1} \tilde{R} v_k^2,$$

subject to the discretized linear flat model, $z_{k+1} = z_k + \delta t v_k$. We use a discretization of 70 Hz, prediction horizon $N = 70$ and weight matrices $\tilde{Q} = 100$ and $\tilde{R} = 0.1$. We consider a reference trajectory, in the flat output space, $y_{\text{ref}}(t) = 2\sin(3t) + 6\sin(10t)$. The difference between our proposed FMPC and MPC + FB linearization is that the desired state (output from MPC) $z_d$, instead of the current flat state $z$, is used in the inverse term, $\Psi^{-1}(\cdot)$, i.e., $u = z_d + z_d^3 + v_d$ in FMPC versus $u = z + z^3 + v_d$ in MPC + FB linearization. We compare results for three cases:

**Nominal Case:** We consider a nonlinear system with the same dynamics as our nominal model (3.3). FMPC and MPC + FB linearization exhibit comparable performance with root mean square (RMS) error of 0.5547 and 0.3854, respectively. In this case, MPC + FB linearization is slightly better as exact cancellation of the nonlinearity is possible.

**Model Parameter Sensitivity Case:** To test robustness to parametric uncertainty, we use model (3.3) but consider a nonlinear system with dynamics $\dot{x} = -0.9x - 0.9x^3 + u$ (taken from [Hagenmeyer et al., 2003 (a)]). FMPC, with RMS error 0.4826, has improved performance over MPC + FB linearization, with RMS error 1.0969. This is observed in Fig. 3.2(a) where inexact cancellation in MPC +

FB linearization (grey) results in the addition of unstable terms leading to poorer tracking performance.

**Input Time Delay Case:** We consider a nonlinear system with dynamics $\dot{x} = -x - x^3 + u(t - t_d)$ where $t_d$ is a known time delay. We simulate the case $t_d = 5/70$. To compensate for this delay, our FMPC feeds forward $z_d(t+t_d)$ and $v_d(t+t_d)$ (computed by the MPC) as the desired flat state and flat input. In MPC + FB linearization, we similarly attempt to compensate by sending $v_d(t+t_d)$. FMPC achieves an RMS error of 0.6956 while MPC + FB linearization achieves 2.3150. Fig. 3.2 shows the success of such time delay compensation in FMPC (red), while the same approach cannot be used for MPC + FB linearization (grey). Alternative approaches for time delay compensation in MPC + FB linearization may require an additional state predictor.

## 3.5 Experiments

### 3.5.1 Multirotor Application

We consider a cascaded control structure with a low-level onboard controller and an MPC outer-loop controller that can send commands $(\dot{z}_{\text{cmd}}, \phi_{\text{cmd}}, \theta_{\text{cmd}}, \dot{\psi}_{\text{cmd}})$, where $\dot{z}_{\text{cmd}}$ is commanded velocity in the $z$-direction, $\phi_{\text{cmd}}$ is the commanded roll angle, $\theta_{\text{cmd}}$ is the commanded pitch angle and $\dot{\psi}_{\text{cmd}}$ is the commanded yaw rate. We compare LMPC, NMPC, and FMPC.

To enhance trajectory tracking, we first perform a simple system identification, as in [Kamel et al., 2017], to approximate the inner-loop attitude dynamics by:

$$\dot{\phi} = \frac{1}{\tau}(\tilde{k}\phi_{\text{cmd}} - \phi), \tag{3.4a}$$

$$\dot{\theta} = \frac{1}{\tau}(\tilde{k}\theta_{\text{cmd}} - \theta), \tag{3.4b}$$

$$\dot{\psi} = \dot{\psi}_{\text{cmd}}, \tag{3.4c}$$

where $\tau$ is an identified time constant, $\tilde{k}$ is an identified gain and $\phi, \theta, \psi$ are the roll, pitch and yaw angles of the vehicle. Unlike in [Kamel et al., 2017], we do not directly send a thrust command $T_{\text{cmd}}$. Consequently, we perform a similar system identification to approximate the $z$-velocity dynamics by a second-order response with a time delay of $t_d = 0.1$ s:

$$\dddot{z}(t) = -\frac{1}{\tau_z}\dot{z}(t) - \frac{1}{\tau_{Iz}}\ddot{z}(t) + \frac{1}{\tau_{Cz}}\dot{z}_{\text{cmd}}(t - t_d), \tag{3.5}$$

where $\tau_z, \tau_{Iz}, \tau_{Cz}$ are identified time constants. Ignoring drag and other external forces, we can describe the lateral motion using the standard model, see [Alexis et al., 2015]:

$$\ddot{x} = \frac{\mathbf{R}_{13}}{\mathbf{R}_{33}}(\ddot{z} + g), \tag{3.6a}$$

$$\ddot{y} = \frac{\mathbf{R}_{23}}{\mathbf{R}_{33}}(\ddot{z} + g), \tag{3.6b}$$

where $x, y, z$ represent the linear position, $\mathbf{R}$ the rotation of the multirotor body frame with respect to an inertial frame and $g$ the gravitational constant. We use the notation $\mathbf{R}_{ij}$ to refer to the $(i, j)$ entry of $\mathbf{R}$.

**Nonlinear Model:** In our NMPC model formulation, we consider the nonlinear model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ described by (3.4a)-(3.4c), (3.5) and (3.6a)-(3.6b) with state and input:

$$\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{z}, \phi, \theta, \psi]^T,$$

$$\mathbf{u} = [\dot{z}_{\text{cmd}}, \phi_{\text{cmd}}, \theta_{\text{cmd}}, \dot{\psi}_{\text{cmd}}]^T.$$

**Linearized Model:** The only nonlinearity in our model formulation for NMPC comes from (3.6a)-(3.6b). In LMPC we consider the linearization of (3.6a)-(3.6b) about hover ($\phi = 0$, $\theta = 0$, $\ddot{z} = 0$) where at each time step we assume that our current yaw angle remains constant.

**Linear Flat Model:** The differential flatness of the standard multirotor model for flat outputs $\mathbf{y} = (x, y, z, \psi)$ is found in [Mellinger et al., 2011]. In a similar procedure, we can show the differential flatness, with the same flat outputs, of our nonlinear model governed by (3.4a)-(3.4c), (3.5) and (3.6a)-(3.6b). Our FMPC model formulation considers the linear flat model (2.8a) with flat state and flat input:

$$\mathbf{z} = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}, \psi]^T,$$

$$\mathbf{v} = [\dddot{x}, \dddot{y}, \dddot{z}, \dot{\psi}]^T.$$

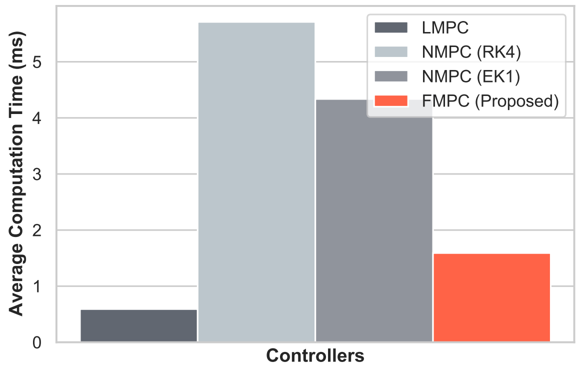


Figure 3.3: Comparison of RMS computation per time step. Our proposed FMPC required more computation than LMPC but less computation than NMPC (both NMPC E1 and NMPC RK4).

**Optimal Control Problem:** In the cost function in (3.7), $\mathbf{y}_k = [x_k, y_k, z_k, \psi_k]^T$ is our flat output at time step $k$, the positive semi-definite matrix $\tilde{\mathbf{Q}} \succeq 0$ weights the error with our reference trajectory and the positive-definite $\tilde{\mathbf{R}} \succ 0$ regulates both the size and change in inputs $\mathbf{u} = [\dot{z}_{\text{cmd}}, \phi_{\text{cmd}}, \theta_{\text{cmd}}, \dot{\psi}_{\text{cmd}}]^T$. In the LMPC OCP in (3.7), we optimize for $\mathbf{u}_k$ subject to the discretized *Linearized Model*. In NMPC, we similarly optimize for $\mathbf{u}_k$ using the cost in (3.7) but instead subject to the discretized *Nonlinear Model*. In LMPC, we use a *direct* method to set up an OCP that is repeatedly solved at each time step:

$$\begin{aligned} \min_{\mathbf{u}_{0\ldots N-1}} \; & \frac{1}{2}\sum_{k=1}^{N}(\mathbf{y}_k - \mathbf{y}_{\text{ref},k})^T\tilde{\mathbf{Q}}(\mathbf{y}_k - \mathbf{y}_{\text{ref},k}) + \frac{1}{2}\sum_{k=0}^{N-1}\mathbf{u}_k^T\tilde{\mathbf{R}}\mathbf{u}_k \\ \text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \; \mathbf{y}_k = \mathbf{C}\mathbf{x}_k, \\ & \mathbf{x}_k \in \Omega_x. \end{aligned} \tag{3.7}$$

For both LMPC and NMPC, we consider the constraint set in (3.7) to be:

$$\Omega_x = \{\mathbf{x} \in \mathbb{R}^{10} \mid |\ddot{z}| < 0.5; |\theta| \leq 0.4; |\phi| \leq 0.4; \psi \in [0, \pi]\}.$$

In the FMPC OCP, our cost is subject to the *Linear Flat Model*. To apply FMPC with a similar quadratic cost that is convex in $\mathbf{v}_k$ (discrete flat inputs) we use the *Linearized Model* to obtain a linear relationship between input $\mathbf{u}$ and our flat state $\mathbf{z}$ and flat input $\mathbf{v}$: $\mathbf{u} = \mathbf{M}\mathbf{z} + \mathbf{N}\mathbf{v}$. We then use this linear relationship to obtain a similar representative convex cost function for FMPC by replacing $\mathbf{u}$ with its linear

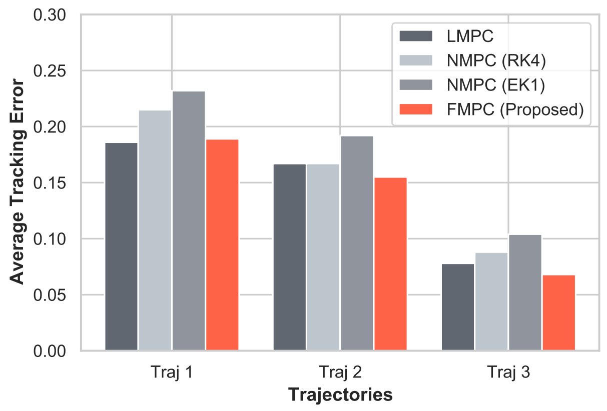


Figure 3.4: Comparison of RMS error for *Trajectory 1*, *Trajectory 2* and *Trajectory 3* (averaged over three trials per trajectory). For more aggressive trajectories (*Trajectory 2* and *Trajectory 3*) FMPC outperforms LMPC.

relationship in $\mathbf{z}$ and $\mathbf{v}$. Consequently, in FMPC we solve the following OCP at each time step:

$$\begin{aligned} \min_{\mathbf{v}_{0\ldots N-1}} \; & \frac{1}{2}\sum_{k=1}^{N}(\mathbf{y}_k - \mathbf{y}_{\text{ref},k})^T\tilde{\mathbf{Q}}(\mathbf{y}_k - \mathbf{y}_{\text{ref},k}) + \frac{1}{2}\sum_{k=0}^{N-1}\mathbf{u}_k^T\tilde{\mathbf{R}}\mathbf{u}_k \\ \text{subject to} \quad & \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{v}_k, \;\; \mathbf{y}_k = \mathbf{C}\mathbf{z}_k, \\ & \mathbf{u}_k = \mathbf{M}\mathbf{z}_k + \mathbf{N}\mathbf{v}_k, \\ & \mathbf{z}_k \in \Omega_z, \end{aligned} \tag{3.8}$$

where, similar to [Mueller et al., 2013], we approximate the state constraint set $\Omega_x$ with the constraint set:

$$\Omega_z = \{\mathbf{z} \in \mathbb{R}^{10} \mid |\ddot{z}| < 0.5; |\ddot{x}| \leq 7; |\ddot{y}| \leq 7; \psi \in [0,\pi]\}.$$

**Time Delay Compensation:** To compensate for the time delay in the $z$-direction in (3.5), both LMPC and NMPC output $\dot{z}_{\text{cmd}}(t+t_d)$. As described in Section 3.3, FMPC feeds forward $z_d(t+t_d), \dot{z}_d(t+t_d), \ddot{z}_d(t+t_d), \dddot{z}_d(t+t_d)$ through the inverse term (3.2).

## 3.5.2 Experimental Setup

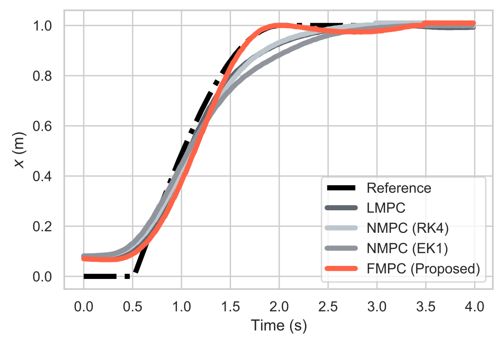
The experiments are conducted on a Parrot AR.Drone multirotor with an overhead motion capture system estimating the state of the multirotor. We interface with the multirotor using the open-source Robot Operating System (ROS), see [ROS]. Outer-loop MPC approaches are run off-board on a ThinkPad P50 with Intel Core i7-6700HQ Processor. We use the formulations described in Section 3.5.1, to compare four different off-board outer-loop model predictive controllers running at 70 Hz, namely: nonlinear model predictive control using a first-order Euler discretization for forward simulation (NMPC E1); nonlinear model predictive control using a fourth-order explicit Runge-Kutta discretization for forward simulation (NMPC RK4); linear model predictive control (LMPC); and our proposed flatness-based model predictive control (FMPC). All controllers consider a prediction at 10 Hz and a look-ahead time of $1\,\mathrm{s}$, where the prediction horizon is $N = 10$ in (3.7) and (3.8). For NMPC, a single iteration of an SQP, with Gauss-Newton Hessian approximation, is performed at each iteration. It is initialized using *warm-starting*. Furthermore, all optimization problems are solved using a *single-shooting* method whereupon the resulting QP is solved using CVXOPT in Python.

For each controller we perform three trials of three different trajectories: *Trajectory 1:* The multirotor follows a circular reference with radius 1 m and angular frequency $0.4\pi$ rad/s in the $x$-$y$ plane. There is no yawing and the vehicle remains at a constant altitude. *Trajectory 2:* The multirotor performs a $2\,\mathrm{s}$ step in $x$ and $z$. There is no motion in the $y$-direction and no yawing. *Trajectory 3:* The multirotor follows the circular reference from trajectory 1, but now also simultaneously yaws to $\pi/2$ while performing a step in $z$. We do this for a parameter estimation $(\tau = 0.25, \tilde{k} = 1.4)$ of the time constants and gains in the inner-loop dynamics model in (3.4a)-(3.4b).

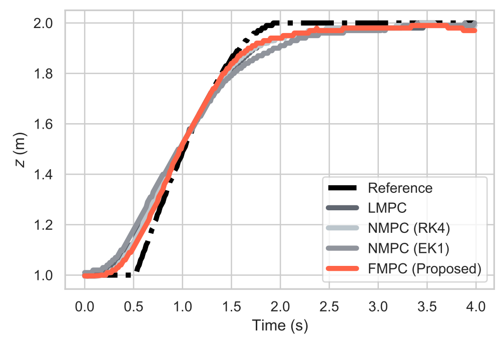
## 3.5.3 Results

While all model predictive controllers solve one QP at each time step, as seen in Fig. 3.3, our proposed FMPC (red) has an average computation between that for LMPC (dark grey) and NMPC (light and medium grey). The additional computation used in our proposed FMPC over LMPC leads to reduced RMS error for *Trajectory 2* and *Trajectory 3* in Fig. 3.4. This reduced error is attributed to FMPC accounting for the nonlinear effects of yawing and vertical acceleration on lateral motion in (3.6a)-(3.6b). In *Trajectory 1*, this effect is negligible since the trajectory requires no yawing or vertical motion. Comparable performance is observed for LMPC and our proposed FMPC.

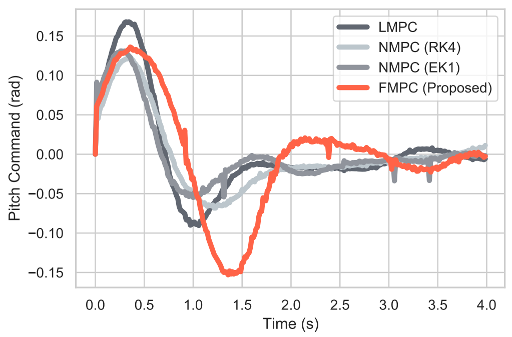
As seen in Fig. 3.5, FMPC (red) accounts for the nonlinear effect of vertical accel-



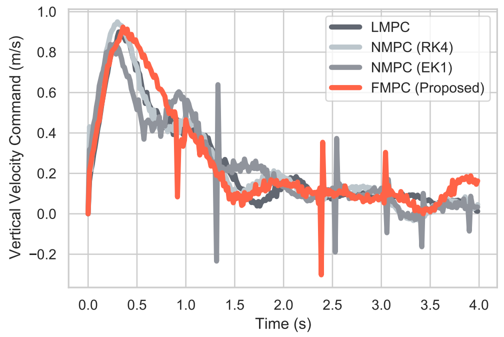
(a) Tracking in x



(b) Tracking in z



(c) Pitch command



(d) z-velocity command

Figure 3.5: Comparison of tracking *Trajectory 2*. Improved performance in $x$-direction tracking by FMPC is a result of a modified pitch command. Unlike LMPC, FMPC considers the effect of $z$-acceleration on lateral tracking. FMPC determines a larger pitch command for a longer period of time in the first 1 s allowing increased $x$-acceleration before making a more substantial negative change than the other controllers.

eration on lateral motion in *Trajectory 2* by modifying the pitch command such that greater lateral acceleration is achieved in the first 1 s before allowing for a more dramatic change in pitch command to slow the vehicle down as it reaches $x = 1\,\text{m}$. This tends to provide a tracking improvement of 5-15% over LMPC (dark grey). NMPC RK4 (light grey) achieves similar performance to LMPC potentially suggesting that for *Trajectory 2* linearization along the simulated trajectory (as is done for NMPC) provides little overall performance advantage over linearization about hover (as is done for LMPC). The spikes in the $\dot{z}_{\text{cmd}}$ in (b) can be attributed to the approximation of the $z$-acceleration. The prevalence of these spikes is not consistent among trials. We believe this can be prevented with better filtering.

## 3.6 Summary

Results show that our proposed Flatness-based Model Predictive Control (FMPC) applied to multirotor trajectory tracking is promising. Its advantages include:

- unlike LMPC, it is able to account for nonlinearities while still solving a convex QP;
- unlike NMPC, it is not sensitive to initial trajectory choice or susceptible to converging to local minima;
- using feedforward linearization instead of feedback linearization improves robustness to modelling errors and can account for known input time delays.

The approach in this chapter has three novel contributions:

- Firstly, we propose a novel Flatness-based Model Predictive Control (FMPC) architecture that couples feedback MPC with feedforward linearization.
- Secondly, we implement our FMPC architecture on a multirotor UAV, accounting for inner-loop dynamics and known input time delays.
- Finally, in Section 3.5.3 we demonstrate promising results for FMPC as an outer-loop controller of a multirotor UAV with improved trajectory tracking performance over NMPC and LMPC.

The *key insight* in this chapter, is that feedforward linearization allows us to use a simplified model in MPC, while using MPC as our feedback controller allows us to satisfy the initial conditions for feedforward linearization.

# Chapter 4

# Flatness-Based Robust Learning Control

**Exploiting Flatness Structure for Safe Learning**

## 4.1 Overview and Related Work

The heightened interest in using learning-based control to achieve high-accuracy tracking has, in part, been driven by advanced robotic applications where accurate models are required but difficult to derive. In practice, many current learning-based control techniques achieve good tracking performance by learning a fairly accurate nonlinear dynamics model. However, the adoption of these techniques has been limited to applications that are not safety-critical as these techniques fail to provide any rigorous analysis of safety such as constraint satisfaction or stability and convergence. The need to provide guarantees while using a data-driven learned model is still a key challenge for robotics.

### 4.1.1 Learning-Based Control

The limitation of many learning-based approaches, for example, standard neural networks (NNs), is that they are ill-suited to quantify any mismatch between the learned model and the real dynamics. For this reason, nonparametric approaches, such as Gaussian processes (GPs), have gained popularity within the control community as they can provide uncertainty estimates for their predictions, see [Berkenkamp et al., 2016] or [Umlauft et al., 2018]. The question then is: *how can we efficiently use this uncertainty measure in the control loop?* This depends on the assumed dynamic structure of the actual system and the selected part of the dynamics to be learned.

GPs can be used to learn the forward nonlinear system dynamics. For example, one common approach is to use this learned GP model in a nonlinear model predictive control framework where the uncertainty estimate from the GP is used to tighten constraints, see [Ostafew et al., 2016]. However, this approach provides no stability analysis of the controlled system. Another approach linearizes the learned nonlinear model about an operating point, which, combined with the uncertainty estimate from the GP is used in a linear robust control framework, see [Berkenkamp et al., 2015]. However, this robust learning-based controller is limited to stabilization tasks.

In this chapter, we impose two structural properties on the system dynamics. Firstly, we assume the system is control-affine. Secondly, we assume the system dynamics exhibit differential flatness [Fliess et al., 1995]. This property is commonly used in feedback (FB) linearization controllers which attempt to cancel the nonlinear term such that outer-loop linear controllers, for example, linear quadratic regulators (LQR), can be designed based on the linear dynamics.

### 4.1.2 Exploiting Differential Flatness in Learning-Based Control

Related work on FB linearization has used learning-based strategies in one of two ways.

**Strategy 1:** The first strategy is to simply update a nominal FB linearization controller with a data-driven learned model. In [Westenbroek et al., 2019], reinforcement learning was used for FB linearization by learning the inverse model of the nonlinear term of the differentially flat system. In [Umlauft et al., 2017], a GP was used to learn the forward model of the nonlinear term, such that, by transferring knowledge of the control-affine structure into the kernel function, it could be used to update a FB linearization controller. Approaches in this category have not provided guarantees of stability or tracking convergence as they do not quantify how well the learned FB linearization controller cancels the nonlinear term and, consequently, how well it linearizes the system.

**Strategy 2:** The second strategy is to use a data-driven model to quantify how well a nominal FB linearization controller linearizes the system. We call the model error between the nominal FB linearization controller and nonlinear term the *Nonlinear Mismatch*. In [Helwa et al., 2019], a GP is used to learn a forward model of the *Nonlinear Mismatch* for Lagrangian systems. The learned prediction and uncertainty model is used to generate a bound on how well the nominal FB linearization controller linearizes the system. The bound is then used in a linear, robust outer-loop controller. While this strategy provides tracking guarantees, it is conservative as it does not update or improve the FB linearization.

### 4.1.3 Contributions

Our proposed approach combines ideas from both strategies. We use a GP to learn an inverse model of the *Nonlinear Mismatch*. As demonstrated in Fig. 4.1, we use our learned model to update the *Inverse Nonlinear Mismatch* which attempts to cancel the *Nonlinear Mismatch*. Considering the key idea from *Strategy 2*, we quantify how well we linearize the system. To do this we generate a probabilistic upper bound on the difference between the designed desired input and the actual input seen by the linear system dynamics. Finding this bound requires exploiting the control-affine flatness structure and several key properties of GPs. We use this bound in an additional robust term which, coupled with a nominal LQR, probabilistically guarantees stability, or more specifically, an ultimate bound on the tracking error. This chapter has three key contributions:

- We present a novel approach that uses a GP to both improve the FB linearization and quantify how well we are able to linearize the system.
- We demonstrate how our quantified uncertainty can be combined with a standard robust LQR to probabilistically guarantee an ultimate bound on the tracking error.
- We show through simulations how our proposed approach results in improved tracking performance over both *Strategy 1* (only improving the FB linearization) and *Strategy 2* (only quantifying how well a nominal controller linearizes the system).

To the best of our knowledge, this is the first approach that achieves robust, online learning-based control with formal guarantees on the tracking error for generic control-affine differentially flat systems.

## 4.2 Problem Statement

Consider a single-input single-output (SISO), control-affine system, with state $\mathbf{x} \in \mathbb{R}^n$, and input $u \in \mathbb{R}$:

$$\dot{\mathbf{x}} = f_1(\mathbf{x}) + f_2(\mathbf{x})u, \tag{4.1}$$

where $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are *unknown* functions.

**Assumption 1** *The system (4.1) is differentially flat in the known output $y = h(\mathbf{x}) \in \mathbb{R}$.*

**Assumption 2** *We have a SISO, control-affine nominal system model:*

$$\dot{\mathbf{x}} = \hat{f}_1(\mathbf{x}) + \hat{f}_2(\mathbf{x})u \tag{4.2}$$

*that is also differentially flat in the output $y = h(\mathbf{x}) \in \mathbb{R}$ where $\hat{f}_1(\mathbf{x})$ and $\hat{f}_2(\mathbf{x})$ are known functions.*

Under Assumptions 1-2, our goal is to design a control law $u$ such that:

- we achieve high-accuracy output tracking;
- we guarantee that the overall, closed-loop system satisfies robust stability (in the sense that the tracking error is probabilistically bounded despite model uncertainties).

To address this problem, we propose a learning-based control law that, by exploiting the differential flatness structure, both updates an inner-loop feedback (FB) linearization controller (red box with *Inverse Nonlinear Mismatch* in Fig. 4.1) and a robust linear controller. Our approach uses a GP as it allows us to quantify uncertainty. We present the proposed approach for the above SISO problem, however, a similar methodology could be applied to the multi-input, multi-output (MIMO) problem.

## 4.3 Methodology

We exploit the differential flatness of both our system (4.1) and our nominal system model (4.2), and apply Lemma 1 from Section 2.1. Following (2.13)-(2.14), our nominal system model is equivalent to (see Fig. 4.1):

$$v_{\text{cmd}} = \frac{u - \hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})}, \tag{4.3}$$

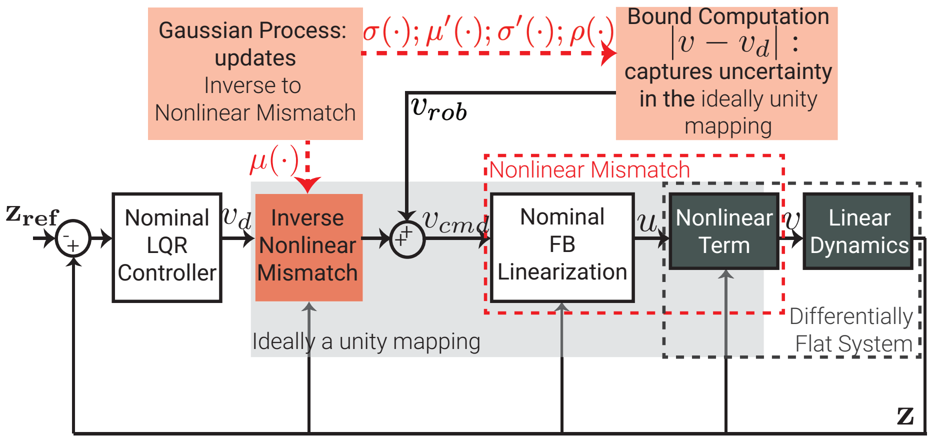

Gaussian Process: updates
Inverse to Nonlinear Mismatch
$\sigma(\cdot); \mu'(\cdot); \sigma'(\cdot); \rho(\cdot)$
Bound Computation
$|v - v_d|$ :
captures uncertainty in the ideally unity mapping
$v_{rob}$
$\mu(\cdot)$
Nonlinear Mismatch
$\mathbf{z}_{\text{ref}}$
Nominal LQR Controller
$v_d$
Inverse Nonlinear Mismatch
$v_{cmd}$
Nominal FB Linearization
$u$
Nonlinear Term
$v$
Linear Dynamics
Ideally a unity mapping
Differentially Flat System
$\mathbf{z}$

Figure 4.1: Our proposed architecture exploiting differential flatness for robust learning-based tracking control has three key components: 1) *Updating the Inverse Nonlinear Mismatch*: a GP learns the model error resulting from using a nominal feedback (FB) linearization; 2) *Bound Computation*: the properties of GPs are exploited to estimate a probabilistic bound on how well we linearize the system; 3) *Robust LQR*: this bound is combined with a nominal LQR to guarantee (probabilistically) an ultimate bound on the tracking error.

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}v_{\text{cmd}}. \tag{4.4}$$

We exploit this structure of our nominal system model to design a nominal FB linearization controller:

$$u = \hat{\alpha}(\mathbf{z}) + \hat{\beta}(\mathbf{z})v_{\text{cmd}}, \tag{4.5}$$

where the commanded input $v_{\text{cmd}}$ can be designed based on the linear model (4.4). Exploiting Lemma 1 and differential flatness leads to both our system (4.1) and nominal model (4.2) having an identical linear dynamics component. However, their nonlinear terms, see Fig. 4.1, will differ, i.e., the nonlinear term for our system (4.1) is given by:

$$v = \frac{u - \alpha(\mathbf{z})}{\beta(\mathbf{z})}. \tag{4.6}$$

Since functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are unknown for our system (4.1), the functions $\alpha(\mathbf{z})$ and $\beta(\mathbf{z})$ in our nonlinear term (4.6) are also unknown.

We exploit this structure in the learning controller design. As seen in Fig. 4.1, we term the mapping from commanded input $v_{\text{cmd}}$ to the actual input $v$ (seen by the linear system) the *Nonlinear Mismatch*. If our nominal model was identical to our system, i.e., $\alpha(\cdot) = \hat{\alpha}(\cdot)$ and $\beta(\cdot) = \hat{\beta}(\cdot)$, this would be a unity mapping. However, given the mismatch between the nominal model and the actual system, this will not be the case. In our proposed scheme we attempt to correct for this *Nonlinear Mismatch* in two phases:

**Learning Phase:** We propose learning the *Inverse Nonlinear Mismatch*, that is, the mapping from actual input $v$ to commanded input $v_{\text{cmd}}$.

**Running Phase:** We use the learned *Inverse Nonlinear Mismatch* to correct the nominal FB linearization controller as shown in Fig. 4.1. The learned *Inverse Nonlinear Mismatch* takes in a desired input $v_d$ and outputs a commanded input $v_{\text{cmd}}$. In a similar vein to FB linearization, if our *Inverse Nonlinear Mismatch* exactly cancels our *Nonlinear Mismatch*, then $v = v_d$. Of course, in practice our *Inverse Nonlinear Mismatch* will not exactly correct for our *Nonlinear Mismatch*, however, in Section 4.3.2 we use the GP uncertainty to estimate a probabilistic bound on the quality of this correction. In Section 4.3.3, we show how this bound can be used in a robust LQR controller to guarantee (probabilistically) an ultimate bound on the trajectory tracking error (see Section 4.4).

### 4.3.1 Update to Inverse Nonlinear Mismatch

**Learning Phase:** To find the *Inverse Nonlinear Mismatch* we write the commanded input $v_{\text{cmd}}$ in terms of state $\mathbf{z}$ and input $v$ by plugging (4.5) into (4.6), $v_{\text{cmd}} = \frac{\alpha(\mathbf{z})-\hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})} + \frac{\beta(\mathbf{z})}{\hat{\beta}(\mathbf{z})}v$, or equivalently $v_{\text{cmd}} = v + v_e(\mathbf{z}, v)$ where the difference $v_{\text{cmd}} - v$ is given by the function:

$$v_e(\mathbf{z}, v) = \frac{\alpha(\mathbf{z}) - \hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})} + \frac{\beta(\mathbf{z}) - \hat{\beta}(\mathbf{z})}{\hat{\beta}(\mathbf{z})}v. \tag{4.7}$$

If our model is identical to the system, then the *Inverse Nonlinear Mismatch* is unity, i.e. $v_{\text{cmd}} = v$. We propose using a GP framework to approximate (4.7) by considering a set of $N_d$ past observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{v}_e(\mathbf{a}_i)\}_{i=1}^{N_d}$ where inputs to the model are given by $\mathbf{a}_i = \{\mathbf{z}_i, v_i\}$, and we assume we have noisy measurements of the true function $\hat{v}_e(\mathbf{a}_i) = (v_{\text{cmd}} - v) + \eta$ with $\eta = \mathcal{N}(0, \sigma_\eta^2)$.

This GP framework can be used to predict the function value at any query point $\mathbf{a}^*$ based on $N_d$ noisy observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{v}_e(\mathbf{a}_i)\}_{i=1}^{N_d}$. It does this by using the key underlying principle that the observed data, or function values, and the function value at a query point $v_e(\mathbf{a}^*)$ are all jointly normal:

$$\begin{bmatrix} \hat{\mathbf{v}}_{\mathbf{e}} \\ v_e(\mathbf{a}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}^T(\mathbf{a}^*) \\ \mathbf{k}(\mathbf{a}^*) & k(\mathbf{a}^*, \mathbf{a}^*) \end{bmatrix}\right),$$

where $\hat{\mathbf{v}}_{\mathbf{e}} = [\hat{v}_e(\mathbf{a}_1), ..., \hat{v}_e(\mathbf{a}_{N_d})]^T$ is the vector of observed function values, the covariance matrix has entries $\mathbf{K}_{ij} = k(\mathbf{a}_i, \mathbf{a}_j)$, $i, j \in 1, ..., N$, and $\mathbf{k}(\mathbf{a}^*) = [k(\mathbf{a}^*, \mathbf{a}_1), ..., k(\mathbf{a}^*, \mathbf{a}_{N_d})]$ is the row vector of the covariances between the query point $\mathbf{a}^*$ and the observed data points in $\mathcal{D}$. By the conditioning property of Gaussian distributions, the mean and variance at our query point $\mathbf{a}^*$ conditioned on the observed data $\mathcal{D}$ are given by:

$$\mu(\mathbf{a}^*) = \mathbf{k}(\mathbf{a}^*)\mathbf{K}^{-1}\hat{\mathbf{v}}_{\mathbf{e}}, \tag{4.8}$$

$$\sigma^2(\mathbf{a}^*) = k(\mathbf{a}^*, \mathbf{a}^*) - \mathbf{k}(\mathbf{a}^*)\mathbf{K}^{-1}\mathbf{k}^T(\mathbf{a}^*). \tag{4.9}$$

**Running Phase:** During the run phase, we consider a query input $\mathbf{a}^* = \{\mathbf{z}, v_d\}$ where $v_d$ is some desired input computed by an outer-loop linear controller. By using the properties of joint Gaussian distributions we predict $v_e(\mathbf{a}^*)$ conditioned on the data set $\mathcal{D}$ by: $v_e(\mathbf{a}^*)|\mathcal{D} = \mathcal{N}(\mu(\mathbf{a}^*), \sigma^2(\mathbf{a}^*))$ where $\mu(\mathbf{a}^*)$ and $\sigma^2(\mathbf{a}^*)$ are obtained using (4.8) and (4.9), respectively.

We update the *Inverse Nonlinear Mismatch* using the mean from our prediction, i.e.,

$$v_{\text{cmd}} = v_d + \mu(\mathbf{a}^*) + v_{\text{rob}}, \tag{4.10}$$

where $v_{\text{rob}}$ is an additional robustness input as described in Section 4.3.3. To find the input $u$, we feed (4.10) through the nominal FB linearization controller (4.5). As seen in Fig. 4.1, ideally we learn an updated $v_{\text{cmd}}$ (4.10) to achieve a unity mapping, i.e., $v = v_d$. However, in practice there is a model uncertainty in this *ideally unity mapping* that we need to quantify in order to provide tracking guarantees. To do this, we can find the actual input $v$ (sent to the system), from (4.5), (4.6) and (4.10): $v = v_d + \left(\frac{\hat{\alpha}(\mathbf{z})-\alpha(\mathbf{z})}{\beta(\mathbf{z})} + \frac{\hat{\beta}(\mathbf{z})-\beta(\mathbf{z})}{\beta(\mathbf{z})}v_d\right) + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) + v_{\text{rob}})$. By noticing that $\frac{\hat{\alpha}(\mathbf{z})-\alpha(\mathbf{z})}{\beta(\mathbf{z})} + \frac{\hat{\beta}(\mathbf{z})-\beta(\mathbf{z})}{\beta(\mathbf{z})}v_d = -\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_e(\mathbf{a}^*)$, this reduces to:

$$v = v_d + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right) + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{\text{rob}}. \tag{4.11}$$

It is clear that if our mean $\mu(\mathbf{a}^*)$ perfectly characterizes $v_e(\mathbf{a}^*)$, we have exactly learned the *Inverse Nonlinear Mismatch* and $v = v_d$ without a robustness term $v_{\text{rob}}$. However, in practice this is not the case. We, therefore, require an estimate of a bound on $\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))$ such that we can use it in a robust control framework to design $v_{\text{rob}}$ such that we can probabilistically guarantee robust stability by establishing an ultimate bound on the tracking error.

### 4.3.2 Bound Computation

We propose finding a probabilistic bound $c$ such that:

$$\Pr\left\{\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*)-v_e(\mathbf{a}^*)\right)\right| < \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\right\} \geq 1-\delta, \tag{4.12}$$

where $\delta \in (0,1)$ is some user-selected small value. We can rewrite (4.12), arbitrarily dividing the probability $1-\delta$, by introducing an intermediate bound $\hat{c}$ such that:

$$\Pr\left\{\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*)-v_e(\mathbf{a}^*)\right)\right| < \hat{c}\right\} \geq \sqrt{1-\delta}, \tag{4.13}$$

and

$$\Pr\left\{\hat{c} < \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\right\} \geq \sqrt{1-\delta}. \tag{4.14}$$

Consequently, to compute bound $c$ we first find the intermediate bound $\hat{c}$ in (4.13). To compute such a bound, we exploit the derivative properties of GPs. From (4.3), we see that $\hat{\beta}(\mathbf{z}) = \frac{\partial u}{\partial v_{\text{cmd}}}$, and from (4.6), $\beta(\mathbf{z}) = \frac{\partial u}{\partial v}$. Consequently, the ratio is given by the partial derivative relationship $\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})} = \frac{\partial v}{\partial v_{\text{cmd}}}$. Using $v_{\text{cmd}} = v + v_e(\mathbf{z}, v)$, this ratio becomes:

$$\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})} = \frac{1}{1+\frac{\partial v_e(\mathbf{z},v)}{\partial v}}.$$

We utilize a key characteristic of the GP framework, which is that the derivative of a GP is a GP as well. By using the fact that the derivative is a linear operator, it can be shown that the derivative of a GP with respect to $v$, where $v$ represents an element of input $\mathbf{a}$, is a GP as well, see [Rasmussen et al., 2006]. Consequently, the observed data and the function derivative with respect to input $v$ at the query point $\mathbf{a}^*$ are also jointly Gaussian:

$$\begin{bmatrix} \hat{\mathbf{v}}_\mathbf{e} \\ \left.\frac{\partial v_e(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \left.\frac{\partial \mathbf{k}^T(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} \\ \left.\frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} & \left.\frac{\partial^2 k(\mathbf{a},\mathbf{a})}{\partial v \partial v}\right|_{\mathbf{a}^*} \end{bmatrix}\right).$$

Similarly, by the conditioning property of joint Gaussian distributions, the mean and variance of the derivative at our query point $\mathbf{a}^*$ with respect to input $v$ conditioned on the data $\mathcal{D}$ is given by:

$$\mu'(\mathbf{a}^*) = \left.\frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} \mathbf{K}^{-1}\hat{\mathbf{v}}_{\mathbf{e}}, \tag{4.15}$$

$$\sigma'^2(\mathbf{a}^*) = \left.\frac{\partial^2 k(\mathbf{a},\mathbf{a})}{\partial v \partial v}\right|_{\mathbf{a}^*} - \left.\frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} \mathbf{K}^{-1} \left( \left.\frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*} \right)^T . \tag{4.16}$$

We can also use this property of a derivative of a GP to infer the coefficient of correlation $\rho(\mathbf{a}^*)$ between the function value at a query point, $v_e(\mathbf{a}^*)$, and its derivative with respect to input $v$ at a query point, $\frac{\partial v_e(\mathbf{a})}{\partial v}\big|_{\mathbf{a}^*}$. Using the kernel function, we first find the covariance $\mathrm{COV}(\cdot,\cdot)$ between the function value $v_e(\mathbf{a}^*)$ and its derivative $\frac{\partial v_e(\mathbf{a})}{\partial v}\big|_{\mathbf{a}^*}$ using $\mathrm{COV}\left(v_e(\mathbf{a}^*), \left.\frac{\partial v_e(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*}\right) = \left.\frac{\partial k(\mathbf{a}^*,\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*}$, see [Rasmussen et al., 2006]. Then by the definition of the coefficient of correlation:

$$\rho(\mathbf{a}^*) = \frac{\mathrm{COV}\left(v_e(\mathbf{a}^*), \left.\frac{\partial v_e(\mathbf{a})}{\partial v}\right|_{\mathbf{a}^*}\right)}{\sigma(\mathbf{a}^*)\sigma'(\mathbf{a}^*)}, \tag{4.17}$$

where $\sigma(\mathbf{a}^*)$ is found from (4.9) and $\sigma'(\mathbf{a}^*)$ is found from (4.16).

In other words, given that we have learned the function $v_e(\mathbf{z}, v)$ as a GP, the partial derivative $\frac{\partial v_e(\mathbf{z},v)}{\partial v}$ is also a GP and we can predict its value at a query point $\mathbf{a}^* = \{\mathbf{z}, v_d\}$ conditioned on the data $\mathcal{D}$ as $\frac{\partial v_e(\mathbf{a}^*)}{\partial v}|\mathcal{D} = \mathcal{N}(\mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$ where $\mu'(\mathbf{a}^*)$ and $\sigma'^2(\mathbf{a}^*)$ can be found from (4.15) and (4.16), respectively.

Our analysis requires sufficient training data $\mathcal{D}$ and a GP kernel that can model our model mismatch to make the following two simplifying assumptions (see [Berkenkamp et al., 2015] where the derivative properties of GPs are used for stabilization):

**Assumption 3** *The actual error $\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)$ is normally distributed and described by the random variable: $Y := \mu(\mathbf{a}^*) - v_e(\mathbf{a}^*) \sim \mathcal{N}(0, \sigma^2(\mathbf{a}^*))$.*

**Assumption 4** *The partial derivative $\frac{\partial v_e(\mathbf{a}^*)}{\partial v}$ is also normally distributed such that we can define the random variable: $X := 1 + \frac{\partial v_e(\mathbf{a}^*)}{\partial v} \sim \mathcal{N}(1 + \mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$.*

Furthermore, $X$ and $Y$ are jointly correlated with some coefficient of correlation $\rho(\mathbf{a}^*)$ given by (4.17). We write this as a bivariate correlated normal random variable: $(Y, X) \sim \mathcal{N}(0, 1 + \mu'(\mathbf{a}^*), \sigma^2(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*), \rho(\mathbf{a}^*)) = \mathcal{N}(\mu_Y, \mu_X, \sigma_Y^2, \sigma_X^2, \rho)$.

We rewrite our probability bound (4.13) as: $\Pr\{|\frac{Y}{X}| < \hat{c}\} \geq 1 - \delta$, where the left-hand side or cumulative density function is found from $\Pr\{|\frac{Y}{X}| < \hat{c}\} = F(\hat{c}) - F(-\hat{c})$

where $F(\hat{c}) = \Pr\{\frac{Y}{X} < \hat{c}\}$ has an analytical form, see [Pollastri et al., 2015]:

$$F(\hat{c}) = L\left(\frac{t_a - t_b t_{\hat{c}}}{\sqrt{1+t_{\hat{c}}^2}}, -t_b, \frac{t_{\hat{c}}}{\sqrt{1+t_{\hat{c}}^2}}\right) + L\left(\frac{t_b t_{\hat{c}} - t_a}{\sqrt{1+t_{\hat{c}}^2}}, t_b, \frac{t_{\hat{c}}}{\sqrt{1+t_{\hat{c}}^2}}\right),$$

where $L(\cdot,\cdot,\cdot)$ is the bivariate normal integral and $t_a = \sqrt{\frac{1}{1-\rho^2}}(\frac{\mu_Y}{\sigma_Y} - \rho\frac{\mu_X}{\sigma_X})$, $t_b = \frac{\mu_X}{\sigma_X}$, $t_{\hat{c}} = \sqrt{\frac{1}{1-\rho^2}}(\frac{\sigma_X}{\sigma_Y}\hat{c} - \rho)$.

Therefore, to find the intermediate probabilistic bound $\hat{c}$, at each query point $\mathbf{a}^*$, we solve the nonlinear optimization problem:

$$\begin{aligned} \min \quad & \hat{c} \\ \text{s.t.} \quad & \hat{c} \geq 0 \\ & F(\hat{c}) - F(-\hat{c}) \geq \sqrt{1-\delta}. \end{aligned} \tag{4.18}$$

To find bound $c$ in (4.12) we use the intermediate bound $\hat{c}$ found from (4.18). To do this, we rewrite (4.14) as $\Pr\left\{\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c < \hat{c}\right\} \leq \sqrt{1-\delta}$. By introducing random variables $W \sim \mathcal{N}(c, 0)$ and $X \sim \mathcal{N}(1+\mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$, by Assumption 4 above, we can rewrite the inequality as a special case of the ratio of uncorrelated normal random variables $\Pr\left\{\frac{W}{X} < \hat{c}\right\} = F(\hat{c})$ where $(W, X) \sim \mathcal{N}(c, 1+\mu'(\mathbf{a}^*), 0, \sigma'^2(\mathbf{a}^*), 0)$. In this case, we can then find bound $c$ (mean of the numerator) such that $F(\hat{c}) = \sqrt{1-\delta}$.

## 4.3.3 Robust Linear Quadratic Regulator

Our aim is to track a reference trajectory with reference state $\mathbf{z}_{\text{ref}} = [y_{\text{ref}}, \dot{y}_{\text{ref}}, ..., y_{\text{ref}}^{(n-1)}]^T$ and reference input $v_{\text{ref}} = y_{\text{ref}}^{(n)}$ where $y_{\text{ref}}$ is the reference output. We design the desired input $v_d$ in (4.10) using a nominal LQR:

$$v_d = -\tilde{\mathbf{K}}(\mathbf{z} - \mathbf{z}_{\text{ref}}) + v_{\text{ref}}. \tag{4.19}$$

The gain $\tilde{\mathbf{K}} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$ is found by solving the algebraic Riccati equation $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \tilde{\mathbf{Q}} = \mathbf{0}$, $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}} > 0$ where matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained from the linear dynamics (4.4). The robustness term in (4.10) is designed as:

$$v_{\text{rob}} = \begin{cases} -c\frac{\mathbf{B}^T\mathbf{P}\mathbf{e}}{||\mathbf{B}^T\mathbf{P}\mathbf{e}||}, & \text{if } ||\mathbf{B}^T\mathbf{P}\mathbf{e}|| > \epsilon \\ -c\frac{\mathbf{B}^T\mathbf{P}\mathbf{e}}{\epsilon}, & \text{otherwise} \end{cases} \tag{4.20}$$

where $\mathbf{e} = \mathbf{z} - \mathbf{z}_{\text{ref}}$ is the tracking error, $c$ is the probabilistic bound found from (4.12) and $\epsilon > 0$ is some small user-selected parameter and $||\cdot||$ denotes the Euclidean norm (see [Helwa et al., 2019]).

## 4.4 Theoretical Guarantees

In this section, we show that under our proposed learning-based controller, the trajectory tracking error is uniformly ultimately bounded.

### 4.4.1 Uniform Ultimate Boundedness

We use Lyapunov theory to prove that using our proposed controller can probabilistically guarantee an ultimate bound on the tracking error. To do this, we make use of the following definition and Theorem 2 below (from [Spong et al., 2006]).

**Definition 3 (Uniform Ultimate Boundedness)** *A solution* $\mathbf{e}(t) : [t_0, \infty) \to \mathbb{R}^n$ *to* $\dot{\mathbf{e}} = \zeta(\mathbf{e})$ *with initial condition* $\mathbf{e}(t_0) = \mathbf{e}_0$ *is uniformly ultimately bounded (u.u.b.) with respect to a set* $S$ *if there is a non-negative constant* $T(\mathbf{e}_0, S)$ *such that* $\mathbf{e}(t) \in S \quad \forall t > t_0 + T(\mathbf{e}_0, S)$.

**Theorem 2** *Let* $V(\mathbf{e}(t))$ *be a Lyapunov function and let* $S$ *be any level set of* $V(\mathbf{e}(t))$*. Then* $\mathbf{e}(t)$ *is u.u.b. with respect to* $S$ *if* $\dot{V} < 0$ *for* $\mathbf{e}(t)$ *outside of* $S$.

**Theorem 3** *Consider the differentially flat system (2.12)-(2.13) and a smooth bounded reference state* $\mathbf{z}_{\text{ref}}(t)$ *and input* $v_{\text{ref}}(t)$ *trajectory. Suppose that Assumptions 1-4 hold and that bound* $c$ *satisfies (4.12). Then the tracking error* $\mathbf{e}(t) = \mathbf{z}(t) - \mathbf{z}_{\text{ref}}(t)$ *is uniformly ultimately bounded (u.u.b.) with probability greater than* $1 - \delta$ *using the proposed robust learning-based control governed by (4.5), (4.10), (4.19) and (4.20).*

*Proof:* Under the proposed robust learning-based control, the closed-loop dynamics are given by (2.13) and (4.11), where $v_d = -\tilde{\mathbf{K}}(\mathbf{z} - \mathbf{z}_{\text{ref}}) + v_{\text{ref}}$. By exploiting the integrator chain structure of matrices $\mathbf{A}$ and $\mathbf{B}$, we write the closed-loop error dynamics as: $\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})\mathbf{e} + \mathbf{B}(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})} v_{\text{rob}} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)))$. We propose the following Lyapunov function $V = \mathbf{e}^T\mathbf{P}\mathbf{e}$ where $\mathbf{P}$ is the positive definite matrix that solves the algebraic Riccati equation, i.e., $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\tilde{\mathbf{R}}^{-1}\mathbf{B}^T\mathbf{P} + \tilde{\mathbf{Q}} = \mathbf{0}$, $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}} > 0$, or equivalently, $(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})^T\mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}}) + \tilde{\mathbf{S}} = 0$, where $\tilde{\mathbf{S}} = \tilde{\mathbf{Q}} + \tilde{\mathbf{K}}^T\tilde{\mathbf{R}}\tilde{\mathbf{K}} > 0$ since $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}} > 0$. The time derivative of our Lyapunov function is $\dot{V} = 2\mathbf{e}^T\mathbf{P}(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})\mathbf{e} +$

$2\mathbf{e}^T\mathbf{P}\mathbf{B}(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{\text{rob}}+\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*)-v_e(\mathbf{a}^*)\right))$, or equivalently, by using the algebraic Ricatti relationship and defining $\mathbf{w} := \mathbf{B}^T\mathbf{P}\mathbf{e}$,

$$\dot{V} = -\mathbf{e}^T\tilde{\mathbf{S}}\mathbf{e} + 2\mathbf{w}^T(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{\text{rob}} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right)).$$

Since the first term $V_1 := -\mathbf{e}^T\tilde{\mathbf{S}}\mathbf{e} < 0$ is strictly negative, we consider only the second term $V_2 := 2\mathbf{w}^T(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{\text{rob}} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right))$. There are two cases. *Case 1:* In this case, $||\mathbf{w}|| > \epsilon$ and $v_{\text{rob}} = -c\frac{\mathbf{w}}{||\mathbf{w}||}$ in (4.20). The second term of $\dot{V}$ becomes $V_2 = 2\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c||\mathbf{w}|| + \mathbf{w}^T\left(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right)\right)\right)$. We can use the Cauchy-Schwartz inequality to show $V_2 \leq 2\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c||\mathbf{w}|| + ||\mathbf{w}||\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right)\right|\right)$. Since $c$ is an upper bound that satisfies (4.12), $V_2$ is less than zero with probability greater than $1-\delta$. *Case 2:* In this case $||\mathbf{w}|| \leq \epsilon$ and $v_{\text{rob}} = -c\frac{\mathbf{w}}{\epsilon}$ in (4.20). The second term of $\dot{V}$ becomes $V_2 \leq 2\mathbf{w}^T\left(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{\text{rob}} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right) = 2\mathbf{w}^T\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\frac{\mathbf{w}}{\epsilon} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right)$. Since $\max\left(2\mathbf{w}^T\left(-\hat{c}\frac{\mathbf{w}}{\epsilon} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right)\right) = \frac{\hat{c}\epsilon}{2}$, and using (4.14), the second term $V_2 \leq \frac{\hat{c}\epsilon}{2}$. Then $\dot{V} = V_1 + V_2 \leq -\mathbf{e}^\mathbf{T}\tilde{\mathbf{S}}\mathbf{e} + \frac{\hat{c}\epsilon}{2} < 0$ provided that:

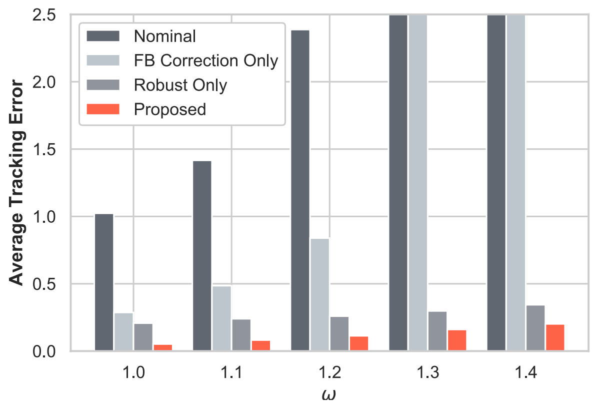
$$||\mathbf{e}|| > \sqrt{\frac{\hat{c}\epsilon}{2\lambda_{\text{min}}(\tilde{\mathbf{S}})}} := B_{\tilde{\mathbf{S}}}.$$

Let $S$ be a level set of V such that it contains $B_{\tilde{\mathbf{S}}}$. Since $\dot{V} < 0$ for $\mathbf{e}(t)$ outside of S, by Theorem 2, $\mathbf{e}(t)$ is u.u.b. with respect to $S$. Note that $\epsilon$ can be chosen to be very small and therefore the ball $B_{\tilde{\mathbf{S}}}$ can be made arbitrarily small.

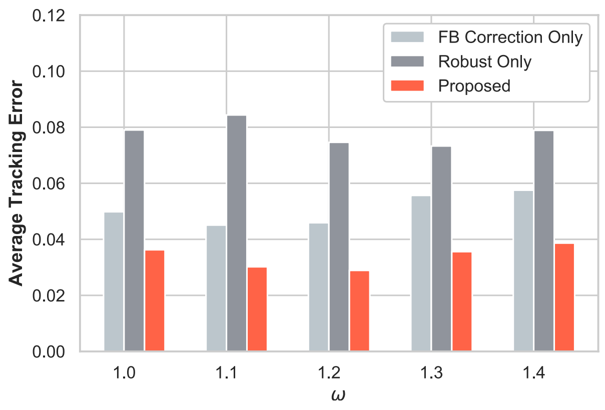
$\square$

## 4.5 Simulations

The proposed approach is verified via simulations on A) a SISO 1-D multirotor moving in the horizontal direction and B) the pole dynamics of an inverted pendulum on a cart (see [Rigatos et al., 2015]). For both simulations, $\delta = 0.01$ and $\epsilon = 0.1$. We compare performance under *Nominal* control (no learning), *FB Correction Only* (a learning-based control that only improves the FB linearization), *Robust Only* (a learning-based control that only uses a bound to design a robust LQR) and our *Proposed* learning-based control (as in Fig. 4.1).



(a) 1-D Quadrotor: Offline Learned Model



(b) 1-D Quadrotor: Online Learned Model

Figure 4.2: Average output tracking error for a 1-D multirotor simulation when using an (a) Offline Learned Model and an (b) Online Learned Model.

### 4.5.1 1-D Multirotor

Our simulation uses the following dynamics $\ddot{x} = T\sin(\theta) - \gamma\dot{x}, \quad \dot{\theta} = \frac{1}{\tau}(u - \theta)$, where $x$ is the horizontal position, $\theta$ is the pitch angle and input $u$ is the commanded pitch angle. The system dynamics (5.1) have a time constant $\tau = 0.2$, thrust $T = 10$ and drag $\gamma = 3$. Our nominal model (4.2) has a time constant $\tau = 0.15$, thrust $T = 10$ and $\gamma = 0$. Both our system and model are differentially flat in the output $y = x$. We use a GP to learn $v_e(\cdot)$ in (4.7). The GP uses a squared-exponential (SE) kernel parametrized with $\sigma_f^2 = 225, \sigma_\eta^2 = 0.1, \mathbf{L} = \text{diag}\{57, 2, 2, 66\}$, where these hyperparameters maximize the log-likehood of the data collected during the *Offline Case*. We consider two cases.

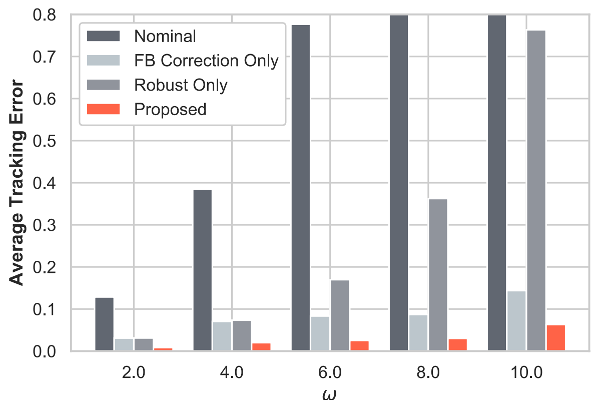


Figure 4.3: Average output tracking error for an inverted pendulum simulation.

**Offline Learned Model:** We consider 500 data points collected from $10s$ of tracking $y_{\text{ref}} = 2\sin(t)$ under nominal control (i.e., no learning). This GP model is fixed as we use it to follow different trajectories.

**Online Learned Model:** We update our GP model online based on the last 100 data points collected during the current trajectory tracking. The previously-tuned hyperparameters stay fixed.

In both cases, we compare our proposed approach with three other approaches. All controllers use LQR parameters $\tilde{\mathbf{Q}} = \text{diag}(100.0, 0.1, 0.1)$, $\tilde{\mathbf{R}} = 0.1$. We use a simulation time of $10s$ for each trajectory. We consider the following reference trajectory to track $y_{\text{ref}} = At\sin(\omega t)$. We fix $A = 0.4$ and vary $\omega$ between 1 and 1.4 to obtain progressively more aggressive trajectories. We compare average output tracking error in Fig. 4.2(a)-(b). In the *Offline Learned Model* simulation, we demonstrate that for some trajectories ($\omega \geq 1.3$) the *FB Linearization Only* case can cause instability. In this case, our *Proposed* approach still outperforms the *Robust Only* approach with an average tracking error reduction of 40-75%. Relying on an *Online Learned Model* improves the performance of all learning-based controllers. However, our *Proposed* approach achieves an average tracking error reduction of 50-65% over *Robust Only* and 27-37% over *FB Linearization Only*.

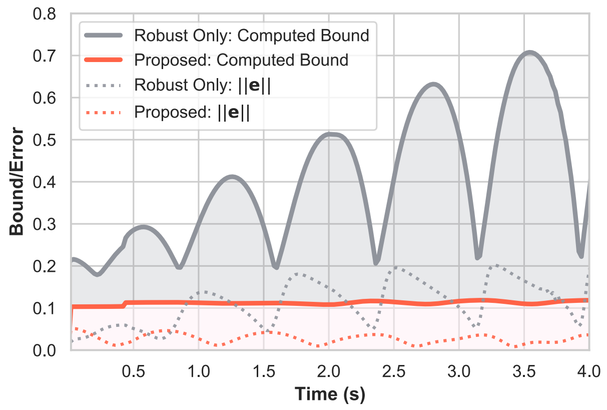


Figure 4.4: Computed tracking error bound (Theorem 3) and actual tracking error for the inverted pendulum when tracking $y_{\text{ref}} = 0.3t\sin(\omega t)$, $\omega = 4.0$, for *Proposed* and *Robust Only* approaches.

### 4.5.2 Inverted Pendulum

Our simulation uses the dynamics given in Chapter 3 (p. 112) in [Rigatos et al., 2015]. Our nominal model considers the mass of cart $M$, the mass of pendulum $m$, and the length of pendulum pole $l$ to be 1.5 kg, 0.05 kg and 0.8 m, respectively, while the actual systems parameters are 1.0 kg, 0.1 kg, and 0.5 m. All controllers use LQR parameters $\tilde{\mathbf{Q}} = \text{diag}(10.0, 10.0)$, $\tilde{\mathbf{R}} = 0.1$. We use a simulation time of 4.5$s$ for each trajectory. We consider the following reference trajectory to track $y_{\text{ref}} = At\sin(\omega t)$. We fix $A = 0.3$ and vary $\omega$ between 2 and 10. We compare average output tracking error in Fig. 4.3. In Fig. 4.4, we highlight the computed tracking error bound (based on Theorem 3). In this simulation, our *Proposed* approach achieves an average tracking error reduction of 50-70% over *FB Linearization Only*.

## 4.6 Summary

By exploiting differential flatness and the ability of GPs to predict derivatives and quantify uncertainty, we develop a learning-based controller that achieves high-accuracy tracking while guaranteeing safety.
The approach in this chapter has three novel contributions:

- We present a novel approach that uses a GP to both improve the FB linearization and quantify how well we are able to linearize the system.
- We demonstrate how our quantified uncertainty can be combined with a stan-

dard robust LQR to probabilistically guarantee an ultimate bound on the tracking error.

- We show through simulations how our proposed approach results in improved tracking performance over both *Strategy 1* (only improving the FB linearization) and *Strategy 2* (only quantifying how well a nominal controller linearizes the system).

The *key insight* in this chapter is that we can use the derivative properties of Gaussian processes to quantify a probabilistic bound on how well we linearize (using feedback linearization) the system.

# Chapter 5

# Flatness-Based Learned Stability Filter

**Exploiting Flatness Structure for Safe Learning**

## 5.1 Overview and Related Work

There is a growing interest in increased autonomy of safety-critical but uncertain and nonlinear systems, such as self-driving vehicles, unmanned aerial vehicles (UAVs), and mobile manipulators. This has motivated bridging formal safety analysis with the flexibility of machine learning to cope with large prior uncertainties.

### 5.1.1 Gaussian Processes

Gaussian processes (GPs) have gained popularity within the control community as a nonparametric machine learning approach that quantifies the uncertainty in its prediction. This uncertainty can be used to generate a probabilistic upper bound for the difference between the true and learned function value based on distance to the training data, see [Srinivas et al., 2012].

GPs are often used to learn the dynamics model which is then included in a model predictive control (MPC) framework. The GP uncertainty quantification can be used to tighten state and input constraints, see [Ostafew et al., 2016] or [McKinnon et al., 2019]. A major limitation is that the resulting optimization problem is generally non-convex, and slow and expensive to solve. Furthermore, this approach provides no stability guarantees for the controlled system.

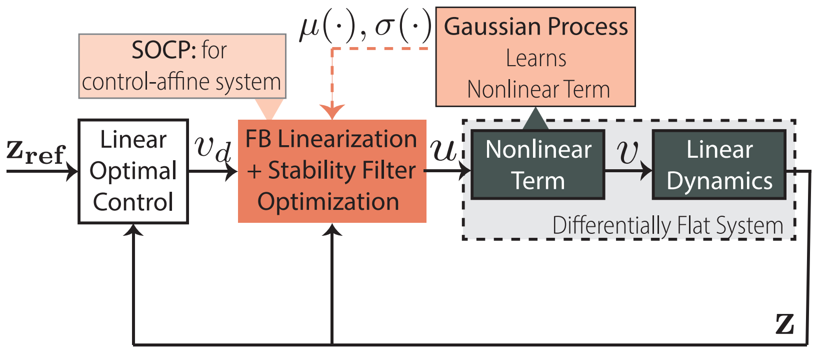

$\mu(\cdot), \sigma(\cdot)$
Gaussian Process Learns Nonlinear Term
SOCP: for control-affine system
$\mathbf{z}_{\mathbf{ref}}$
Linear Optimal Control
$v_d$
FB Linearization + Stability Filter Optimization
$u$
Nonlinear Term
$v$
Linear Dynamics
Differentially Flat System
$\mathbf{z}$

Figure 5.1: Overview of the proposed learning-based control architecture. Our proposed approach learns the nonlinear term as a Gaussian Process (GP). We combine feedback linearization with a stability filter by using the learned GP model to 1) optimize for an input $u$ that is most likely to feedback linearize the system (i.e., cancel the nonlinear term) 2) guarantee stability with high probability through a stability filter. For systems that are control-affine, we show that the resulting optimization is a convex second-order cone program (SOCP).

### 5.1.2 Exploiting Differential Flatness using Gaussian Processes

Stability guarantees, for example, asymptotic stability or tracking, have been combined with GPs by exploiting two structural assumptions about the dynamics: 1) the system is control-affine and 2) the system is either fully actuated, see [Umlauft et al., 2018], or underactuated but differentially flat, see [Greeff et al., 2020(a)]. Intuitively, differential flatness allows us to separate the nonlinear model into a linear dynamics component and a nonlinear term, see Fig. 5.1. Given that these assumptions (control-affine and differential flatness) are true for many first-principle models of physical systems, for example, multirotors, see [Mellinger et al., 2011], cranes, and manipulators, they are not limiting in practice.

However, previous work typically includes one of two additional limiting assumptions: 1) the actuation function is fully known, see [Umlauft et al., 2018] or [Zheng et al., 2020] (i.e., the unknown disturbance is only a function of the state) or 2) there are no actuation or input constraints, see [Greeff et al., 2020(a)], or [Umlauft et al., 2017]. Practically, there is often uncertainty in the actuation function as a result of delays, error in the system parameters (e.g., mass or inertia), or unaccounted dynamics of low-level controllers. Input constraints can represent physical limitations or may be added for user safety.

Differential flatness is commonly used in feedback (FB) linearization controllers, which attempt to cancel the nonlinear term such that outer-loop linear controllers can be designed based on the linear dynamics alone. In [Greeff et al., 2020(a)] and [Umlauft et al., 2017], the nonlinear term (or inverse nonlinear term) is learnt as a GP. The predicted GP mean is used to update the inverse nonlinear term (or FB linearization) while the uncertainty is used in a robust outer-loop linear controller. In [Greeff et al., 2020(a)], the control-affine structure is leveraged when taking the derivative of the learnt GP. In [Umlauft et al., 2017], similar to the approach we propose here, the control-affine structure is exploited in the selection of the GP kernel structure. Both approaches can guarantee asymptotic trajectory tracking. However, because the robustness is accounted for in the outer-loop controller, it is difficult to account for actuation or input constraints and, therefore, they are often neglected.

Using the strong assumption that the actuation function is known, in [Umlauft et al., 2018], a Lyapunov function is proposed that maximizes the probability of asymptotic stability of a known equilibrium accounting for input constraints. The approach is limited to fully actuated systems and can only be applied to stabilizing a known equilibrium. Another approach that makes this assumption, see [Umlauft et al., 2017], combines a known Control Lyapunov Function (CLF), to encourage stability, with input constraints in an optimization framework that can be solved as a quadratic program (QP). Control Lyapunov Functions (CLFs) have traditionally been used in minimum input-norm controllers to stabilize an equilibrium, see [Sontag et al., 1989].

Motivated by the work in [Taylor et al., 2019], our proposed approach uses the idea that, for systems that are FB linearizable, we can exploit their linear tracking error dynamics to construct a Lyapunov function to ensure exponential tracking convergence. Similar to [Taylor et al., 2019], we propose using such a Lyapunov function as a CLF. However, unlike [Taylor et al., 2019], we do not need to learn in an episodic fashion. Instead, we propose using a GP, with carefully selected kernel structure, to learn the nonlinear term and then leverage its quantified uncertainty in our controller.

### 5.1.3 Contributions

In this chapter, we develop a method to efficiently handle robust tracking guarantees and input constraints in the presence of model uncertainty. As illustrated in Fig. 5.1, our proposed approach uses the GP to learn the uncertain nonlinear term and combines feedback linearization, commonly applied to differentially flat systems, with a stability filter, described by a CLF, in an optimization framework. The three key contributions of this work are:

- We provide a novel approach that uses a GP to learn the uncertain nonlinear term and then use the GP in an optimization problem to optimize for a control input $u$ that is 1) most likely to cancel the nonlinear term while 2) guaranteeing a stability filter condition with high probability and 3) adhering to input constraints.
- We show that for control-affine systems, by exploiting this structure in the GP kernel selection, the resulting optimization is not only convex but can be solved efficiently as a second-order cone program (SOCP).
- We demonstrate, in simulation, a significant reduction in average computation over previous robustness methods using GPs, while still achieving high tracking performance. This makes our proposed approach suitable for online and onboard implementation in high-rate feedback loops, for example, on autonomous UAVs.

## 5.2 Problem Statement

Consider a single-input, single-output (SISO), control-affine system with state $\mathbf{x} \in \mathbb{R}^n$, input $u \in \mathbb{R}$ and output $y = h(\mathbf{x}) \in \mathbb{R}$:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u), \tag{5.1}$$

where $f(\mathbf{x}, u)$ is an *unknown* function and the dimension of the state $n$ is known. The input is subject to bound constraints, that is, $u_{\min} \leq u \leq u_{\max}$ where $u_{\min}$ and $u_{\max}$ are known minimum and maximum input constraints.

**Assumption 5** *The system (5.1) is differentially flat in the known output* $y = h(\mathbf{x}) \in \mathbb{R}$.

Under Assumption 5, our goal is to design a control law $u$ that:

- Achieves high trajectory tracking performance;
- Can be efficiently computed online;
- Guarantees that the closed-loop system satisfies robust stability (in the sense that tracking errors are bounded);
- Accounts for actuation/input constraints.

## 5.3 Methodology

We use Lemma 1 to rewrite the dynamics (5.1) as:

$$v = \psi(\mathbf{z}, u), \tag{5.2}$$

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}v, \tag{5.3}$$

where $\mathbf{z} = [y, \dot{y}, ..., y^{(n-1)}]^T$ and $v = y^{(n)}$. Moreover, if (5.1) is control-affine, we can use (2.14) in Lemma 1, to write (5.2) as a control-affine function:

$$v = a(\mathbf{z}) + b(\mathbf{z})u. \tag{5.4}$$

In the proposed approach, we exploit the differential flatness structure and learn (5.2) using a GP as described in Section 2.2. We combine feedback linearization with a stability filter by using the learned GP model to 1) optimize for an input $u$ that most likely feedback linearizes the system (5.2)-(5.3) while 2) guaranteeing that the stability filter, described by the CLF (5.12) is decreasing with high probability. We combine feedback linearization with a stability filter in an optimization framework, which can also account for input constraints. Further, we show that for control-affine systems, by carefully selecting the kernel (5.5), the resulting optimization can be described by a second-order cone program that can be solved in polynomial time by standard interior point methods.

**Kernel selection for control-affine systems:** For control-affine systems, the nonlinear map $\psi(\mathbf{a}) = \psi(\mathbf{z}, u)$ is affine in the control input, i.e., we can write $\psi(\mathbf{z}, u) = a(\mathbf{z}) + b(\mathbf{z})u$. We can encode this structure in the selection of the kernel of the GP as:

$$k(\mathbf{a}_i, \mathbf{a}_j) = k_a(\mathbf{z}_i, \mathbf{z}_j) + u_i k_b(\mathbf{z}_i, \mathbf{z}_j)u_j + \delta_{ij}\sigma_\eta^2, \tag{5.5}$$

where $\sigma_\eta^2$ is the observation noise and $k_a(\cdot, \cdot)$ and $k_b(\cdot, \cdot)$ are often selected to be common kernel functions (e.g., squared-exponential (SE) functions). The intuition for this kernel selection is that the predicted mean – that is, the mostly likely function (conditioned on the data) – will also be control-affine like our underlying function $\psi(\mathbf{z}, u)$. Further, since there is no parametric structure assumed for functions $a(\mathbf{z})$ and $b(\mathbf{z})$, this prior is still very flexible and can represent a rich class of nonlinear functions depending on the selection of kernels $k_a(\cdot, \cdot)$ and $k_b(\cdot, \cdot)$.

**Assumption 6** *$k_a$ and $k_b$ are positive definite kernels.*

**Assumption 7** *$k_a$ and $k_b$ are bounded kernels.*

**Lemma 2** *Given Assumption 6, then the kernel (5.5) is also positive definite. Moreover, if Assumption 7 holds, then the kernel (5.5) is also a bounded kernel.*

*Proof:* It follows from [Castaneda et al., 2020] that the affine dot product compound kernel, i.e., $u_i k_b(\mathbf{z}_i, \mathbf{z}_j) u_j$, is positive definite and bounded provided that $k_b(\mathbf{z}_i, \mathbf{z}_j)$ is positive definite and bounded. Consequently, the kernel (5.5) is also positive definite and bounded as it is the addition of two positive definite and bounded kernels.

□

Using this kernel structure (5.5), the predicted mean $\mu(\mathbf{a})$ and variance $\sigma^2(\mathbf{a})$ at any query point $\mathbf{a} = (\mathbf{z}, u)$, conditioned on $N_d$ noisy observations $\mathcal{D} = \{\mathbf{a}_i, \hat{\psi}(\mathbf{a}_i)\}_{i=1}^{N_d}$, are linear and quadratic in $u$, respectively, or more explicitly:

$$\mu(\mathbf{a}) = \gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z})u, \tag{5.6}$$

$$\sigma^2(\mathbf{a}) = \gamma_3(\mathbf{z}) + \gamma_4(\mathbf{z})u + \gamma_5(\mathbf{z})u^2, \tag{5.7}$$

where:

$$\gamma_1(\mathbf{z}) = \mathbf{k}_a(\mathbf{z})\mathbf{K}^{-1}\hat{\mathbf{\Psi}}, \quad \gamma_2(\mathbf{z}) = \mathbf{k}_b(\mathbf{z})\mathbf{K}^{-1}\hat{\mathbf{\Psi}},$$

$$\gamma_3(\mathbf{z}) = \left(k_a(\mathbf{z},\mathbf{z}) - \mathbf{k}_a(\mathbf{z})\mathbf{K}^{-1}\mathbf{k}_a^T(\mathbf{z})\right),$$

$$\gamma_4(\mathbf{z}) = -\left(\mathbf{k}_b(\mathbf{z})\mathbf{K}^{-1}\mathbf{k}_a^T(\mathbf{z}) + \mathbf{k}_a(\mathbf{z})\mathbf{K}^{-1}\mathbf{k}_b^T(\mathbf{z})\right),$$

$$\gamma_5(\mathbf{z}) = \left(k_b(\mathbf{z},\mathbf{z}) - \mathbf{k}_b(\mathbf{z})\mathbf{K}^{-1}\mathbf{k}_b^T(\mathbf{z})\right).$$

The vectors $\mathbf{k}_a(\mathbf{z}) = [k_a(\mathbf{z},\mathbf{z}_1), ..., k_a(\mathbf{z},\mathbf{z}_{N_d})]$ and $\mathbf{k}_b(\mathbf{z}) = [k_b(\mathbf{z},\mathbf{z}_1)u_1, ..., k_b(\mathbf{z},\mathbf{z}_{N_d})u_{N_d}]$.

Our proposed approach has three key components:

**Probabilistic Feedback Linearization - see Section 5.3-A** Based on our learned GP model for (5.2), we optimize for an input $u$ such that the predicted output of the GP is likely to match the designed input to the linear dynamics $v_d$, typically computed by a linear optimal controller based on the linear dynamics (5.3).

**Probabilistic Stability Filter - see Section 5.3-B** Using the learned GP model for (5.2), we include a stability filter that guarantees that (5.12) is decreasing with high probability.

**Linearization & Stability Filter Optimization - see Section 5.3-C** We combine the *probabilistic feedback linearization* with the *probabilistic stability filter*. At each time step, we propose to solve an optimization problem that finds the input $u$ that is mostly likely to result in a unity mapping between $v_d$ and the input seen by the linear dynamics (5.3) while ensuring robust tracking stability and input constraint satisfaction. For control-affine systems, we exploit the encoded structure and show that the resulting optimization is a second-order cone program.

## 5.3.1 Probabilistic Feedback Linearization

The objective of feedback linearization is to create a unity mapping between the desired input $v_d$, determined by some outer-loop linear controller, and the input $v = \psi(\mathbf{z}, u)$ seen by the linear dynamics. While $\psi(\mathbf{z}, u)$ is unknown, we have approximated this mapping from data using a GP. We propose finding an input $u$ that is most likely to result in such a feedback linearization. More precisely, we compute an input $u$ that minimizes the expected squared distance between $\psi(\mathbf{z}, u)$ and $v_d$:

$$\min_u \mathbb{E}(||\psi(\mathbf{z}, u) - v_d||^2).$$

Given that we have learnt the function $\psi(\mathbf{z}, u)$ using a GP, we can infer its value at any given query point $\mathbf{a}^* = (\mathbf{z}, u)$ conditioned on the data $\mathcal{D}$ as a Gaussian, i.e., $\psi(\mathbf{a}^*)|\mathcal{D} = \mathcal{N}(\mu(\mathbf{a}^*), \sigma^2(\mathbf{a}^*))$ where the mean and covariance are given by (2.15) and (2.16). Using this, we can rewrite the optimization problem as:

$$\min_u (\mu(\mathbf{z}, u) - v_d)^2 + \sigma^2(\mathbf{z}, u). \tag{5.8}$$

**Probabilistic feedback linearization for control-affine systems:** For control-affine systems, we can exploit the kernel structure in (5.5). This allows us to rewrite the mean $\mu(\mathbf{z}, u)$ as a linear function of input $u$, using (5.6), and the covariance $\sigma^2(\mathbf{z}, u)$ as a quadratic function of input $u$, using (5.7). Plugging in (5.6) and (5.7) into (5.8), we rewrite the optimization problem, neglecting constant terms with respect to the optimization variable $u$, as a convex quadratic program in $u$:

$$\min_u (\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}))u^2 + (2\gamma_1(\mathbf{z})\gamma_2(\mathbf{z}) - 2\gamma_2(\mathbf{z})v_d + \gamma_4)u. \tag{5.9}$$

*Remark:* The optimization (5.9) is convex because $\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}) \geq 0$ since the function $\gamma_5(\mathbf{z}) \geq 0$ is the predicted covariance of $b(\mathbf{z})$ in (5.4) conditioned on the data $\mathcal{D}$.

**Theorem 4** *The functions $\gamma_i(\mathbf{z})$ are real-valued and Lipschitz continuous on the compact set $\mathbf{z} \in \mathcal{Z}$. The functions $\gamma_2(\mathbf{z})$ and $\gamma_5(\mathbf{z})$ are never both zero. The desired input $v_d(\mathbf{z})$ is also real-valued and Lipschitz continuous on the compact set $\mathbf{z} \in \mathcal{Z}$. Under these assumptions, the input $u(\mathbf{z})$ computed using (5.9) is also Lipschitz continuous on the compact set $\mathbf{z} \in \mathcal{Z}$.*

*Proof:* The solution of (5.9) has a closed-form solution:

$$u = \frac{-\gamma_1(\mathbf{z})\gamma_2(\mathbf{z}) + \gamma_2(\mathbf{z})v_d(\mathbf{z}) - \frac{1}{2}\gamma_4(\mathbf{z})}{\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z})}.$$

The numerator is Lipschitz continuous on $\mathcal{Z}$ since it is a linear combination of the products of Lipschitz continuous functions that are bounded on $\mathcal{Z}$. Similarly, the denominator is also Lipschitz continuous on $\mathcal{Z}$. Since, $\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}) > 0$, it follows that the resulting quotient is also Lipschitz continuous on $\mathcal{Z}$.

$\square$

While the resulting control law (5.9) is Lipschitz continuous, it cannot guarantee robust stability and tracking convergence. We propose extending the probabilistic feedback linearization formulation (5.9) by also including a stability filter that guarantees tracking convergence with high probability.

## 5.3.2 Probabilistic Stability Filter

When the nonlinear term (5.2) is known, we can exploit the structure of differentially flat systems (5.2)-(5.3) to construct a Control Lyapunov Function (CLF). We will use this CLF as a stability filter when the nonlinear term is uncertain.

Using standard feedback linearization, we can design the controller $u$ to cancel the nonlinear term (5.2) and then select the input to the linear dynamics $v$ such that the resulting linear error dynamics are Hurwitz. Consider a smooth reference state $\mathbf{z}_{\text{ref}}(t)$ and reference input $v_{\text{ref}}(t)$, we can write the tracking error dynamics of (5.2)-(5.3) as:

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{z} + \mathbf{B}\psi(\mathbf{z}, u) - \dot{\mathbf{z}}_{\text{ref}}, \tag{5.10}$$

where the tracking error is $\mathbf{e} = \mathbf{z} - \mathbf{z}_{\text{ref}}$ and $\dot{\mathbf{z}}_{\text{ref}} = [\dot{y}_{\text{ref}}, ..., y_{\text{ref}}^{(n-1)}, v_{\text{ref}}]^T$.

Consider a standard feedback linearization controller $u = \psi^{-1}(\mathbf{z}, v_d)$ where

$$v_d = -\tilde{\mathbf{K}}\mathbf{e} + v_{\text{ref}} \tag{5.11}$$

is designed such that the closed-loop error dynamics are stable. Under this control law, it follows that the error dynamics in (5.10) simplifies to:

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})\mathbf{e},$$

where $\tilde{\mathbf{K}}$ is selected such that $(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})$ is Hurwitz.

Given the Hurwitz stability of the closed-loop system it follows from converse stability theorems that we can construct a Lyapunov function that guarantees exponential tracking convergence, see [Taylor et al., 2019]. Specifically, we can select the Lyapunov function $V(\mathbf{e}) = \mathbf{e}^T\mathbf{P}\mathbf{e}$ where $\mathbf{P} > 0$ is a positive definite matrix that satisfies the algebraic Riccati equation:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\tilde{\mathbf{R}}^{-1}\mathbf{B}^T\mathbf{P} + \tilde{\mathbf{Q}} = \mathbf{0},$$

for selected positive definite matrices $\tilde{\mathbf{Q}} > 0$, $\tilde{\mathbf{R}} > 0$.

**Definition 4** *A function $V : \mathbb{R}^n \to \mathbb{R}_+$ is a Control Lyapunov Function (CLF) for (5.10) certifying exponential stability if there exists positive constants $c_1, c_2, c_3 > 0$ such that:*

$$c_1||\mathbf{e}||^2 \leq V(\mathbf{e}) \leq c_2||\mathbf{e}||^2,$$
$$\dot{V}(\mathbf{e}) \leq -c_3V(\mathbf{e}).$$

We can use the previously constructed Lyapunov function $V(\mathbf{e}) = \mathbf{e}^T\mathbf{P}\mathbf{e}$ as a CLF with positive constants $c_1 = \lambda_{\min}(\mathbf{P}), c_2 = \lambda_{\max}(\mathbf{P})$ and $c_3 = \frac{\lambda_{\min}(\mathbf{S})}{\lambda_{\max}(\mathbf{P})}$ where $\mathbf{S} = \tilde{\mathbf{Q}} + \tilde{\mathbf{K}}^T\tilde{\mathbf{R}}\tilde{\mathbf{K}}$. In the absence of a chosen controller $u$, the decreasing time derivative condition $\dot{V}(\mathbf{e}) \leq -c_3V(\mathbf{e})$ in Definition 4 becomes:

$$\mathbf{e}^T\mathbf{P}(\mathbf{A}\mathbf{z} + \mathbf{B}\psi(\mathbf{z}, u) - \dot{\mathbf{z}}_{\text{ref}}) \leq -c_3\mathbf{e}^T\mathbf{P}\mathbf{e}. \tag{5.12}$$

We probabilistically bound the error between the true nonlinear function value (5.2) and the learnt mean value (2.15).

**Assumption 8** *The function $\psi(\mathbf{a})$ has a bounded reproducing kernel Hilbert space (RKHS) norm $||\psi(\mathbf{a})||_k$ with respect to the kernel $k(\mathbf{a}_i, \mathbf{a}_j)$ of the GP, and the observation noise $\eta$ is uniformly bounded by $\sigma_\eta$.*

**Theorem 5** *[Srinivas et al., 2012] Given Assumption 8. Let $\delta \in (0,1)$, then:*

$$Pr\{\forall \mathbf{a} \in \mathcal{A}, |\mu(\mathbf{a}) - \psi(\mathbf{a})| \leq \xi^{1/2}\sigma(\mathbf{a})\} \geq 1-\delta,$$

*where $Pr\{\cdot\}$ is the probability, $\mathcal{A}$ is compact, $\mu(\mathbf{a})$ is the GP mean, $\sigma^2(\mathbf{a})$ is the GP covariance and*

$$\xi = 2||\psi(\mathbf{a})||_k + 300\gamma \ln^3((N+1)/\delta),$$

*where $\gamma \in \mathbb{R}$ is the maximum information gain.*

We now exploit Theorem 5 to rewrite the decreasing CLF condition in (5.12) using the learnt GP mean (2.15) and variance (2.16) such that this condition holds with high probability.

We can rewrite the decreasing CLF condition in (5.12) using $\mathbf{e}^T\mathbf{P}(\mathbf{A}\mathbf{z}+\mathbf{B}\psi(\mathbf{z},u)+\mathbf{B}v_d-\mathbf{B}v_d-\dot{\mathbf{z}}_{\text{ref}}) \leq -c_3\mathbf{e}^T\mathbf{P}\mathbf{e}$, where $v_d$ comes from (5.11). Using the algebraic Riccati equation, this simplifies to $-\mathbf{e}^T\mathbf{S}\mathbf{e} + 2\mathbf{e}^T\mathbf{P}\mathbf{B}(\psi(\mathbf{z},u)-v_d) \leq -c_3\mathbf{e}^T\mathbf{P}\mathbf{e}$.

We have learnt $\psi(\mathbf{z},u)$ as a GP. Defining $w := \mathbf{e}^T\mathbf{P}\mathbf{B}$ and recalling that the query $\mathbf{a}$ comprises of the state and input, i.e., $\{\mathbf{z},u\}$, we use Theorem 5 to obtain:

$$\Pr\left\{w\left(\psi(\mathbf{z},u)-v_d\right) \leq w\left(\mu(\mathbf{z},u)-v_d\right) + |w|\xi^{1/2}\sigma(\mathbf{z},u)\right\} \geq 1-\delta.$$

We use this probabilistic condition to rewrite (5.12):

$$-\mathbf{e}^T\mathbf{S}\mathbf{e} + 2w(\mu(\mathbf{z},u)-v_d) + 2|w|\xi^{1/2}\sigma(\mathbf{z},u) \leq -c_3\mathbf{e}^T\mathbf{P}\mathbf{e}, \tag{5.13}$$

where $c_3 = \frac{\lambda_{\min}(\mathbf{S})}{\lambda_{\max}(\mathbf{P})}$. While the constraint (5.13) is not necessarily convex, for systems that are control-affine, we propose exploiting this structure in the kernel selection of the GP (5.5).

**Probabilistic stability filter for control-affine systems:** For control-affine systems, we choose to exploit the kernel structure in (5.5). By Lemma 2, the kernel (5.5) is bounded and positive definite and, therefore, we can construct a corresponding RKHS. We make a similar assumption to Assumption 8 for control-affine systems and kernel (5.5) such that we can similarly apply Theorem 5.

**Assumption 9** *The control-affine nonlinear function (5.4) has a bounded RKHS norm with respect to kernel (5.5) used in the GP, and the observation noise $\eta$ is uniformly bounded by $\sigma_\eta$.*

Under Assumption 9, we apply Theorem 5 and can rewrite the probabilistic decreasing CLF condition (5.13) using the mean (5.6), which is linear in control input

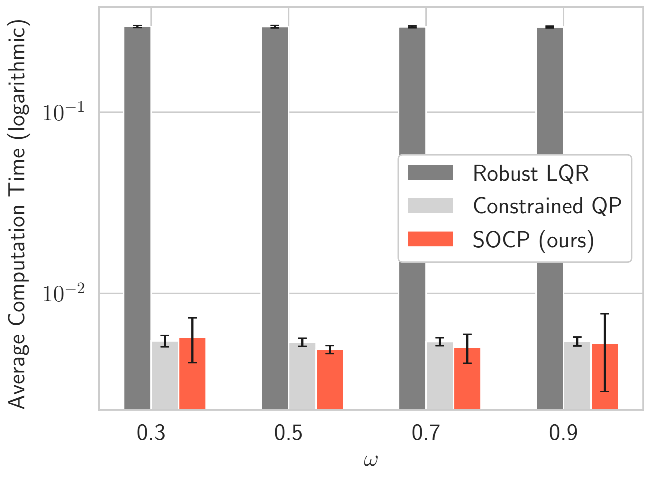


Figure 5.2: Comparison of average computation times where the error bars correspond to one standard deviation. On average *Robust LQR* requires around 0.3s to compute an input while our proposed *SOCP* requires 0.005s.

$u$, and covariance (5.7), which is quadratic in control input $u$. Plugging (5.6) and (5.7) into (5.13), the filter condition becomes:

$$2w(\gamma_1(\mathbf{z}) + \gamma_2(\mathbf{z})u - v_d) + 2|w|\xi^{1/2}\sqrt{\gamma_3(\mathbf{z}) + \gamma_4(\mathbf{z})u + \gamma_5(\mathbf{z})u^2} \leq -c_3\mathbf{e}^T\mathbf{P}\mathbf{e} + \mathbf{e}^T\mathbf{S}\mathbf{e}. \tag{5.14}$$

## 5.3.3 Linearization and Stability Filter Optimization

We combine feedback linearization with the safety filter by optimizing (5.8) subject to the probabilistically robust decreasing CLF condition (5.13). We can write this as an optimization problem including input constraints:

$$\begin{aligned} \min_{u,d} \quad & (\mu(\mathbf{z},u) - v_d)^2 + \sigma^2(\mathbf{z},u) + \rho d^2 \\ \text{s.t.} \quad & 2w\,(\mu(\mathbf{z},u) - v_d) + 2|w|\xi^{1/2}\sigma(\mathbf{z},u) \leq \mathbf{e}^T(-c_3\mathbf{P} + \mathbf{S})\mathbf{e} + d, \\ & u_{\min} \leq u \leq u_{\max}, \end{aligned} \tag{5.15}$$

where $d$ is a slack variable added to ensure feasibility of the above optimization problem and $\rho$ is a large weight. This optimization problem is not necessarily convex.

**Optimization for control-affine systems:** For control-affine systems (5.4), we use the simplications made in Section 5.3 such that we can rewrite the optimization (5.15) as a second-order cone program (SOCP) as stated below in Theorem 6.

**Theorem 6** *Given Assumptions 5, 6, 7 and 9, the optimization problem (5.15) can* *be written as a convex optimization problem. Moreover, it is a SOCP.*

*Proof:* We can rewrite both the CLF constraint (5.14), including the slack variable $d$, and the convex quadratic optimization problem (5.9) as second-order cone (SOC) constraints.

In (5.14), we can rewrite:

$$\sqrt{\gamma_3(\mathbf{z}) + \gamma_4(\mathbf{z})u + \gamma_5(\mathbf{z})u^2} = \left|\left| \begin{bmatrix} \sqrt{\gamma_5(\mathbf{z})}u + \frac{\gamma_4(\mathbf{z})}{2\sqrt{\gamma_5(\mathbf{z})}} \\ \sqrt{\gamma_3(\mathbf{z}) - \frac{\gamma_4^2(\mathbf{z})}{4\gamma_5(\mathbf{z})}} \end{bmatrix} \right|\right|_2$$

where $||\cdot||_2$ denotes the $L_2$-norm. This is possible because $\gamma_3(\mathbf{z}) - \frac{\gamma_4^2(\mathbf{z})}{4\gamma_5(\mathbf{z})} \geq 0$ since the covariance in (5.7) is positive. We can, therefore, rewrite the first SOC constraint in the standard form:

$$||\bar{\mathbf{A}}_\mathbf{1}\bar{\mathbf{u}} + \bar{\mathbf{b}}_\mathbf{1}||_2 \leq \bar{\mathbf{c}}_\mathbf{1}\bar{\mathbf{u}} + \bar{\mathbf{d}}_\mathbf{1}, \tag{5.16}$$

where the optimization variables $\bar{\mathbf{u}} = [u, d, f]^T$ include the input $u$, the slack variable $d$ and a dummy variable $f$. The matrix $\bar{\mathbf{A}}_\mathbf{1} = \text{diag}(|w|\sqrt{\gamma_5(\mathbf{z})}, 0, 0)$ and the vector $\bar{\mathbf{b}}_\mathbf{1} = [|w|\frac{\gamma_4(\mathbf{z})}{2\sqrt{\gamma_5(\mathbf{z})}}, |w|\sqrt{\gamma_3(\mathbf{z}) - \frac{\gamma_4^2(\mathbf{z})}{4\gamma_5(\mathbf{z})}}, 0]^T$. By rewriting (5.14), the vectors $\bar{\mathbf{c}}_\mathbf{1} = [\frac{-w\gamma_2(\mathbf{z})}{\xi^{1/2}}, \frac{1}{2\xi^{1/2}}, 0]$, and $\bar{\mathbf{d}}_\mathbf{1} = \frac{w}{\xi^{1/2}}(v_d - \gamma_1(\mathbf{z})) + \frac{1}{2\xi^{1/2}}\mathbf{e}^T(\mathbf{S} - c_3\mathbf{P})\mathbf{e}$.

We rewrite the optimization (5.9) including a dummy variable $f \geq (\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}))u^2 + \rho d^2$ where $\rho$ is a large weight and $d$ is the slack variable. It follows that $0 \geq 4((\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}))u^2 + \rho d^2) - 4f$ which can be rewritten as $(1+f)^2 \geq 4((\gamma_2^2(\mathbf{z}) + \gamma_5(\mathbf{z}))u^2 + \rho d^2) + (1-f)^2$. Since both sides of the inequality are positive, we can rewrite this condition as:

$$\left|\left| \begin{bmatrix} 2\sqrt{\gamma^2(\mathbf{z}) + \gamma_5(\mathbf{z})}u \\ 2\rho^{1/2}d \\ 1-f \end{bmatrix} \right|\right|_2 \leq 1 - f,$$

which allows us to write the second SOC constraint in the standard form as:

$$||\bar{\mathbf{A}}_\mathbf{2}\bar{\mathbf{u}} + \bar{\mathbf{b}}_\mathbf{2}||_2 \leq \bar{\mathbf{c}}_\mathbf{2}\bar{\mathbf{u}} + \bar{\mathbf{d}}_\mathbf{2}, \tag{5.17}$$

where $\bar{\mathbf{A}}_\mathbf{2} = \text{diag}(2\sqrt{\gamma^2(\mathbf{z}) + \gamma_5(\mathbf{z})}, 2\rho^{1/2}, -1)$, $\bar{\mathbf{b}}_\mathbf{2} = [0, 0, 1]^T$, $\bar{\mathbf{c}}_\mathbf{2} = [0, 0, 1]^T$ and

$\bar{\mathbf{d}}_\mathbf{2} = 1$. We rewrite the optimization problem in standard SOCP form:

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \begin{bmatrix} 2\gamma_1(\mathbf{z})\gamma_2(\mathbf{z}) - 2\gamma_2(\mathbf{z})v_d + \gamma_4 & 0 & 1 \end{bmatrix} \bar{\mathbf{u}} \\ \text{s.t.} \quad & \text{SOC constraints (5.16) \& (5.17)}, \\ & u_{\min} \le u \le u_{\max}, \end{aligned} \tag{5.18}$$

where we recall that $\bar{\mathbf{u}} = [u, d, f]^T$ includes the input $u$, the slack variable $d$ and the dummy variable $f$.

□

*Remark:* SOCPs can be solved in polynomial time by interior point methods, see [Nesterov et al., 1994].

At each time step, we solve the SOCP (5.18), which efficiently solves for an input $u$ that balances feedback linearization objectives with robust stability requirements and input constraints.

## 5.4 Simulations

We compare our proposed *SOCP* (5.18) method, with similar GP learning-based methods on a SISO 1-D quadcoptor moving in the horizontal direction. The dynamics follow [Greeff et al., 2020(a)], with $\ddot{x} = T\sin(\theta) - \gamma\dot{x}$ and $\dot{\theta} = \frac{1}{\tau}(u - \theta)$, where $x$ is the horizontal position, $\theta$ is the pitch angle, and the input $u$ is the commanded pitch angle.

To compare the algorithms in the unconstrained case, input constraints are neglected, and our proposed *SOCP* method is compared against the *closed-form* solution of (5.9), a *nominal LQR* that uses an inaccurate prior model, and a learned *robust LQR* from [Greeff et al., 2020(a)]. To compare the algorithms in the input-constrained case, input constraints are included, our proposed *SOCP* method is compared against a *constrained QP* that optimizes (5.9) subject to input constraints, *nominal LQR* that uses an inaccurate prior model and saturates the inputs at the constraints, and a learned *robust LQR* from prior work with saturated inputs.

For all simulations, we use $\xi^{1/2} = 2$ and $\rho = 625$. The true dynamics have a time constant $\tau = 0.2$, thrust $T = 10$, and drag $\gamma = 0.3$. The nominal model has estimated parameters $\hat{\tau} = 0.15$, $\hat{T} = 7$, and $\hat{\gamma} = 0$. Both these models are differentially flat in the output $y = x$. A GP, using the kernel function (5.5), is used to learn the unknown nonlinear term $\psi(\mathbf{z}, u)$. All GP parameters are optimized to minimize the GP's log-likelihood over the training data. The training trajectory is generated using the *nominal LQR*, with gains $\tilde{\mathbf{Q}} = \text{diag}(20, 15, 5)$ and $\tilde{\mathbf{R}} = 0.1$, and feedback linearization, based on the nominal model, to track $y_{\text{ref}} = At\sin(\omega t)$ with $A = 1$, $\omega = 1$ for 5 seconds.

To compare closed-loop performance, each controller's tracking performance is assessed along 4 different trajectories with $\omega = 0.3,\ 0.5,\ 0.7,\ 0.9$ and $A = 0.2$. The average output tracking error and computation times are compared in Fig.5.3.

In the unconstrained case, our proposed *SOCP* method provides up to an 90% decrease in average tracking error as compared with *robust LQR*, with larger decreases in error occurring when the trajectory is further away from the training data. In the constrained case, using input constraints $-15 \leq u \leq 15$, the average tracking error of our proposed *SOCP* approach is nearly as good as the *robust LQR*, and up to 85% less for trajectories further away from the training trajectory. We note that the tracking errors of all the compared approaches increase significantly and are comparable in the constrained cases when $\omega = 0.7$ and $\omega = 0.9$. This is due to the input constraints being reached for a significant portion of the trajectory, dominating the tracking error. Additionally, our *SOCP* approach has an average computational time nearly two orders of magnitude smaller than the *robust LQR* case and is comparable to a standard QP, significantly reducing the required computational power.

Our proposed approach (SOCP) has significantly better performance than the constrained QP approach at a similar computational cost. While a Robust LQR can outperfom us in rare cases (i.e., when the trajectory is infeasible for the given input constraints) it comes at a significantly higher computational cost. Our proposed approach efficiently handles robust tracking stability and input-based constraints in the presence of model uncertainty by exploiting the structure of control-affine differentially flat systems. As future work, we will investigate a data-driven CLF selection and extend our approach to include Control Barrier Functions [Fan et al., 2020].

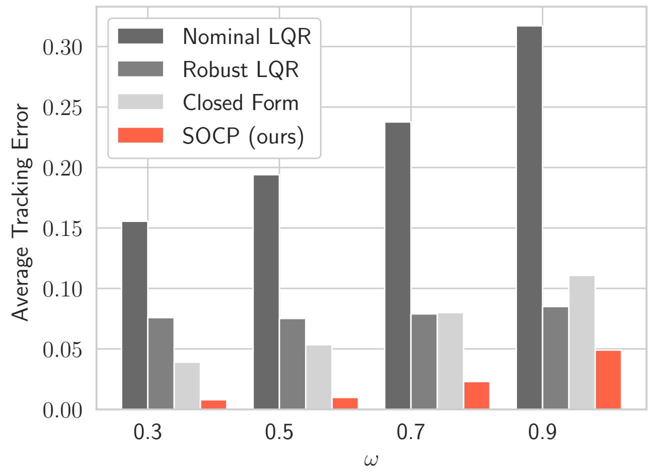
## 5.5 Summary

The approach in this chapter has three novel contributions:

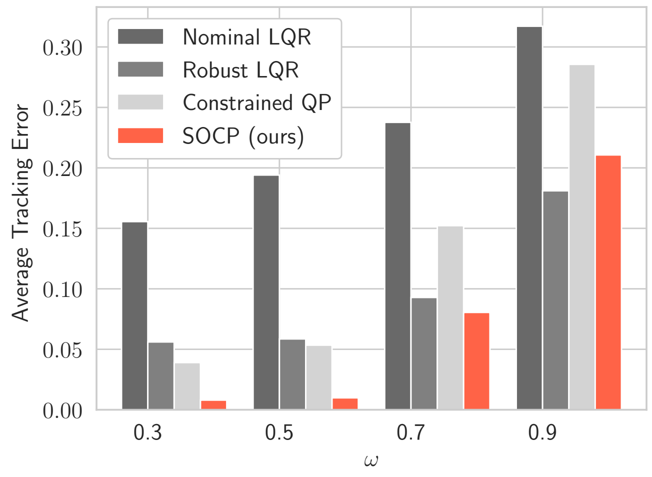
- We provide a novel approach that uses a GP to learn the uncertain nonlinear term and then use the GP in an optimization problem to optimize for a control input $u$ that is 1) most likely to cancel the nonlinear term while 2) guaranteeing a stability filter condition with high probability and 3) adhering to input constraints.

- We show that for control-affine systems, by exploiting this structure in the GP kernel selection, the resulting optimization is not only convex but can be solved efficiently as a second-order cone program (SOCP).
- We demonstrate, in simulation, a significant reduction in average computation over previous robustness methods using GPs, while still achieving high tracking performance. This makes our proposed approach suitable for online and onboard implementation in high-rate feedback loops, for example, on autonomous UAVs.

The *key insight* in this chapter is that for control-affine systems we can exploit this structure in the GP kernel selection to ensure that safe probabilistic feedback linearization is a convex optimization that can be solved efficiently as a second-order cone program (SOCP).



(a) Average tracking error w/o input constraints



(b) Average tracking error w/ input constraints

Figure 5.3: Average tracking error (a) without input constraints and (b) with input constraints for *nominal LQR* (no learning), *robust LQR*, *constrained QP* (or *closed form* without input constraints) optimizing (5.9), and our proposed *SOCP* approach (red).

# Part II

# Autonomous Vision-Based Flight

# Chapter 6

# Background on Multirotor Visual Teach and Repeat

## 6.1 Overview

UAVs are required to operate safely in beyond visual line of sight (VLOS) operations. Most outdoor operations rely predominantly on GPS. However, GPS can be susceptible to jamming or interference and may have limited accuracy to reliably support close-proximity, safety-critical or high-value autonomous flight operations.

In this chapter, we present a vision-based route-following system for autonomous navigation of UAVs, see [Warren et al., 2019]. Furthermore, we demonstrate its use as a backup system in the event of GPS loss in field experiments in Section 6.4. In these experiments, we show how our system can be used to autonomously return the UAV home by relying predominantly on an onboard camera and without reliance on any external infrastructure.

Sections 6.1, 6.2, 6.3 and 6.4.1 are taken from [Warren et al., 2019]. Additional figures not included in [Warren et al., 2019] are Fig. 6.3, Fig. 6.4, Fig. 6.8, Fig. 6.9, Fig. 6.10, Fig. 6.11, Fig. 6.12 and Fig. 6.13. This chapter includes an additional unpublished Section 6.5 on the *Challenges to Autonomous Visual Navigation*.

We use a vision-based navigation framework known as Visual Teach and Repeat (VT&R). Traditionally, VT&R has been used on wheeled ground vehicles as a path-following approach to autonomously drive a previously traversed route, see [Furgale et al., 2010] or [Paton et al., 2016]. It does this by estimating path offset, through matching visual features from a live view to those in a locally metric map, which is then corrected by a path-following controller, see [Ostafew et al., 2016].

VT&R on ground vehicles can be used for a variety of applications without external infrastructure, for example, navigating factory floors, orchards, mines, urban road networks, and exploratory search-and-return missions. As described in [Warren et al., 2019], using VT&R on UAVs has a number of unique use cases: just-in-time deliveries between warehouses, where flight paths are generally restricted to a few, high-frequency routes; monitoring of sensitive assets such as property borders or high-value infrastructure; and autonomous patrol in close-proximity environments, where poor sky view and jamming are notable concerns.

Our visual system is designed for flight approximately 10 - 20 m above ground. This is to avoid significant motion blur prevalent in low-altitude flight as well as significant depth uncertainty prevalent for small-baseline stereo cameras at high altitude.

## 6.2 Hardware System

As shown in Fig. 6.1, 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, e.g. a change from teach phase to repeat phase).

## 6.3 Visual Teach and Repeat Navigation

This section summarizes the visual teach and repeat (VT&R), see [Furgale et al., 2010], navigation system used in the experiments in [Warren et al., 2019]. In VT&R, we consider two phases.

**Teach:** During the *teach* phase, the UAV flies using autonomous GPS waypoint following while the VT&R algorithm performs passive visual odometry (VO). The purpose of the teach path is to create a map of visual landmarks along the path.

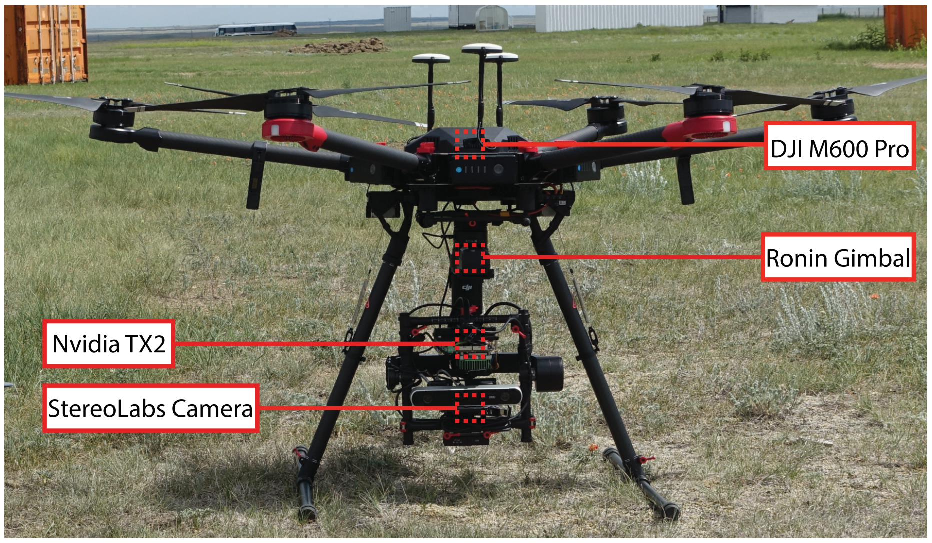


Figure 6.1: Multirotor system used for autonomous vision-based flight experiments. We use a DJI M600 Pro with Ronin-MX gimbal and a StereoLabs camera. All computation is performed onboard on the Nvidia TX2.

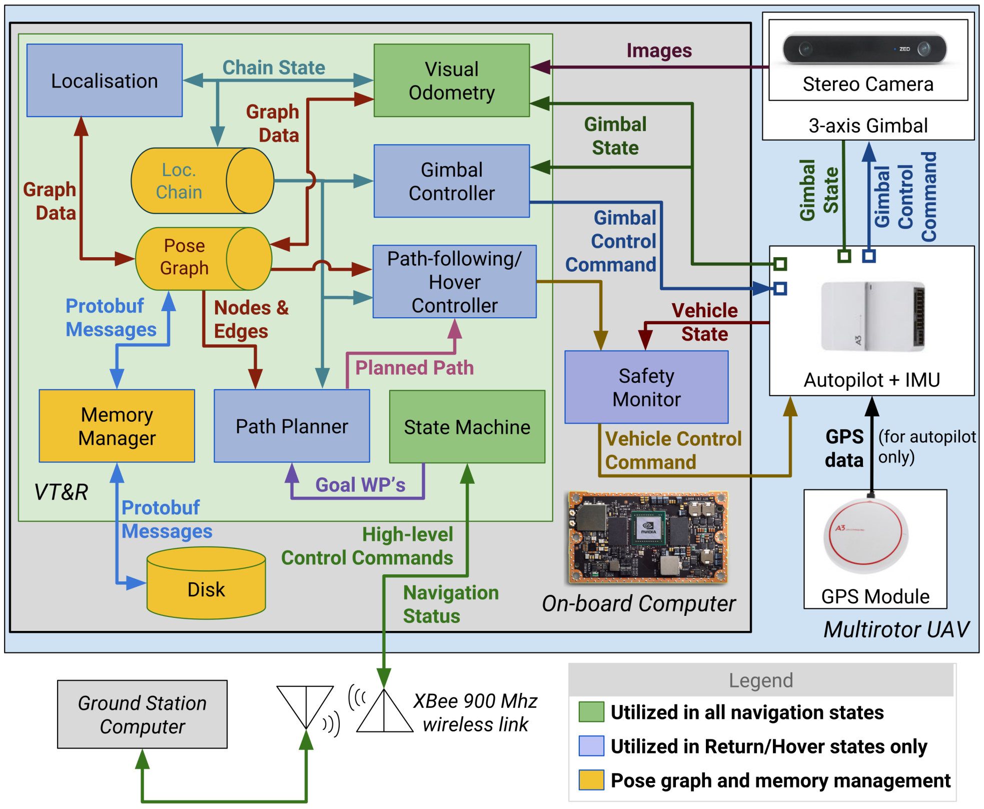


Figure 6.2: The architecture of the VT&R system for multirotor UAVs.

This taught path is stored as a set of vertices and edges which include the landmarks observed at each vertex.

**Repeat:** During the *repeat* phase, GPS is disabled and the vehicle needs to perform autonomous navigation (of the teach path) using vision-based navigation. During the repeat phase, in addition to performing the same VO as in teach, the VT&R algorithm performs visual localization using a local segment from the taught path. The vehicle repeats the taught path by sending high-frequency estimate to a path-following controller.

The VT&R software system consists of several interacting components - see Fig. 6.2. We mention 4 key components: 1) visual odometry (VO), 2) visual localization, 3) gimbal control and 4) path-following control.

### 6.3.1 Visual Odometry

During both the *teach* and *repeat* phases, grayscale image pairs are captured by the calibrated stereo camera (in Fig. 6.1) at a frame rate of $\sim$ 15Hz, see [Warren et al., 2019]. Speeded Up Robust Features (SURF) feature matching is used with respect to the last dropped keyframe vertex. The raw matches are then passed through a Maximum Likelihood Estimation SAmple Consensus (MLESAC) robust estimator to find the relative transform to the last keyframe. If the motion exceeds a threshold, the frame is set as a keyframe. The features (including the 3-D landmark position associated with each feature) and vehicle-to-sensor transform are stored in a vertex in the pose graph. The relative transform is stored as an edge to the previous vertex. This set of dead-reckoned linked poses represents the taught privileged path to be repeated.

### 6.3.2 Visual Localization

During the *repeat* phase an additional thread performs visual matching to the local map of 3-D landmarks in the privileged path. As seen in Fig. 6.3, to enable this process, a localization chain is used to keep track of important vertices in the graph and their respective transforms. We use a ‘tree’ model to name vertices in the chain:

- twig $\mathfrak{w}$ - closest vertex on current path with successfully estimated transform relative to the privileged path.
- branch $\mathfrak{b}$ - corresponding vertex on privileged path.

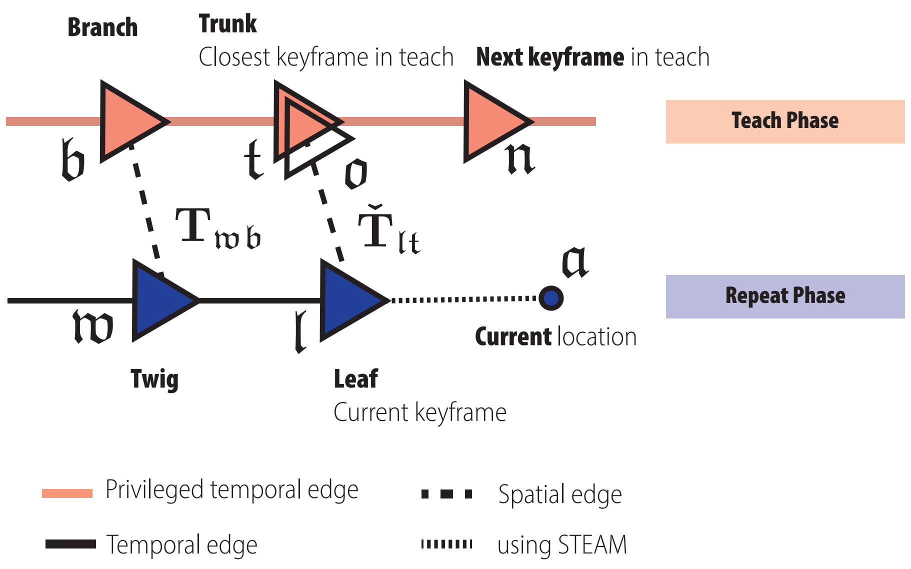


Figure 6.3: Overview of VT&R localization chain: As the vehicle moves, at every step of VO, the localization chain is updated with the estimated transform from trunk to leaf: $\check{\mathbf{T}}_{\mathfrak{lt}} = \mathbf{T}_{\mathfrak{lw}}\mathbf{T}_{\mathfrak{wb}}\mathbf{T}_{\mathfrak{bt}}$.

- leaf $\mathfrak{l}$ - current keyframe.
- trunk $\mathfrak{t}$ - spatially closest vertex from current position on privileged path.

At every step of VO, the localization chain is updated with the estimated transform from trunk to leaf $\check{\mathbf{T}}_{\mathfrak{lt}}$. Upon insertion of a new vertex (from VO) at the current keyframe (leaf $\mathfrak{l}$) in the graph, SURF feature matching is performed with respect to the trunk $\mathfrak{t}$. The raw feature matches are passed through RANSAC to obtain a set of *localization inliers* (or *inliers*). This is used in a MLESAC robust estimator to estimate the relative transform from trunk to leaf $\mathbf{T}_{\mathfrak{lt}}$. If the number of *localization inliers* drops below some threshold, a localization failure occurs as a poor estimate of the relative transform will be obtained. After the transform is optimized, the localization chain is updated and the branch and twig are re-set: $\mathbf{T}_{\mathfrak{wb}} \leftarrow \mathbf{T}_{\mathfrak{lt}}$.

### 6.3.3 Gimbal Control

Use of a gimbal decouples the camera orientation from the roll/pitch-to-move actuation of the multirotor UAV. This improves the robustness of VT&R on a multirotor, see [Patel et al., 2019]. During *repeat* trials the gimbal control is designed to reduce the orientation mismatch between trunk and leaf.

### 6.3.4 Path-following Control

In this thesis, we consider the *controller implemented during repeat.* The high-rate real-time pose (or higher-dimensional state), at the current time $\mathfrak{a}$, used by the controller is determined by extrapolating from the last VO keyframe, i.e., the leaf $\mathfrak{l}$, forward using Simultaneous Trajectory Estimation And Mapping (STEAM). From STEAM, we obtain the transform from the trunk $\mathfrak{t}$ to the current frame $\mathfrak{a}$: $\check{\mathbf{T}}_{\mathfrak{at}}$.

STEAM was first introduced in [Anderson et al., 2015] and details on the implementation used in this thesis can be found in [Wong et al., 2020(a)] and [Wong et al., 2020(b)]. 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. We obtain our controller state at the current time, associated with frame $\mathfrak{a}$, with respect to a gravity-aligned inertial $\mathfrak{o}$ frame with origin placed at the trunk $\mathfrak{t}$ (see Fig. 6.3). From the onboard IMU, we can obtain the orientation $\mathbf{C}_{\mathfrak{ot}}$ from the trunk to the inertial frame and consequently $\mathbf{T}_{\mathfrak{ot}}$. We can then estimate the transform from the current frame $\mathfrak{a}$ to the inertial frame $\mathfrak{o}$ as $\check{\mathbf{T}}_{\mathfrak{oa}} = \mathbf{T}_{\mathfrak{ot}}\check{\mathbf{T}}_{\mathfrak{at}}^{-1}$.

From this we extract our current position $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}} = (x, y, z)$[1] and yaw $\psi$ in the inertial frame. Similarly, we transform our speed in the trunk $\mathbf{v}_{\mathfrak{t}}^{\mathfrak{at}}$ obtained from STEAM into the inertial frame $\mathbf{v}_{\mathfrak{o}}^{\mathfrak{ao}} = (\dot{x}, \dot{y}, \dot{z})$. From the IMU, we obtain our acceleration $\mathbf{a}_{\mathfrak{o}}^{\mathfrak{ao}} = (\ddot{x}, \ddot{y}, \ddot{z})$ in the inertial frame. We use the following notation for state $\mathbf{z}$ and output $\mathbf{y}$:

$$\mathbf{z} = (\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}}, \mathbf{v}_{\mathfrak{o}}^{\mathfrak{ao}}, \mathbf{a}_{\mathfrak{o}}^{\mathfrak{ao}}, \psi), \tag{6.1}$$

$$\mathbf{y} = (\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}}, \psi). \tag{6.2}$$

**Baseline PD Control:** We present a baseline *PD Control* that runs at 50 Hz. We generate a path by connecting a straight-line through successive privileged vertices in the teach path. We use the localization chain to obtain a transform $\mathbf{T}_{\mathfrak{tn}}$ from the next privileged vertex $\mathfrak{n}$ to the trunk $\mathfrak{t}$. From this we obtain the position $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}}$ of the next privileged vertex $\mathfrak{n}$ with respect to the inertial frame $\mathfrak{o}$ using:

$$\mathbf{T}_{\mathfrak{tn}} = \begin{bmatrix} \mathbf{C}_{\mathfrak{tn}} & \mathbf{p}_{\mathfrak{t}}^{\mathfrak{nt}} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

[1]Notation: $\mathbf{p}_{\mathbf{c}}^{\mathbf{ab}}$ denotes a vector from $b$ to $a$ expressed in frame $c$. The transformation $\mathbf{T}_{\mathbf{ab}}$ represents the pose of $b$ with respect to frame $a$.

and $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}} = \mathbf{C}_{\mathfrak{ot}}\mathbf{p}_{\mathfrak{t}}^{\mathfrak{nt}}$. At each time step, we determine the reference position $\mathbf{p}_{\text{ref}} = (x_{\text{ref}}, y_{\text{ref}}, z_{\text{ref}})$ by projecting our current multirotor position $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}}$ onto the straight-line segment connecting the trunk and the next privileged vertex using:

$$\mathbf{p}_{\text{ref}} = \mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}} \cdot \mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}} \frac{\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}}}{|\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}}|}.$$

We obtain a reference velocity $\mathbf{v}_{\text{ref}} = (\dot{x}_{\text{ref}}, \dot{y}_{\text{ref}}, \dot{z}_{\text{ref}})$, where the magnitude is a user-selected parameter $s_{\text{des}}$, in the direction of the next privileged vertex using:

$$\mathbf{v}_{\text{ref}} = s_{\text{des}} \frac{\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}}}{|\mathbf{p}_{\mathfrak{o}}^{\mathfrak{nt}}|}.$$

Our path-following control is designed to send commands $(\dot{z}_{\text{cmd}}, \dot{\psi}_{\text{cmd}}, \theta_{\text{cmd}}, \phi_{\text{cmd}})$ where $\dot{z}_{\text{cmd}}$ is a commanded $z$-velocity, $\dot{\psi}_{\text{cmd}}$ is a command yaw rate, and $\theta_{\text{cmd}}$ and $\phi_{\text{cmd}}$ are commanded pitch and roll, respectively. The $z$-velocity command is designed using a PD controller:

$$\dot{z}_{\text{cmd}} = \frac{2\zeta_z}{\tau_z}(z_{\text{ref}} - z) + \frac{1}{\tau_z^2}(\dot{z}_{\text{ref}} - \dot{z}), \tag{6.3}$$

where $\zeta_z$ is the damping ratio parameter and $\tau_z$ is the time constant parameter. A P controller (with tuned time constant $\tau_\psi$) is used to correct for any yaw-mismatch between the current frame and the trunk:

$$\dot{\psi}_{\text{cmd}} = \frac{1}{\tau_\psi}(\psi_{\text{ref}} - \psi), \tag{6.4}$$

where $\psi_{\text{ref}}$ is the yaw of the trunk $\mathfrak{t}$ relative to the inertial frame $\mathfrak{o}$ obtained from $C_{\mathfrak{ot}}$. Lateral-motion control commands are determined by first designing translational acceleration commands using PD controllers:

$$a_x = \frac{2\zeta_\theta}{\tau_\theta}(x_{\text{ref}} - x) + \frac{1}{\tau_\theta^2}(\dot{x}_{\text{ref}} - \dot{x}), \tag{6.5a}$$

$$a_y = \frac{2\zeta_\theta}{\tau_\theta}(y_{\text{ref}} - y) + \frac{1}{\tau_\theta^2}(\dot{y}_{\text{ref}} - \dot{y}), \tag{6.5b}$$

where $\zeta_\theta$ and $\tau_\theta$ are tuned damping ratio and time constant parameters. Assuming small lateral acceleration ($\ddot{x} \approx \ddot{y} \approx 0$) and using standard feedback linearization,

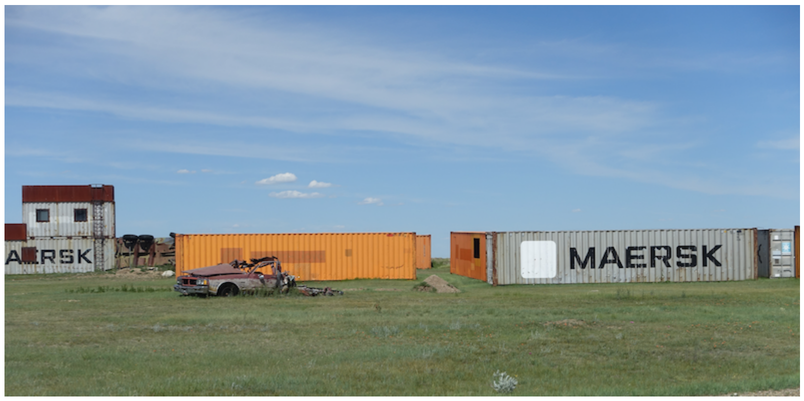


Figure 6.4: Suffield field testing location.

these linear acceleration commands are transformed into pitch and roll commands:

$$\theta_{\text{cmd}} = \arcsin(\frac{a_x}{g}\cos\psi + \frac{a_y}{g}\sin\psi), \tag{6.6a}$$

$$\phi_{\text{cmd}} = -\arcsin(-\frac{a_x}{g}\sin\psi + \frac{a_y}{g}\cos\psi), \tag{6.6b}$$

where $g$ is the gravitational constant.

## 6.4 Field Testing

### 6.4.1 Suffield Field Testing

The multirotor VT&R field tests performed in this section took place at a simulated village at the Defense Research and Development Canada (DRDC) Suffield Research Centre in southern Alberta, Canada. The Suffield location, see Fig. 6.4, consists of a number of shipping containers placed to emulate buildings and narrow alleys in flat grassland.

In these field test experiments, we evaluate the performance of the full closed-loop VT&R system. GPS navigation is used during the *teach* phase and the baseline *PD Control* in Section 6.3.4 is used in VT&R to *repeat* the path. We teach the path shown in Fig. 6.5 in a clockwise direction at an altitude of 12 m AGL, before returning using vision in an anticlockwise direction at the same altitude at a target speed of 3 m/s.

In Fig. 6.5, we show three trials where VT&R was able to complete the return phase of flight under path-following control over an approximately 2.5 minute period. The outbound path under GPS control is shown in dashed blue, while the return path under path-following control is shown in light blue, green and red for trials T1, T2

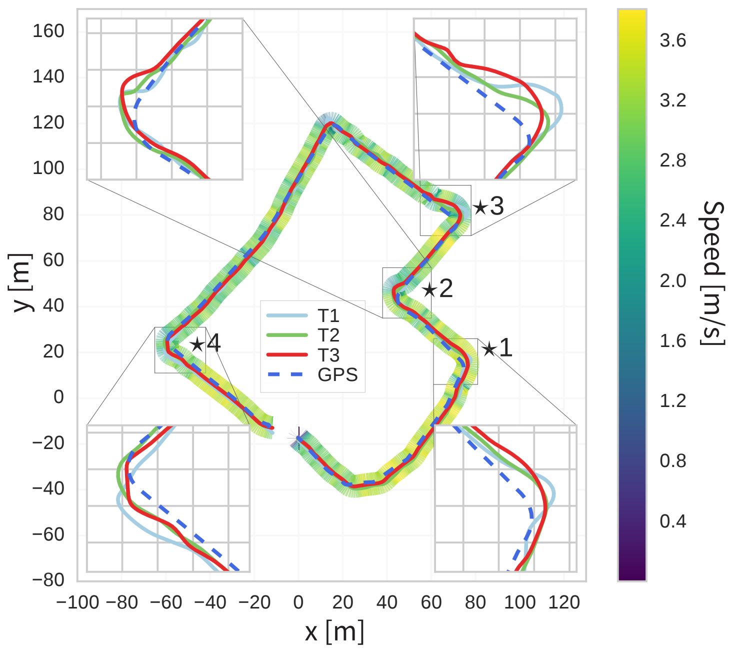


Figure 6.5: Visualization of the outbound path flown under GPS (dashed blue) and returned path using autonomous vision-based navigation for three separate trials (light blue, green, red). Some offset is seen on sharp turns. Velocity profile for trial T3 overlaid.

and T3, respectively.

Fig. 6.6 shows the number of *localization inliers* along the path for each repeat trial. For comparison, we also show the number of *localization inliers* when repeating the path under GPS control. In Fig. 6.7 we show the path error (estimated using vision) for the three vision-based autonomous trials T1, T2 and T3, and a return flight under GPS (in blue). Specific segments of the path are highlighted in Fig. 6.5 and annotated with numbers that correlate to those in Fig. 6.6 and Fig. 6.7.

For all three vision-based autonomous trials the positional error in Fig. 6.7 is less than 1.5 m over most of the path and is comparable to a return trajectory under GPS control. However, at the corners, the path error increases to a maximum of 4.5 m. Correspondingly, as observed in Fig. 6.6, the number of *localization inliers* obtained during visual localization drops dramatically at these corners.

## 6.4.2 Montreal Field Testing

The multirotor VT&R field tests performed in this section took place at a silo in downtown Montreal, see Fig. 6.8. This environment is used to demonstrate the application of autonomous vision-based navigation for inspection and patrol tasks.

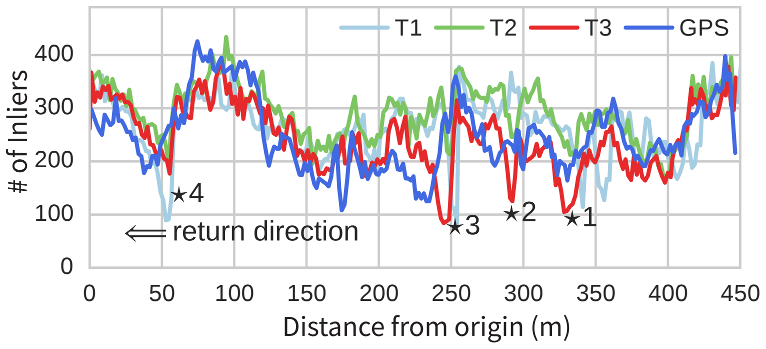


Figure 6.6: Localization inliers (# of Inliers) for three trials (light blue, green, red) using autonomous vision-based navigation. The number of localization inliers while using our baseline *PD Control* is of a similar order to that for GPS-based control (blue) over the majority of the path.

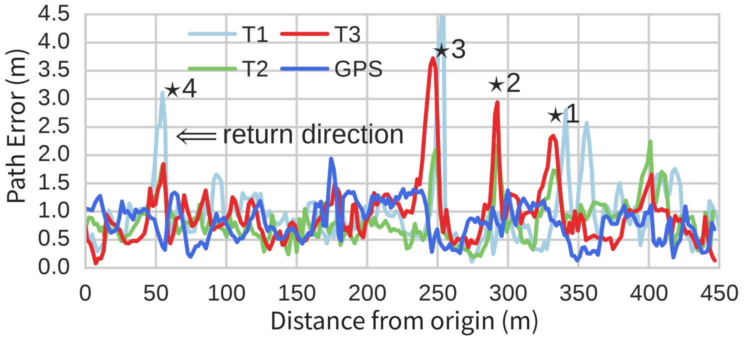


Figure 6.7: Path error (estimated from vision) for three trials (light blue, green, red) using autonomous vision-based navigation. The path error (determined from the localization transform estimated by VT&R) is of a similar order to that for GPS-based control (blue) over the majority of the path using our baseline *PD Control*.

Over three separate trials (each consisting of a single *teach* and then *repeat* flight), we traverse the path shown in Fig. 6.9 at 2 m/s. Unlike at Suffield, we test variations in altitude up to 30 m above our initial height of 10 m above ground level (AGL). We demonstrate that our baseline *PD Control* is able to operate with elevation changes but large path errors are similarly experienced when sharp turns are needed.

## 6.5 Challenges to Autonomous Visual Navigation

In this section, we highlight two distinct challenges to high speed autonomous vision-based navigation. In the field tests in Section 6.4, we demonstrated the viability of the VT&R system in two different environments at lower speed autonomous flight

Figure 6.8: Montreal field testing location.

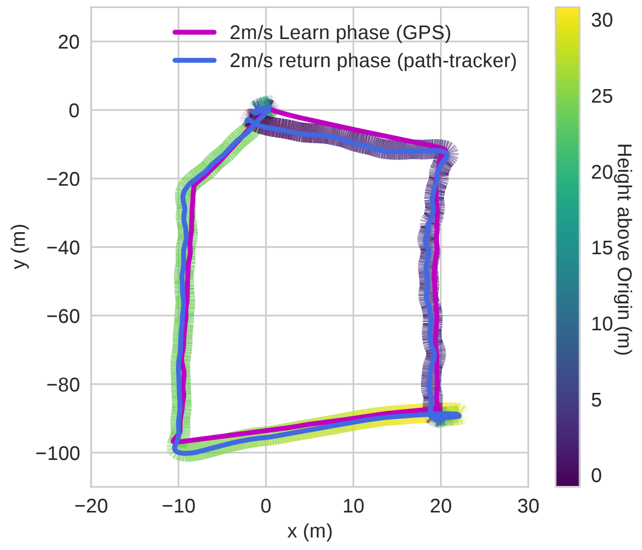


Figure 6.9: *Montreal Field Testing*: The outbound *teach* (GPS, magenta) and *repeat* (with baseline *PD Control*, blue) paths during a single trial. At low speeds (2 m/s) control-in-the-loop performs for an elevation change of 30 m.

(less that 3 m/s). We showed that our baseline *PD Control* performed well at these speeds and on straight paths. However, at sharp turns it produced a large error as it did not predict for the change in the curvature of the path. Consequently, this lead to a drop in localization performance (i.e., the number of *localization inliers*) at these sharp turns.

Model predictive control (MPC), like that presented in Chapter 3, is used to combine predictive planning and control to achieve high performance around sharp turns while accounting for constraints. Standard MPC is often perception-agnostic, i.e., it is agnostic to the perception module and limitations. There are two important requirements to consider in the implementation of MPC in vision-based navigation:

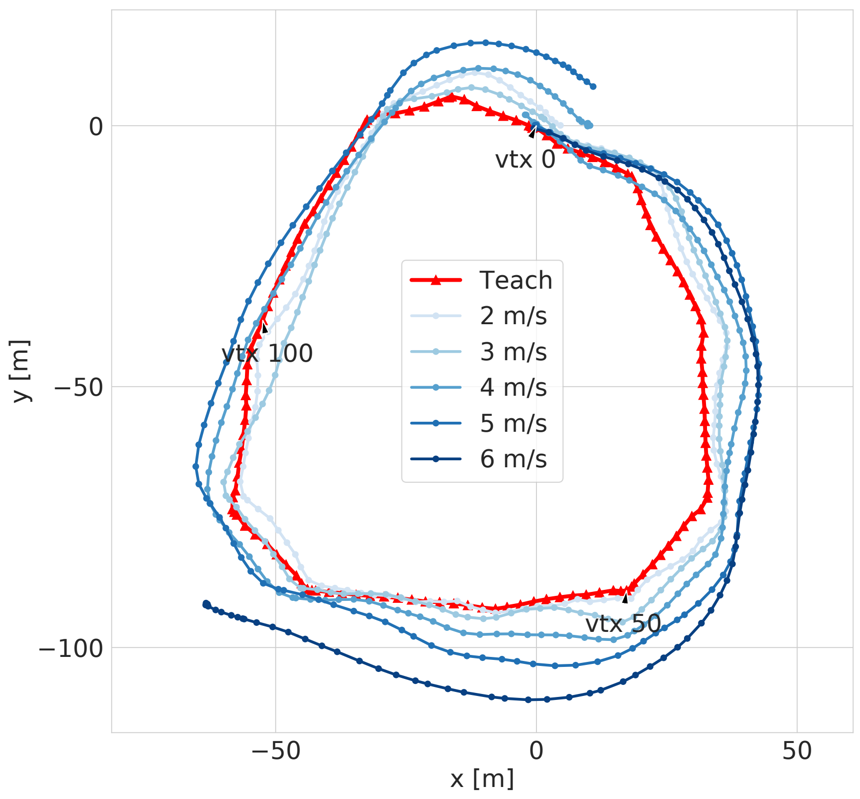


Figure 6.10: Visualization of a circular *teach* path (red) at altitude 20 m AGL and *repeat* trials with increasing desired speeds $s_{\text{des}}$ (blues) using the baseline *PD Control* for autonomous vision-based navigation.

- MPC requires specifying constraints. These constraints often represent physical barriers for safety. However, constraints that represent the limitations of the perception module are often neglected.

- MPC often relies on accurate state estimation to make optimal predictions.

These requirements underpin two limitations to the use of perception-agnostic controllers in vision-based navigation:

- Limitation 1 - Partial Environment Knowledge: They assume that the commanded action computed by the controller has no effect on the ability of vision-based navigation to determine the UAV's location (or localization).

- Limitation 2 - Imperfect State Estimation: They rely on perfect state estimation from vision-based navigation.

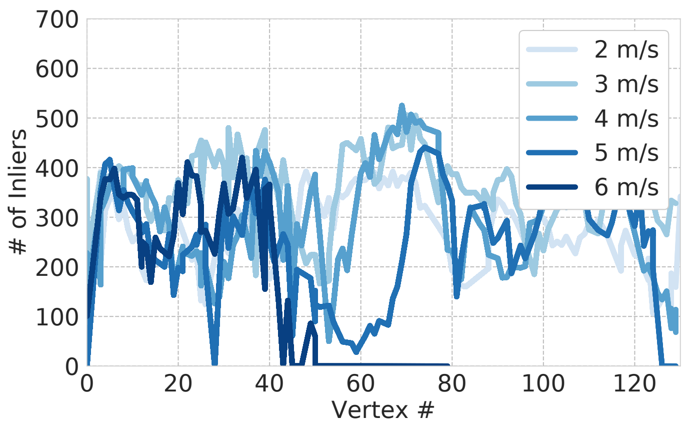


Figure 6.11: Localization inliers (# of Inliers) for various speeds of *repeat* trials of the circular *teach* path. At 6 m/s, localization is lost as the cross-track error gets too large.

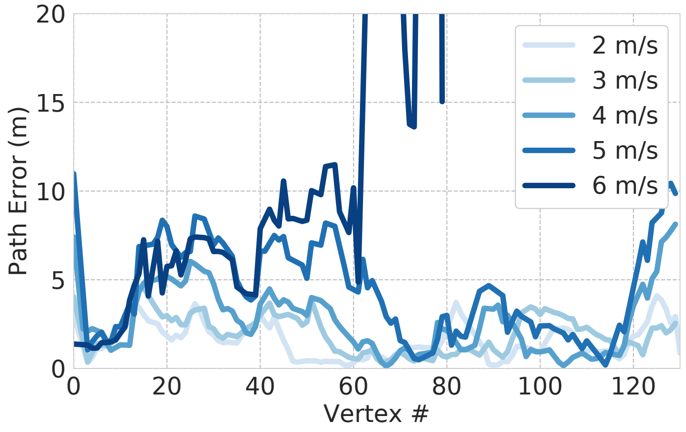


Figure 6.12: Path error (estimated from vision) for various speeds of *repeat* trials of the circular *teach* path. At 6 m/s, localization is lost as the cross-track error gets too large.

### 6.5.1 Partial Environment Knowledge

Most controllers, including our baseline *PD Control*, are perception-agnostic and neglect the effect of their actions on the visual localization capabilities of the system. This becomes especially relevant around sharp turns where there is often a trade-off between high speed and allowable path error. An incorrect trade-off highly prioritizing low path error leads to suboptimal slower flight while an incorrect trade-off highly prioritizing fast flight can lead to a path error that results in localization failure by the visual system. Therefore, the controller can produce an action (e.g., to prioritize speed) that leads to a location where we cannot localize well with respect to the map (i.e., poor location estimate from vision). Poor performance of either control or

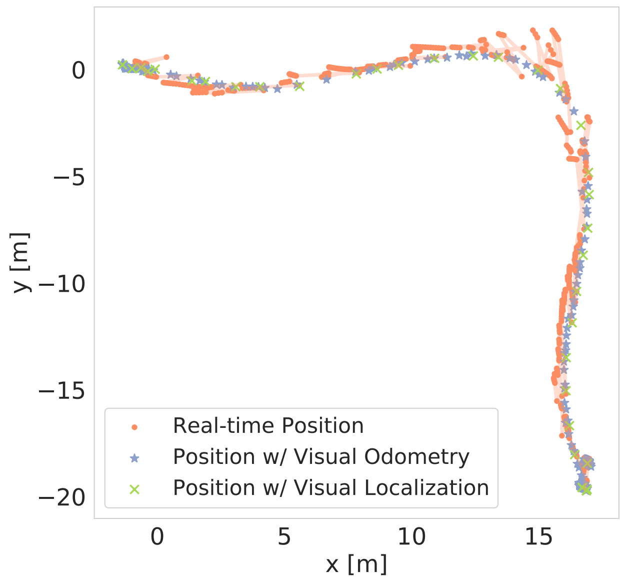


Figure 6.13: Comparison of the estimated noisy real-time position at 50 Hz (red), the estimated position from VO and bundle adjustment (blue) and including visual localization (green) for a path flown at 9 m/s.

perception is reinforced through the feedback loop between these subsystems.

We perform a simple speed test of the baseline *PD Control* at UTIAS of a circular *teach* path (red) shown in Fig. 6.10. This test was done at low wind speeds at a constant altitude of 20 m AGL. As expected, as we increase the speed during the *repeat* trials, the path error increases in Fig. 6.12 and, consequently, the number of *localization inliers* decreases in Fig. 6.11. At a desired speed of $s_{\text{des}} = 6$ m/s, the number of *localization inliers* is too low and we are no longer able to relocalize.

The challenge is that VT&R relies on partial knowledge of the environment. In VT&R, prior knowledge of the environment is limited to a visual-map created during the teach phase. Therefore, a sufficient path error with the *teach* visual-map may cause visual localization to fail. When visual localization fails, we may not be able to obtain a good position estimate.

In Chapter 7, we address this challenge of partial environment knowledge and present a perception-aware model predictive controller that accounts for its effect on visual localization in a VT&R framework.

## 6.5.2 Imperfect State Estimation

The challenge is that controllers often have to rely on noisy high-rate state estimation as an accurate full-state estimate is often challenging due to typically noisy IMU measurements, an infrequent position update from the vision system and an imperfect motion model in STEAM used to obtain high-rate state estimates required by control.

In Fig. 6.13 we show 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. 6.13 for sharp turns at higher speeds STEAM does not accurately estimate the trajectory.

In Chapter 8, we address this challenge of requiring accurate full-state estimation by presenting an alternative approach that exploits discrete-time flatness to avoid inaccurate velocity and acceleration estimates and instead relies on only a window of outputs, specifically position and yaw, and previously sent inputs.

# Chapter 7

# Perception-Aware Control

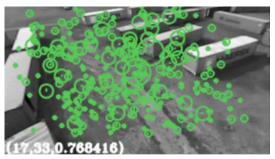
**Planning for Partial Environment Knowledge using a Perception Model in Control**

## 7.1 Overview and Related Work

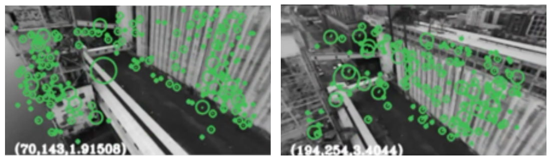
In recent years, inspired in part by the DARPA FLA challenge, see [DARPA], and the potential for autonomous drone racing, see [Kaufmann et al., 2019], there has been a significant push toward fast vision-based multirotor unmanned aerial vehicle (UAV) flight, see, for example, [Beul et al., 2018] or [Mohta et al., 2018]. One such vision-based approach uses a Visual Teach and Repeat (VT&R) framework that allows the UAV to repeat a previously taught path by matching current visual features to those in the locally metric map created during teach, see [Warren et al., 2019] or [Gao et al., 2019].

Most of these vision-based navigation systems still rely on perception-agnostic control techniques. For example, a conventional model predictive control (MPC) computes a control input by optimizing a cost function over a prediction horizon but ignores the effect of its planning on the visual localization capabilities of the system. This becomes especially relevant around sharp turns where there is often a trade-off between high speed and allowable path error. An incorrect trade-off highly prioritising low path error leads to suboptimal slower flight while an incorrect trade-off highly prioritising fast flight can lead to a path error that results in a localization failure by the visual system.

This trade-off between fast and reliable flight is often addressed by careful tuning or by adding a fixed allowable path error constraint, see [Ostafew et al., 2016]. How-



(a) Example images from *Suffield dataset*



(b) Example images from *Montreal dataset*

Figure 7.1: Localization inliers (green) associated with successfully matched landmarks between teach and repeat.

ever, because this trade-off is visual-environment dependent, these approaches do not capture variability of this trade-off along a path and they have to be retuned in every new visual environment. Consequently, they tend to lead to suboptimal speeds or unreliable vision-based navigation.

Our aim is to implement a *perception-aware controller, that accounts for its effect on visual localization in a VT&R framework and to demonstrate its ability to achieve reliable but fast vision-based flight* compared to conventional perception-agnostic controllers in outdoor experiments.

### 7.1.1 Perception-Aware Control Design

Similar to [Falanga et al., 2018], we adopt a perception-aware MPC strategy. In [Falanga et al., 2018], by using additional perception error terms in the MPC cost function, a multirotor UAV with static camera was able to follow a trajectory while tracking a single visual target. Unlike this approach, we are not limited by a multirotor with static camera combination as we decouple the camera orientation from the multirotor UAV using a gimbal, see [Warren et al., 2019]. However, instead of a single target, there are features associated with many potential landmarks (observed during teach) that we could match and use to localise. A simple strategy of pointing the camera toward the centroid of these landmarks, see [Patel et al., 2019], cannot overcome viewpoint changes as a result of UAV path error.

Another key difference, that avoids additional cost function tuning, is that in this chapter we choose to treat the limitations of our visual perception as a constraint in the MPC formulation. The problem then becomes: *How do we successfully integrate our visual perception limits as a constraint online into a real-time MPC?*

### 7.1.2 Modelling Perception Limits

Similar to [Churchill et al., 2015] and [Gurau et al., 2017], we aim to model the limits of perception performance in a VT&R framework by generating an area around the *teach* path where localization is probabilistically guaranteed (coined *localization envelope* in [Churchill et al., 2015]). Unlike these approaches, we close the control loop around vision by developing a perception model that is used in a constraint in real-time MPC. Unlike early work in [Furgale et al., 2010], the authors in [Churchill et al., 2015] developed a *localization envelope* that can vary at different points along the *teach* path. They did this by modelling the likely number of feature matches around the *teach* path, where some low number of feature matches embodies the boundary of the *localization envelope.* This is modelled using a Gaussian Process (GP), which takes as input the position relative to the path, the associated closest point on the teach path and the localiser's performance during multiple *repeat* runs. The authors propose *repeat* trials that deviate from the path until failure. This is both time consuming and potentially unsafe. In [Gurau et al., 2016] and [Gurau et al., 2017], the work in [Gurau et al., 2016] is extended to similar (but not necessarily the same) *teach* paths by also utilizing curvature and an appearance model in the GP. Both approaches are limited by a place-dependent model to assess localization performance. Both approaches are limited by a place-dependent model to assess localization performance. To overcome this limitation, our proposed approach instead creates a simple but conservative global *localization envelope* model that does not have to be retrained for every new path. We do this by incorporating scene structure through associating landmarks seen in teach with a probability of being matched.

### 7.1.3 Contributions

There are three points of novelty to the proposed perception-aware MPC presented in this chapter:

- We develop and validate a simple geometric perception model (for nominal lighting conditions - i.e., little difference in lighting conditions between teach and repeat) using over 12 km of data for our visual localization system in [Warren et al., 2019].

- We show how to integrate this perception model in a chance constraint, such that localization is guaranteed, in our MPC and how to convert it to a deter-

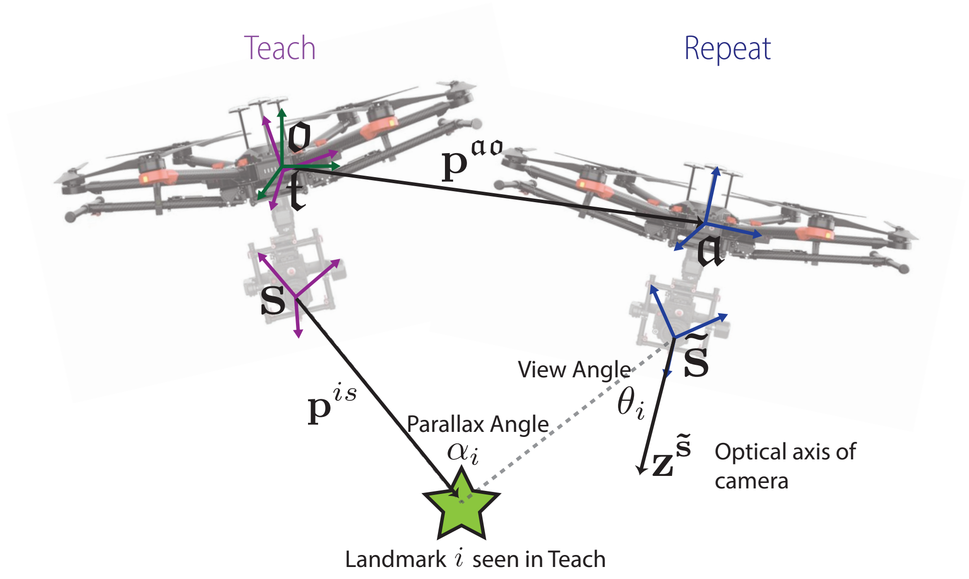


Figure 7.2: Geometric angles used to model of the probability $p_i$ that a landmark $i$ becomes a *localization inlier*. *Parallax angle* $\alpha_i$ captures the perspective change as a result of path error between teach and repeat while *view angle* $\theta_i$ captures whether the landmark is still visible.

ministic nonlinear constraint.

- Using real-world perception data, we provide experimental simulation results demonstrating the value of our perception-aware MPC in terms of reliably but optimally self-regulating speed along a path compared to a similar perception-agnostic control.

## 7.2 Problem Statement

Consider a multirotor with a continuous-time, nonlinear model of the form:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad t \in \mathbb{R}^+,$$

described by (3.4a)-(3.4c), (3.5) and (3.6a)-(3.6b) with state $\mathbf{x} = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{z}, \phi, \theta, \psi)$ and input $\mathbf{u} = (\dot{z}_{\text{cmd}}, \phi_{\text{cmd}}, \theta_{\text{cmd}}, \dot{\psi}_{\text{cmd}})$. Given a teach path and some user-defined desired speed $s_{\text{des}}$ by which to repeat the path, determine an optimal control problem (OCP) for real-time MPC that can be used to compute an input $\mathbf{u}(t)$ such that the following is achieved:

- We achieve high-speed flight, i.e., we track our desired speed such that $|s - s_{\text{des}}|$

remains small where $s = ||[\dot{x}, \dot{y}, \dot{z}]^T||$.

- We guarantee that we can successfully localize using visual perception. In our VT&R navigational system, the success of our visual localization during repeat is associated with the number of localization inliers. When the number of localization inliers $L$ is below some threshold $L_{\text{min}}$ we do not have a reliable estimate of our position. We, therefore, choose to represent the limitations of our visual perception by a perception failure chance constraint where the probability of the number of inliers going below our determined threshold remains very low. Specifically, we can write this perception constraint as:

$$\Pr(L < L_{\text{min}}) \leq \delta, \tag{7.1}$$

  where $\delta$ is small.

## 7.3 Methodology

We approach the problem in three steps:

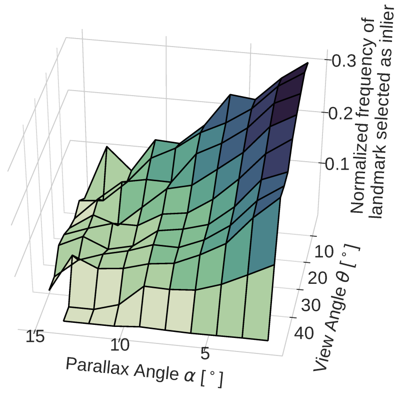
**Geometric Perception Model:** By treating each landmark from the closest teach frame (trunk) independently, we develop a simple geometric model to associate a landmark with a probability of being matched successfully and becoming a localization inlier. This is based on significant previous data (in nominal lighting conditions).

**Perception Chance Constraint:** Using this probability model, we transform our perception failure chance constraint from (7.1) into a deterministic but nonlinear constraint on the position of the multirotor UAV.

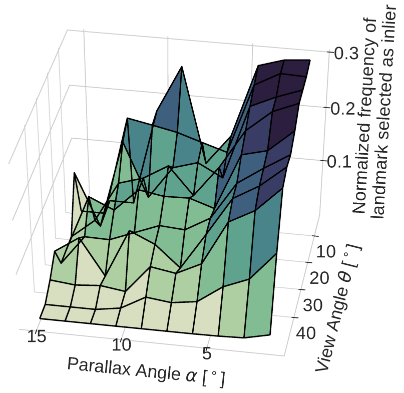
**Perception-Aware MPC:** In a method similar to standard nonlinear MPC approaches, we linearize this constraint at each time step about our previously predicted trajectory and landmarks and perform 1 iteration of a sequential quadratic program (SQP) online.

### 7.3.1 Geometric Perception Model

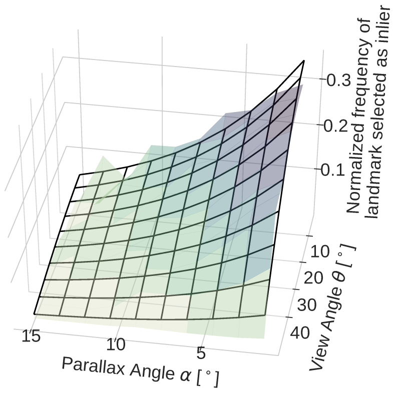
At the current position, we consider all landmarks from the trunk (closest in *teach* path) that we could potentially successfully match and could become *localization*



(a) Suffield dataset



(b) Montreal dataset



(c) Best model fit on combined dataset

Figure 7.3: Normalized frequency of landmark selected as inlier vs **parallax angle** $\alpha$ & **view angle** $\theta$ shown for (a) *Suffield dataset*, (b) *Montreal dataset* and (c) the combined dataset. In (c) a best fit model $c_1 \cos(c_2\theta)\exp(c_3\alpha)$ is overlayed in black where for optimal parameters $(c_1, c_2, c_3) = (0.424, 0.028, -0.145)$, a sufficient fit was achieved ($R^2 = 0.92$).

*inliers*. Our aim is to associate each landmark $i$ with a probability of being an inlier. Specifically, we treat each landmark $i$ as an *independent Bernoulli random variable* $l_i$ with some probability $p_i$ of being an inlier, that is:

$$l_i \sim \mathrm{Ber}(p_i). \tag{7.2}$$

Our aim is to develop a simple perception model that can be used to estimate the probability $p_i$ for each landmark $i$. Let $\mathbf{p}_{\mathfrak{o}}^{is}$ be the vector from the trunk sensor $s$ to landmark $i$ described in the inertial frame $\mathfrak{o}$. As shown in Fig. 7.2, we develop this perception model as a function of two geometric angles.

**Parallax Angle:** $\alpha_i$ is the angle between the rays from landmark $i$ to the camera sensors $s$ and $\tilde{s}$ as a result of the positional offset between the current frame and trunk. This angle captures the perspective change, i.e., how much the landmark has moved in the scene when reviewing it. It is computed as a function of the position offset $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{a}\mathfrak{o}}$ and is independent of the camera orientation:

$$\alpha_i = \angle(-\mathbf{p}_{\mathfrak{o}}^{is}, -\mathbf{p}_{\mathfrak{o}}^{is} + \mathbf{p}_{\mathfrak{o}}^{\mathfrak{a}\mathfrak{o}}). \tag{7.3}$$

**View Angle:** $\theta_i$ is the angle between the landmark $i$ and the optical axis of the current sensor $\tilde{s}$. It captures both the effect of the field-of-view of the camera and potential degradation at the edges of the image. Let $\mathbf{z}_{\mathfrak{o}}^{\tilde{s}}$ be the vector representing the current sensor optical axis described in the inertial frame $\mathfrak{o}$. We can compute the $\theta_i$ as:

$$\theta_i = \angle(\mathbf{z}_{\mathfrak{o}}^{\tilde{s}}, \mathbf{p}_{\mathfrak{o}}^{is} - \mathbf{p}_{\mathfrak{o}}^{\mathfrak{a}\mathfrak{o}}). \tag{7.4}$$

We use teach and repeat paths from two datasets to identify the perception model. *Montreal dataset*: We use 6 trials of a teach followed by a repeat in an urban environment near downtown Montreal, Canada, see Fig. 6.8. *Suffield dataset*: We use 6 trials of a teach followed by a repeat at a rural environment in Alberta, Canada, see Fig. 6.4. Fig. 7.1 shows examples of images observed. For all trials, for each landmark $i$ in the trunk we compute the associated angles $\alpha_i$ and $\theta_i$ (as a result of position and camera offset) and we mark whether the landmark was selected as a *localization inlier* or not. For each angle range (i.e., some $\alpha$ and $\theta$), we compute the normalized frequency of a landmark being selected as an inlier by dividing the total number of inliers by the total number of landmarks seen. We show the results for the Montreal dataset, Suffield dataset and then the combined dataset in Fig. 7.3.

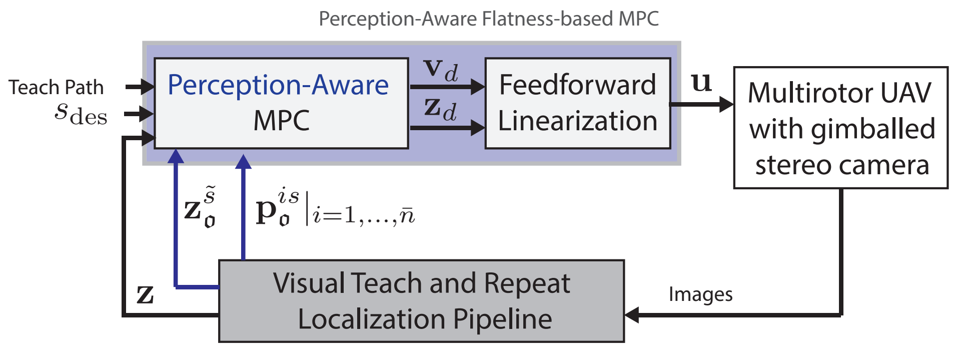

Perception-Aware Flatness-based MPC
Teach Path
$s_{\text{des}}$
Perception-Aware MPC
$\mathbf{v}_d$
$\mathbf{z}_d$
Feedforward Linearization
$\mathbf{u}$
Multirotor UAV with gimballed stereo camera
$\mathbf{z}_{\mathfrak{o}}^{\tilde{s}}$
$\mathbf{p}_{\mathfrak{o}}^{is}|_{i=1,\ldots,\bar{n}}$
$\mathbf{z}$
Visual Teach and Repeat Localization Pipeline
Images

Figure 7.4: Block diagram of proposed Perception-Aware Flatness-Based Model Predictive Control used during repeat to fly at some desired speed $s_{\text{des}}$. Including perception-awareness into MPC requires additional inputs: 1) landmarks $\mathbf{p}_{\mathfrak{o}}^{is}$ from the trunk (closest vertex in the teach path) and 2) the current camera orientation $\mathbf{z}_{\mathfrak{o}}^{\tilde{s}}$. The Perception-Aware MPC involves solving an optimal control problem, for example, (3.8), with an additional constraint (7.8).

We consider a frequentist approach and associate the determined normalized frequency based on our data as a probability of that landmark being an inlier. We then fit the model:

$$p_i = c_1 \cos(c_2\theta_i)\exp(c_3\alpha_i) \tag{7.5}$$

where $p_i$ is the probability of being an inlier, $\theta_i$ is the view angle and $\alpha_i$ is the parallax angle. The best least squares fit gives parameter constant values $(c_1, c_2, c_3) = (0.424, 0.028, -0.145)$. Using (7.3) and (7.4) in (7.5), we can rewrite the probability model as a nonlinear function of the output $\mathbf{y} = (\mathbf{p}_{\mathfrak{o}}^{\mathfrak{a}\mathfrak{o}}, \psi)$ from (6.2):

$$p_i = h_L(\mathbf{y}, \mathbf{p}_{\mathfrak{o}}^{is}, \mathbf{z}_{\mathfrak{o}}^{\tilde{s}}). \tag{7.6}$$

where $\mathbf{p}_{\mathfrak{o}}^{is}$ is determined from the known landmark position (determined during teach) and $\mathbf{z}_{\mathfrak{o}}^{\tilde{s}}$ is determined using the current gimbal orientation.

### 7.3.2 Perception Chance Constraint

We consider $\bar{n}$ landmarks at the trunk where we treat each landmark $l_i$ as an independent non-identical Bernoulli trial given by (7.2). Each landmark is associated with a probability $p_i$ based on (7.5) of being an inlier. Let $L$ be the total number of successes of these trials (i.e., the total number of inliers) where $L$ has a distribution associated with the sum of these $\bar{n}$ independent, non-identical Bernoulli trials. This distribution is known as a *Poisson binomial distribution* where its first two moments can be found by summing the expectation and variance of each non-identical Bernoulli trial respectively. This gives the expectation for $L$ as:

$$\mathbb{E}(L|p_1, ..., p_{\bar{n}}) = \bar{n}\bar{p}$$

where $\bar{p} = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} p_i$ and the variance for $L$ as:

$$\mathbb{V}(L|p_1, ..., p_{\bar{n}}) = \bar{n}\bar{p}(1 - \bar{p}) - \bar{n}s_p^2$$

where $\bar{n}s_p^2 = \sum_{i=1}^{\bar{n}} (p_i - \bar{p})^2 \geq 0$. Note from this that the expectation for $L$ is an upper bound for its variance. Specifically:

$$\mathbb{V}(L|p_1, ..., p_{\bar{n}}) \leq \bar{n}\bar{p}(1 - \bar{p}) \leq \bar{n}\bar{p} = \mathbb{E}(L|p_1, ..., p_{\bar{n}}).$$

We tend to have many landmarks at the trunk (i.e. $\bar{n}$ tends to be around 600 - 800). By using *Lyapanov's Central Limit Theorem*, see Theorem 7 below, as justification, we make a normal approximation for the Poisson binomial distribution $L$, that is $L \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = \mathbb{E}(L|p_1, ..., p_{\bar{n}})$ and $\sigma^2 = \mathbb{V}(L|p_1, ..., p_{\bar{n}})$ as given by the expressions above. More specifically, we apply Lyapanov's Central Limit Theorem (CLT) in Theorem 7 to our Poisson binomial distribution to show that it converges to a normal distribution in its limit.

**Theorem 7 (Lyapanov's CLT)** *Suppose $\{X_1, X_2, ..., X_{\bar{n}}\}$ are independent (not necessarily identically distributed) random variables each with a finite mean $\mu_i$ and variance $\sigma_i^2$. If for some $\rho > 0$: $\lim_{n\to\infty} \frac{1}{s_{\bar{n}}^{\rho+2}} \sum_{i=1}^{\bar{n}} \mathbb{E}(|X_i - \mu_i|^{\rho+2}) = 0$, where $s_{\bar{n}}^2 = \sum_{i=1}^{\bar{n}} \sigma_i^2$, then the Central Limit Theorem holds, i.e. $\frac{1}{s_{\bar{n}}} \sum (X_i - \mu_i) \to \mathcal{N}(0, 1)$ as $\bar{n} \to \infty$, see [Billingsley et al., 1995].*

We treat the landmarks as independent Bernoulli random variables $l_i \sim \text{Ber}(p_i)$ where the first two moments of each variable, $\mu_i = p_i$ and $\sigma_i^2 = p_i(1 - p_i)$, are finite. We can show that Theorem 7 holds by, for example, selecting $\rho = 2$. We use the fact that for a Bernoulli random variable $\mathbb{E}(X_i^k) = \mathbb{E}(X_i) = p_i, \quad \forall k$, to show:

$$\mathbb{E}(|X_i - p_i|^4) = p_i(1 - p_i) - 3p_i^2(p_i - 1)^2 \leq p_i(1 - p_i) = \sigma_i^2.$$

Therefore, $\frac{1}{s_{\bar{n}}^4} \sum_{i=1}^{n} \mathbb{E}(|X_i - p_i|^4) \leq \frac{1}{s_{\bar{n}}^2}$. We are now left to show that $\frac{1}{s_{\bar{n}}^2} \to 0$ as $\bar{n} \to \infty$. For $0 < p_i < 1$, $s_{\bar{n}}^2 = \sum_{i=1}^{\bar{n}} \sigma_i^2 = \sum_{i=1}^{\bar{n}} p_i(1 - p_i) \to \infty$ as $\bar{n} \to \infty$. Therefore, $\frac{1}{s_{\bar{n}}^2} \to 0$ as $\bar{n} \to \infty$. This shows that Lyapanov's CLT holds which justifies the normal approximation made. Using this normal approximation for $L$, we can now rewrite the

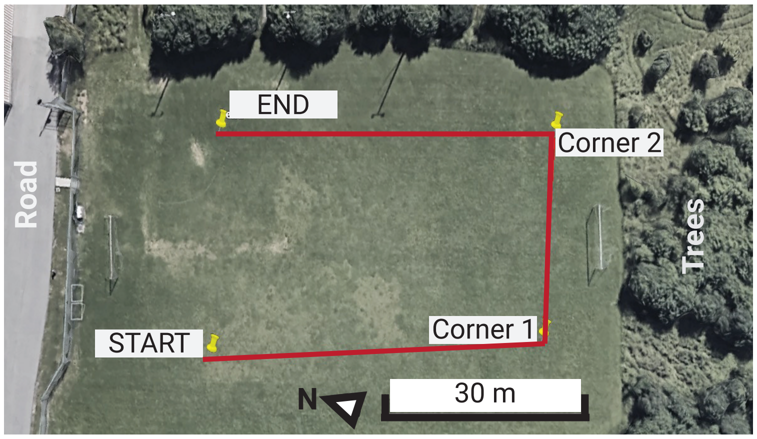


Figure 7.5: Overview of teach path (red) physically flown while looking toward the trees (Case 1) and toward the road (Case 2).

visual perception failure chance constraint from (7.1):

$$\Pr(L < L_{\min}) \leq \delta, \quad L \sim \mathcal{N}(\mu, \sigma^2),$$

as an equivalent deterministic constraint:

$$\mu - L_{\min} \geq \bar{c}\sigma,$$

where $\bar{c} = \sqrt{2}\mathrm{erf}^{-1}(1-2\delta)$ and $\mathrm{erf}^{-1}$ is the inverse error function. By using the upper bound for our variance, $\sigma^2 \leq \mu$, a more conservative constraint becomes:

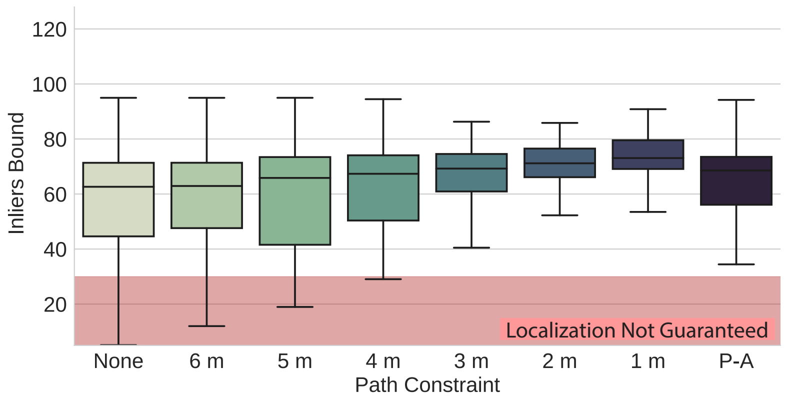
$$\mu - \bar{c}\sqrt{\mu} \geq L_{\min}. \tag{7.7}$$

By recalling that $\mu = \sum_{i=1}^{\bar{n}} p_i$ and plugging in (7.6) we can rewrite this constraint as a nonlinear constraint on position $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}}$ or output $\mathbf{y}$ in (6.2):

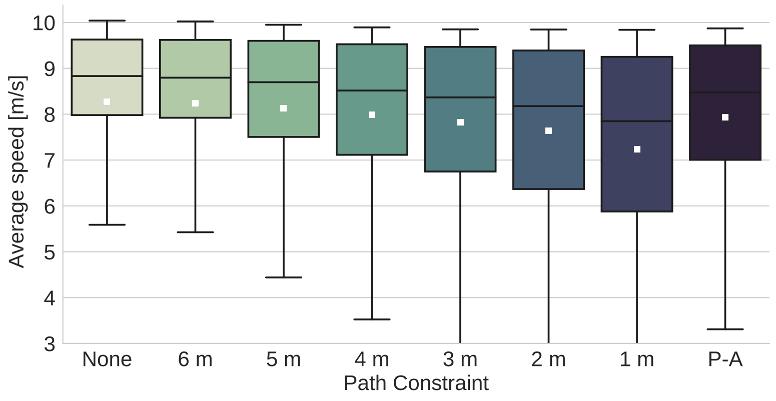
$$\sum_{i=1}^{\bar{n}} h_L(\mathbf{y}, \mathbf{p}_{\mathfrak{o}}^{is}, \mathbf{z}_{\mathfrak{o}}^{\bar{s}}) - \bar{c}\sqrt{\sum_{i=1}^{\bar{n}} h_L(\mathbf{y}, \mathbf{p}_{\mathfrak{o}}^{is}, \mathbf{z}_{\mathfrak{o}}^{\bar{s}})} \geq L_{\min}. \tag{7.8}$$

## 7.3.3 Perception-Aware Model Predictive Control

We propose using the deterministic perception constraint on the multirotor position $\mathbf{p}_{\mathfrak{o}}^{\mathfrak{ao}}$ in (7.8) in a flatness-based MPC framework, see Chapter 3 or [Greeff et al., 2018] for details. In such a framework the only nonlinearity in the proposed optimal control

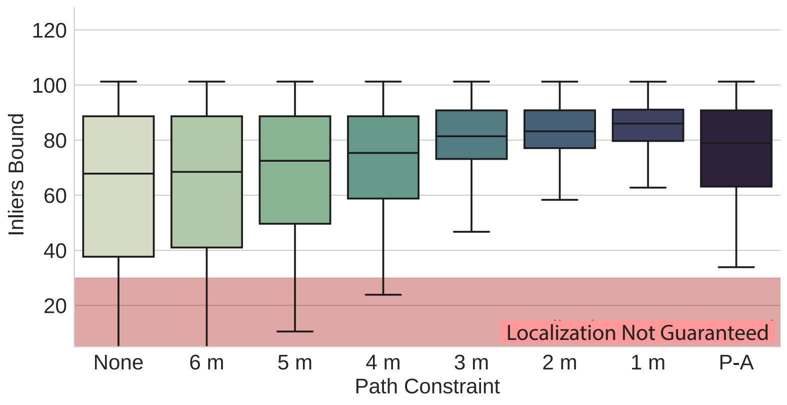


(a) Inlier Bound

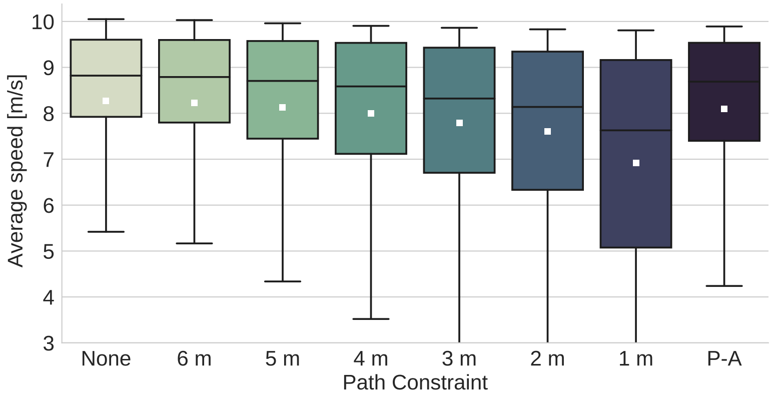


(b) Average Speed

Figure 7.6: *Case 1 - Looking Toward Trees:* (a) inlier bound for guaranteed localization ($\mu - 3\sigma$) and (b) average speed (m/s) when repeating the teach path (taught facing the camera toward the trees) with a desired speed $s_{\text{des}} = 10$ m/s using a flatness-based MPC with (i) no constraint (None), (ii) a fixed path error constraint (1 m, 2 m, 3 m, 4 m, 5 m, 6 m) and (iii) our proposed perception-aware (P-A) constraint given by (7.8).



(a) Inlier Bound



(b) Average Speed

Figure 7.7: *Case 2 - Looking Toward Road:* (a) inlier bound for guaranteed localization ($\mu - 3\sigma$) and (b) average speed (m/s) when repeating the teach path (taught facing the camera toward the road) with a desired speed $s_{\text{des}} = 10$ m/s using a flatness-based MPC with (i) no constraint (None), (ii) a fixed path error constraint (1 m, 2 m, 3 m, 4 m, 5 m, 6 m) and (iii) our proposed perception-aware (P-A) constraint given by (7.8).

problem (OCP) comes from this perception constraint. We use the current state $\mathbf{z}$ from (6.1) and output $\mathbf{y}$ from (6.2) from the vision system. In standard real-time nonlinear MPC fashion, at each time step we linearize this constraint about the previously predicted optimal trajectory and perform 1 iteration of the sequential quadratic program (SQP) online. A block diagram can be seen in Fig. 7.4.

## 7.4 Simulations

We fly the same geometric teach path in the environment shown in Fig. 7.5. However, we fly the path with the camera pointed in different (but fixed) directions. Specifically, we first fly the teach path with the camera pointing toward the trees. This is Case 1, see Fig. 7.8(a). We then fly the same teach path with the camera pointing toward the road. This is Case 2, see Fig. 7.8(b). In both cases, we keep the camera at a constant downward pitch of 50° from the horizontal.

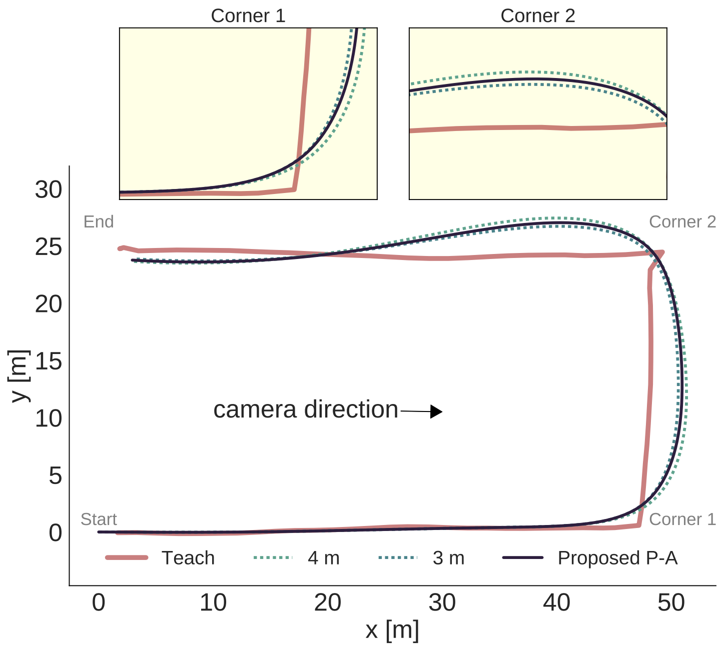
In each of these cases, we simulate the repeat path that would be flown using a flatness-based MPC (3.8) with (i) no path constraint, (ii) a fixed path constraint of 1 m, 2 m, 3 m, 4 m, 5 m and 6 m, and (iii) our proposed perception-aware (P-A) constraint governed by (7.8). In each of these cases and trials we run our outer-loop controller at 50 Hz. We implement our MPC with a discretization of 10 Hz and a 1.5 s prediction horizon (i.e. N = 15). We use the threshold for the minimum number of inliers as $L_{\text{min}} = 30$. We consider our probabilistic inlier bound to be 3 standard derivations, i.e. $\bar{c} = 3$. In other words, we guarantee probabilistically that 99.9% of the time we will maintain more than 30 inliers.

### 7.4.1 Case 1 - Looking Toward Trees

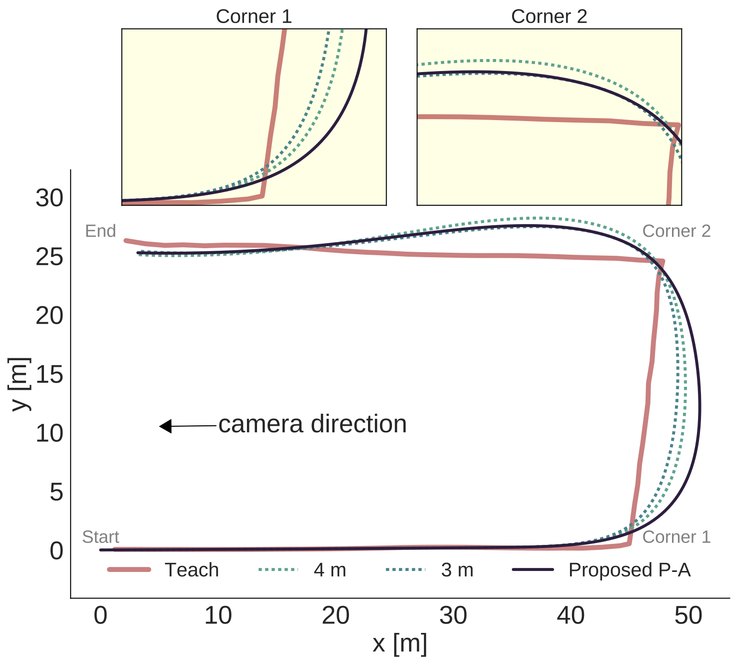
As seen in Fig. 7.6(a), we cannot guarantee localization when repeating the taught path using flatness-based MPC with no constraint (None) or with fixed path-error constraint of greater than or equal to 4 m. This is because there is some portion of our repeat path where our inlier bound ($\mu - \bar{c}\sigma$) is less than our threshold of 30 inliers. Consequently, we cannot guarantee localization (and thus to complete the path when flying under vision).

While localization is guaranteed when we simulate our repeat path using a flatness-based MPC with fixed path-error constraint of 3 m, we manage to achieve a *slightly* higher average speed (Fig. 7.6(b)) with our proposed perception-aware (P-A) constraint. As illustrated in Fig. 7.8(a), we are able to achieve a slightly higher average speed using our perception-aware constraint because we fly a profile with a path error between 3 m and 4 m. Consequently, our average speed when using our perception-aware constraint is between that flown under a fixed constraint of 3 m and 4 m in Fig. 7.6(b).

Moreover, a key benefit over using a fixed path error constraint is that we do not need to simulate (or fly) under different fixed path error constraints in order to find the constraint that ensures localizability. Instead, we explicitly guarantee it in our perception-aware constraint.



(a) Case 1 - Looking Toward Trees



(b) Case 2 - Looking Toward Road

Figure 7.8: (a) Simulated path profile of repeating taught path in Case 1 using flatness-based MPC with (i) a fixed path error constraint of 3 m, (ii) a fixed path error constraint of 4 m, (iii) our perception-aware constraint (P-A); (b) Simulated path profile of repeating taught path in Case 2 using flatness-based MPC with (i) a fixed path error constraint of 3 m, (ii) a fixed path error constraint of 4 m, (iii) our perception-aware constraint (P-A).

### 7.4.2 Case 2 - Looking Toward Road

As seen in Fig. 7.7(a), we cannot guarantee localization when repeating the taught path using flatness-based MPC with no constraint (None) or with fixed path-error constraint of greater than or equal to 4 m.

While localization is guaranteed when we simulate our repeat path using a flatness-based MPC with fixed path-error constraint of 3 m, we manage to achieve a *significantly* higher average speed (Fig. 7.7(b)) with our proposed perception-aware (P-A) constraint. In this case, our perception-aware approach achieves an average speed between that obtained with a fixed path error constraint of 4 m and 5 m. However, with both a fixed path-error constraint of 4 m and 5 m we do not guarantee localization (see Fig. 7.7(a)). As illustrated in Fig. 7.8(b), under our perception-aware constraint we fly a repeat profile that allows a path error greater than 4 m after the first corner, but then comes in to a 3 m path error after the second corner. This shows that the allowable error under which we can localize can significantly vary along the path. This is not only captured in our perception-aware constraint, but also exploited to achieve a higher overall speed.

Another key result highlighted in Fig. 7.8 is that under our perception-aware constraint, we do not repeat the taught path in the same way in Case 1 and Case 2. This is because it accounts for the landmark locations with respect to the path. In Case 1, the direction of our path error relative to the teach path after corner 1 means that we move closer to the scene. In Case 2, the direction of our path error relative to the teach path after corner 1 means that we move further from the scene. By moving further away from the scene we can actually afford a greater path error compared to when moving into the scene. This is because when moving away from the scene we tend to achieve a smaller parallax angle for the same path error and keep more landmarks within view. This effect is captured in our perception-aware constraint and is, consequently, why we fly the same geometric path differently just by looking in different directions.

## 7.5 Summary

We have presented an approach of building perception-awareness into flatness-based MPC which allows for optimal fast flight under reliable vision-based navigation. The key is that it does this by capturing:

- The variability of the localization envelope (allowable path error under which we can still localize) along a path;

- The difference in the localization envelope from one path to another. This includes geometrically identical paths that are flown at different altitudes, in different environments or with the camera pointed in different directions (as we illustrated in simulation based on real vision data).

Practically, the proposed perception-aware approach is also beneficial as it does not require ‘re-tuning’ the allowable constraint for every new path or environment flown under vision-based navigation.
The approach in this chapter has three novel contributions:

- We develop and validate a simple geometric perception model (for nominal lighting conditions - i.e., little difference in lighting conditions between teach and repeat) using over 12 km of data for our visual localization system in [Warren et al., 2019].
- We show how to integrate this perception model in a chance constraint, such that localization is guaranteed, in our MPC and how to convert it to a deterministic nonlinear constraint.
- Using real-world perception data, we provide simulation results demonstrating the value of our perception-aware MPC in terms of reliably but optimally self-regulating speed along a path compared to a similar perception-agnostic control.

The *key insight* of this chapter is that different prior visual environment knowledge, for example, when flying a path using different camera directions, results in planning a different optimal path.

# Chapter 8

# Discrete-Time Flatness Control

**Bypassing Full State Estimation Using Discrete-Time Flatness**

## 8.1 Overview and Related Work

Multirotor unmanned aerial vehicles (UAVs), see Fig. 6.1, are mechanically simple and highly maneuverable, which makes them suitable to a wide range of applications such as infrastructure inspection, see [Thakur et al., 2019], 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, see, for example, [Foehn et al., 2020] or [Gao et al., 2020].

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, see, for example, [Warren et al., 2019] or [Song et al., 2021].

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.

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., see [Foehn et al., 2020] or [Kaufmann et al., 2019]; (ii) designing robust control to account for the worst-case state estimation error learnt offline from data, see [Jarin-Lipschitz et al., 2020] or [Dean et al., 2020]; or (iii) performing output feedback control by coupling control and state estimation, see [Brunke et al., 2021] or [Kogel et al., 2017].

In this chapter, we present an alternative approach that bypasses 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.

### 8.1.1 Improving 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), see [Foehn et al., 2020], or visual teach and repeat (VT&R) framework, see [Warren et al., 2019] or [Gao et al., 2020]. For example, vision-based drone racing relies on SLAM, see [Foehn et al., 2020] or [Kaufmann et al., 2019]. 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, see [Song et al., 2021], 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, see [Nguyen et al.,2018].

### 8.1.2 Designing robust control for worst-case state estimation

In [Dean et al., 2020], 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 [Jarin-Lipschitz et al., 2020]. 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.

### 8.1.3 Coupling 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, see [Brunke et al., 2021]. Robust output feedback MPC can be a tube-based method (which guarantees constraint satisfaction despite measurement noise), see [Brunke et al., 2021] or [Kogel et al., 2017], or a minmax method (which optimizes the worst-case performance), see [Copp et al.,2017]. Current theory applies to linear models only, see [Brunke et al., 2021] or [Kogel et al., 2017]. 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, see [Brunke et al., 2021]. 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.

### 8.1.4 Contributions

In this chapter, we present a predictive controller that does not rely on the full-state estimate for vision-based multirotor flight. An accurate full-state estimate is often challenging due to typically noisy IMU measurements, an infrequent position update from the vision system and an imperfect motion model used to obtain high-rate state estimates required by control. Our controller avoids inaccurate velocity and acceleration estimates and instead relies on only a window of outputs, specifically position and yaw, and previously sent inputs.

In our flatness-based MPC in Chapter 3 (based on continuous-time flatness or differential flatness), we rely on an accurate state estimate to ensure that the initial condition in Theorem 1 holds. As we demonstrate below, this approach significantly underperforms in the case of state estimator errors. In our vision-based system, state estimation errors are a result of visual drift, time-delays, and an inaccurate motion model used in the estimator. In this chapter, we show that flatness holds for the discretization of multirotor dynamics (or discrete-time flatness). This is a useful result because it allows us to use a finite window of output (or position) observations and inputs to characterize the control system. We can, therefore, perform output feedback and the controller does not rely on a state estimate.

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.

## 8.2 Background on 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 [Guillot et al., 2020] and [Kolar et al., 2019]. The main difference is that time derivatives in Definition 1 are replaced by forward-shifts of the output, see [Diwold et al., 2020].

**Definition 5 (Discrete-Time Flatness)** *A nonlinear system model,*

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{8.1}$$

*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:*

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

$$\mathbf{x}_k = \tilde{\tilde{\Phi}}\left(\mathbf{y}_k, \mathbf{y}_{k+1}, ..., \mathbf{y}_{k+r-1}\right), \tag{8.2}$$

$$\mathbf{u}_k = \tilde{\tilde{\Psi}}^{-1}\left(\mathbf{y}_k, \mathbf{y}_{k+1}, ..., \mathbf{y}_{k+r}\right), \tag{8.3}$$

*where* $\tilde{\Lambda}, \tilde{\tilde{\Phi}}, \tilde{\tilde{\Psi}}$ *are smooth functions,* $\delta$ *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 (8.1) 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 $\mathbf{y}_k$, see [Diwold et al., 2020]. 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*, see [Alsalti et al., 2021].

Unfortunately, differential flatness of (2.1) does not necessarily imply difference flatness of a discretization (either exactly if possible or using Euler discretization) of (2.1). A counterexample is provided in [Diwold et al., 2020]. 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).

## 8.3 Methodology

We show that discrete-time flatness, as described in Definition 5, holds for the Euler discretization of the full multirotor dynamics model (from [Mellinger et al., 2011]). 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. 8.4 and

8.5. In Sec. 8.3.3, we illustrate how discrete-time flatness can be used in a predictive controller that uses only input and (flat) output information.

## 8.3.1 Discrete Flatness of Euler-Discretized Multirotor Dynamics Model

### Dynamics Model

We consider the following discrete-time model (using the Euler discretization approximation of the model in [Mellinger et al., 2011]):

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

where $\delta t$ 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 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 $\phi_k$, pitch $\theta_k$ and yaw $\psi_k$, and the angular velocities (in the vehicle frame) $p_k, q_k, r_k$ at time step $k$. The translational dynamics are:

$$\begin{bmatrix} \ddot{x}_k \\ \ddot{y}_k \\ \ddot{z}_k + g \end{bmatrix} = \frac{T_k}{m} \begin{bmatrix} \mathbf{R}_{13,k} \\ \mathbf{R}_{23,k} \\ \mathbf{R}_{33,k} \end{bmatrix}, \tag{8.4}$$

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 $\mathbf{R}$ is the rotation matrix from the body to inertial frames.

The notation $\mathbf{R}_{ij,k}$ denotes the $i$ row and $j$ column entry of rotation matrix $\mathbf{R}$ at time step $k$. The rotational dynamics are:

$$\begin{bmatrix} \dot{p}_k \\ \dot{q}_k \\ \dot{r}_k \end{bmatrix} = \begin{bmatrix} -\frac{(I_{zz}-I_{yy})}{I_{xx}} q_k r_k + \frac{1}{I_{xx}} \tau_k^x \\ -\frac{(I_{xx}-I_{zz})}{I_{yy}} p_k r_k + \frac{1}{I_{yy}} \tau_k^y \\ -\frac{(I_{yy}-I_{xx})}{I_{zz}} p_k q_k + \frac{1}{I_{zz}} \tau_k^z \end{bmatrix}, \tag{8.5}$$

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 $I_{xx}, I_{yy}, I_{zz}$.

## Discrete-Time Flatness Derivation

We consider the output comprising 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 $\mathbf{x}_k$, (8.2) in Definition 5, and input $\mathbf{u}_k = [T_k, \tau_k^x, \tau_k^y, \tau_k^z]^T$, (8.3) in Definition 5, can be determined from forward shifts of the output $\mathbf{y}_k$.

**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 $[\mathbf{R}_{13,k}, \mathbf{R}_{23,k}, \mathbf{R}_{33,k}]^T$ is a unit vector in (8.4), we can obtain this vector from forward shifts of the output as $[\mathbf{R}_{13,k}, \mathbf{R}_{23,k}, \mathbf{R}_{33,k}]^T = \frac{\mathbf{t}_k}{|\mathbf{t}_k|}$ where:

$$\mathbf{t}_k = \begin{bmatrix} \frac{x_{k+2}-2x_{k+1}+x_k}{\delta t^2} \\ \frac{y_{k+2}-2y_{k+1}+y_k}{\delta t^2} \\ \frac{z_{k+2}-2z_{k+1}+z_k}{\delta t^2} + g \end{bmatrix}. \tag{8.6}$$

We obtain the pitch $\theta_k$ and roll $\phi_k$ at time step $k$ in terms of 2 forward shifts of the output by plugging in the above expressions for $\mathbf{R}_{13,k}, \mathbf{R}_{23,k}, \mathbf{R}_{33,k}$ into:

$$\theta_k = \text{atan}\left(\frac{\mathbf{R}_{13,k}}{\mathbf{R}_{33,k}} C_{\psi_k} + \frac{\mathbf{R}_{23,k}}{\mathbf{R}_{33,k}} S_{\psi_k}\right), \tag{8.7}$$

$$\phi_k = \text{atan}\left(\left(\frac{\mathbf{R}_{13,k}}{\mathbf{R}_{33,k}} S_{\psi_k} - \frac{\mathbf{R}_{23,k}}{\mathbf{R}_{33,k}} C_{\psi_k}\right) C_{\theta_k}\right), \tag{8.8}$$

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 $\mathbf{R}_k$ at time step $k$ in terms of forward shifts of the output.

We can obtain similar expressions for $\mathbf{R}_{13,k+1}, \mathbf{R}_{23,k+1}, \mathbf{R}_{33,k+1}$ at time step $k+1$ by computing $\mathbf{t}_{k+1}$ in (8.6). We determine the pitch $\theta_{k+1}$ and roll $\phi_{k+1}$ at time step $k+1$ by using $\mathbf{R}_{13,k+1}, \mathbf{R}_{23,k+1}, \mathbf{R}_{33,k+1}$ and $\psi_{k+1}$ in (8.7) - (8.8). We can obtain similar expressions for the pitch $\theta_{k+2}$ and roll $\phi_{k+2}$ at time step $k+2$. The angular velocity of the vehicle in the inertial frame $\omega_k$ at time step $k$ is (see [Mellinger et al., 2011] for details):

$$\omega_k = \begin{bmatrix} C_{\psi_k} & \mathbf{R}_{12,k} & 0 \\ S_{\psi_k} & \mathbf{R}_{22,k} & 0 \\ 0 & \mathbf{R}_{32,k} & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi}_k \\ \dot{\theta}_k \\ \dot{\psi}_k \end{bmatrix} = \mathbf{R}_k \begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix}.$$

Exploiting the Euler discretization, we obtain:

$$\begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix} = \mathbf{R}_k^{-1} \begin{bmatrix} C_{\psi_k} & \mathbf{R}_{12,k} & 0 \\ S_{\psi_k} & \mathbf{R}_{22,k} & 0 \\ 0 & \mathbf{R}_{32,k} & 1 \end{bmatrix} \begin{bmatrix} \frac{\phi_{k+1}-\phi_k}{\delta t} \\ \frac{\theta_{k+1}-\theta_k}{\delta t} \\ \frac{\psi_{k+1}-\psi_k}{\delta t} \end{bmatrix}. \tag{8.9}$$

Using the relationship between forward shifts of the output $\mathbf{y}_k$ and the Euler angles $\theta_k, \phi_k, \psi_k$ at time step $k$, see (8.7) - (8.8), and $\theta_{k+1}, \phi_{k+1}, \psi_{k+1}$ at time step $k+1$ in (8.9), 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.

**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 (8.4) as $T_k = m|\mathbf{t}_k|$, where $|\mathbf{t}_k|$ is the magnitude of $\mathbf{t}_k$ in (8.6). Using the Euler discretization of the rotational dynamics (8.5), the torques at time step $k$ are:

$$\begin{bmatrix} \tau_k^x \\ \tau_k^y \\ \tau_k^z \end{bmatrix} = \begin{bmatrix} I_{xx} \frac{p_{k+1}-p_k}{\delta t} + (I_{zz} - I_{yy})\, q_k r_k \\ I_{yy} \frac{q_{k+1}-q_k}{\delta t} + (I_{xx} - I_{zz})\, p_k r_k \\ I_{zz} \frac{r_{k+1}-r_k}{\delta t} + (I_{yy} - I_{xx})\, p_k q_k \end{bmatrix}. \tag{8.10}$$

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 (8.10) 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 (8.3) in Definition 5, we call this a window of $r = 4$.

### 8.3.2 Discrete Flatness of Multirotor Model in 2-D with First-Order Pitch-Roll Dynamics

We consider a slightly simpler model in the simulations, Sec. 8.4, and experiments, Sec. 8.5, performed in this chapter. 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. 6.1) where a black-box attitude controller is implemented onboard, see [Greeff et al., 2018].

### 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 the 2-D position $x_k$, $y_k$, the 2-D velocity $\dot{x}_k$, $\dot{y}_k$, and the pitch $\theta_k$ and roll $\phi_k$ angles. We consider the dynamics:

$$\dot{\mathbf{x}}_k = [\dot{x}_k, g\frac{\mathbf{R}_{13,k}}{\mathbf{R}_{33,k}}, \dot{y}_k, g\frac{\mathbf{R}_{23,k}}{\mathbf{R}_{33,k}}, \frac{\tilde{k}}{\tau}\theta_{\text{cmd},k} - \frac{\theta_k}{\tau}, \frac{\tilde{k}}{\tau}\phi_{\text{cmd},k} - \frac{\phi_k}{\tau}]^T, \tag{8.11}$$

where $\mathbf{R}$ is the rotation matrix from the body to inertial frames, $\mathbf{R}_{ij,k}$ is the $i$ row and $j$ column of rotation matrix $\mathbf{R}$ at time step $k$, $\tau$ is the first-order time constant and $\tilde{k}$ is the first-order gain for the roll and pitch dynamics. The control input $\mathbf{u}_k = [\theta_{\text{cmd},k}, \phi_{\text{cmd},k}]^T$ includes the commanded pitch and roll. We assume the yaw is fixed, i.e., $\psi_k = \psi_\star \quad \forall k$.

### 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 $\mathbf{y}_k$.

**State from Forward Shifts of Output:** Similar to Sec. 8.3.1, 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 (8.11), we obtain the rotation at time step $k$ in terms of forward-shifts of the output. That is, $\frac{\mathbf{R}_{13,k}}{\mathbf{R}_{33,k}} = \frac{x_{k+2}-2x_{k+1}+x_k}{g\delta t^2}$ and $\frac{\mathbf{R}_{23,k}}{\mathbf{R}_{33,k}} = \frac{y_{k+2}-2y_{k+1}+y_k}{g\delta t^2}$. The pitch $\theta_k$ and roll $\phi_k$ at time step $k$ in terms of forward shifts of the output is obtain from (8.7) - (8.8) with $\psi_k = \psi_\star$.

**Input from Forward Shifts of Output:** Similar to Sec. 8.3.1, the rotation at time step $k+1$, i.e., $\frac{\mathbf{R}_{13,k+1}}{\mathbf{R}_{33,k+1}} = \frac{x_{k+3}-2x_{k+2}+x_{k+1}}{g\delta t^2}$ and $\frac{\mathbf{R}_{23,k+1}}{\mathbf{R}_{33,k+1}} = \frac{y_{k+3}-2y_{k+2}+y_{k+1}}{g\delta t^2}$, is used to determine the pitch $\theta_{k+1}$ and roll $\phi_{k+1}$ at $k+1$ in terms of forward shifts of the output. From the first-order pitch and roll dynamics we can determine the pitch and roll commands $\theta_{\mathrm{cmd},k}$ and $\phi_{\mathrm{cmd},k}$ from the pitch and roll at time steps $k$ and $k+1$, i.e.:

$$\begin{aligned}\theta_{\mathrm{cmd},k} &= \frac{1}{\tilde{k}}\left(\tau\theta_{k+1} - \theta_k\right),\\ \phi_{\mathrm{cmd},k} &= \frac{1}{\tilde{k}}\left(\tau\phi_{k+1} - \phi_k\right).\end{aligned}$$

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.,

$$\mathbf{u}_k = \tilde{\Psi}^{-1}\left(\mathbf{y}_k, \mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \mathbf{y}_{k+3}\right). \tag{8.12}$$

We coin this function (8.12) the *Output-To-Input Map* with a window $r=3$ in (8.3). The *Input-To-Output Map* is described by its inverse, i.e.,

$$\mathbf{y}_{k+3} = \tilde{\Psi}\left(\mathbf{y}_k, \mathbf{y}_{k+1}, \mathbf{y}_{k+2}, \mathbf{u}_k\right). \tag{8.13}$$

### 8.3.3 Predictive Control Exploiting Discrete Flatness

We present a predictive control design in Fig. 8.1 that exploits discrete-time flatness, from Definition 5, 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 (8.14); *(ii) Input-To-Output Map* – maps a window of past inputs and output measurements to constraint the future trajectory using (8.15); and *(iii) Output Trajectory Optimization* – optimizes the output trajectory using (8.16).

#### 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

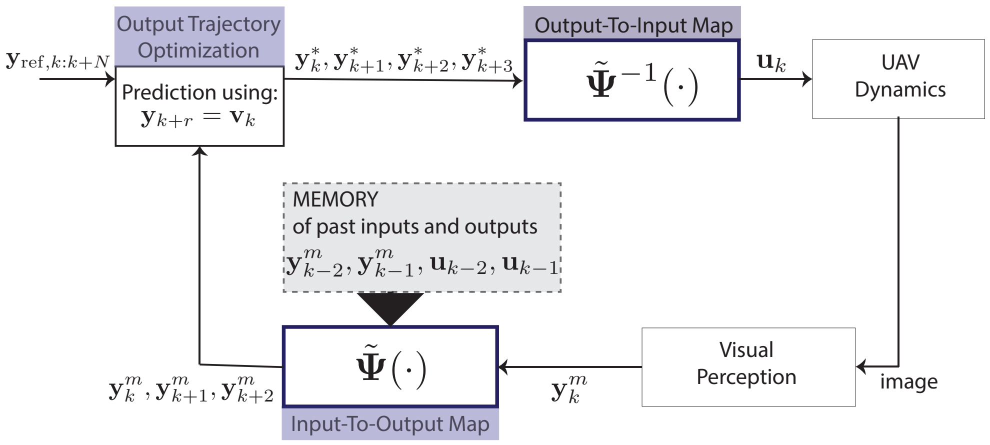

Output Trajectory Optimization
Prediction using: $\mathbf{y}_{k+r} = \mathbf{v}_k$
$\mathbf{y}_{\mathrm{ref},k:k+N}$
$\mathbf{y}^*_k, \mathbf{y}^*_{k+1}, \mathbf{y}^*_{k+2}, \mathbf{y}^*_{k+3}$
Output-To-Input Map
$\tilde{\boldsymbol{\Psi}}^{-1}(\cdot)$
$\mathbf{u}_k$
UAV Dynamics
MEMORY of past inputs and outputs
$\mathbf{y}^m_{k-2}, \mathbf{y}^m_{k-1}, \mathbf{u}_{k-2}, \mathbf{u}_{k-1}$
$\tilde{\boldsymbol{\Psi}}(\cdot)$
Input-To-Output Map
$\mathbf{y}^m_k, \mathbf{y}^m_{k+1}, \mathbf{y}^m_{k+2}$
$\mathbf{y}^m_k$
Visual Perception
image

Figure 8.1: Overview of predictive control (horizon $N$) design using a window $r = 3$ of input and (flat) output data by exploiting discrete-time flatness.

to determined the input from (8.3) as:

$$\mathbf{u}_k = \tilde{\Psi}^{-1}\left(\mathbf{y}^*_k, \mathbf{y}^*_{k+1}, ...\mathbf{y}^*_{k+r}\right). \tag{8.14}$$

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 (8.12) for our 2-D multirotor model (Sec. 8.3.2) with a window size $r = 3$.

## 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:

$$\mathbf{y}^m_{k+(j-1)} = \tilde{\Psi}\left(\mathbf{y}^m_{k-(r-j+1)}, ..., \mathbf{y}^m_{k+(j-2)}, \mathbf{u}_{k-(r-j+1)}\right), \tag{8.15}$$

where $j = 2, ..., r$ and $r$ is the window size. For example, the *Input-to-Output Map* in (8.13) for our 2-D multirotor model (Sec. 8.3.2) 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:

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

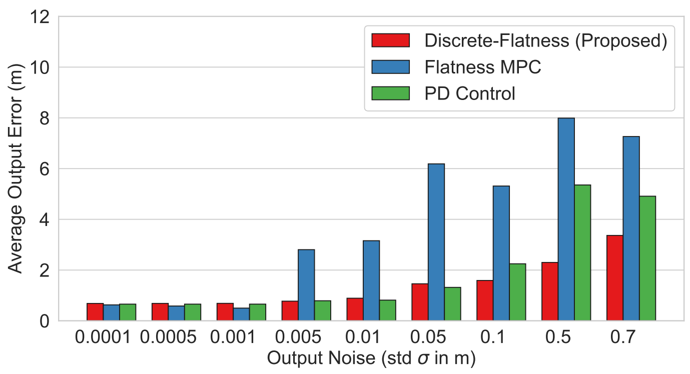


Figure 8.2: Simulation comparison of the effect of position noise on the tracking performance of *PD Control* in green, *Flatness MPC* (predictive control using continuous-time flatness) in blue, and our discrete-time flatness or *Discrete-Flatness* in red. Measurements of the $x$-position have zero-mean Gaussian noise $\mathbf{y}^m = \mathbf{y} + [\mathcal{N}(0, \sigma^2), 0]^T$, where we vary the standard deviation $\sigma$ (Output Noise). For each $\sigma$, 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.

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

### Output Trajectory Optimization

At each time step, we solve the following optimization problem:

$$\begin{aligned} \mathbf{y}^*_{k:k+N} = \underset{\mathbf{y}_{k:k+N}}{\text{argmin}} \quad & J\left(\mathbf{y}_{k:k+N}\right) \\ \text{s.t.} \quad & \mathbf{y}_{k+j-1} = \mathbf{y}^m_{k+j-1}, \forall j = 1, .., r, \end{aligned} \tag{8.16}$$

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.

## 8.4 Simulations

We consider the dynamics in (8.11) from Sec. 8.3.2 and consider a 1-D motion in the $x$-direction with $\psi^* = 0$, $y = 0$, $\dot{y} = 0$ and $\phi = 0$. The dynamics (8.11) are executed

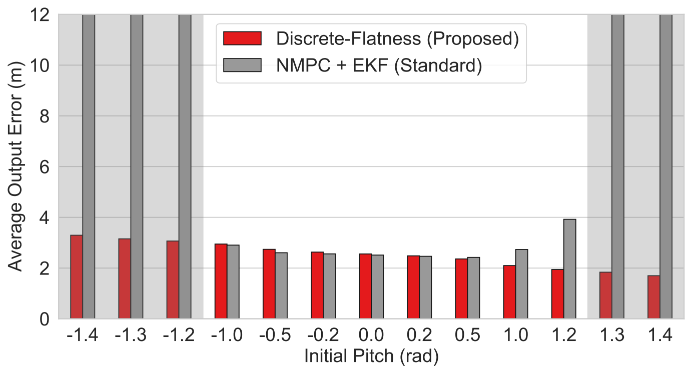


Figure 8.3: Simulation comparison of the effect of initial state uncertainty on the performance of the proposed *Discrete-Flatness* in red and *NMPC+EKF*, i.e., using a standard nonlinear MPC with an extended Kalman filter (EKF) for state estimation, in grey. Our proposed *Discrete-Flatness* has robustness 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 (8.2) to discard biased state estimates after $r$ time steps. The shaded grey region shows where the *NMPC+EKF* approach is unstable.

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 (8.16):

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

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, $\tilde{\mathbf{Q}}$ weights the error with the reference point and $\tilde{\mathbf{R}}$ weights a regularisation term.

## 8.4.1 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 [Greeff et al., 2018] 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 $\sigma$, we compute the average position tracking error for the feasible reference trajectory $\mathbf{y}_{\text{ref}} = At\sin(\omega t)$ where $A = 1.4, \omega = 1.0$ over 10 trials for each controller. In Fig. 8.2, 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. 8.2) significantly worsens as a result of heightened noise in the higher-order derivatives used in the state estimate. PD 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.

### 8.4.2 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. 8.3, 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.

## 8.5 Experiments

We compare three controllers (*PD Control* [Warren et al., 2019], *Flatness MPC* [Greeff et al., 2018], 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. 8.3.2. As described in Sec. 6.3 in Chapter 6, 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. 6.13 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. 6.13 for sharp turns at higher speeds STEAM does not accurately estimate the trajectory. *Flatness MPC* relies on position, velocity and acceleration estimates from

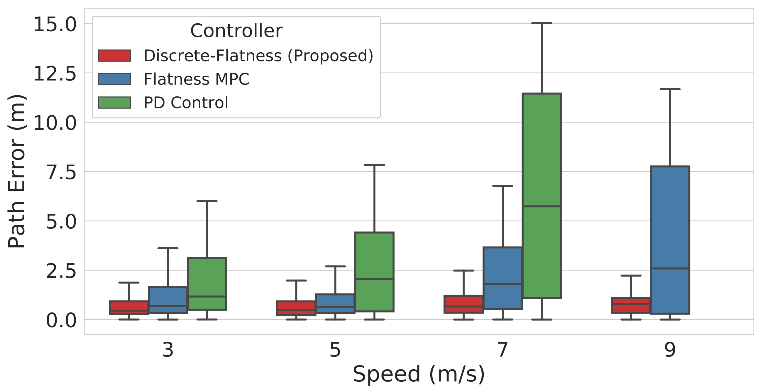


Figure 8.4: Comparison of path error at increasing desired speeds for autonomous vision-based flight of multirotor in Fig. 6.1 using *PD Control* (green), *Flatness MPC* (blue) and our proposed *Discrete-Flatness* controller (red). The box plots are computed over 3 trials (for each controller at each speed) repeating the L-shaped path in Fig. 8.5. 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 $\leq 5$ m/s and $65-80\%$ for high speeds $> 5$ m/s over *Flatness MPC*.

STEAM. *PD Control* relies on position and velocity estimates from STEAM. Our proposed *Discrete-Flatness* approach relies on only the position estimate as feedback.

### 8.5.1 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., $\mathbf{y}_{\text{ref},k} = \mathbf{y}_{\text{closest}}$, and computing the reference velocity $\dot{\mathbf{y}}_{\text{ref},k}$ as the desired speed in the direction of the next keyframe. *PD Control* [Warren et al., 2019] determines the commands at time step $k$ by weighting the position error $\mathbf{y}_k - \mathbf{y}_{\text{ref},k}$ with the velocity error $\dot{\mathbf{y}}_k - \dot{\mathbf{y}}_{\text{ref},k}$. 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 $\mathbf{y}_{\text{closest}}$ on the path, i.e., $\mathbf{y}_{\text{ref},k} = \mathbf{y}_{\text{closest}}$. 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) $\dot{\mathbf{y}}_{\text{ref},k}$ as $\mathbf{y}_{ref,k+1} = \mathbf{y}_{\text{ref},k} + \delta t \dot{\mathbf{y}}_{\text{ref},k}$, for the prediction horizon $k+1, k+2, ...k+N$. The reference velocity $\dot{\mathbf{y}}_{\text{ref},k}$ changes

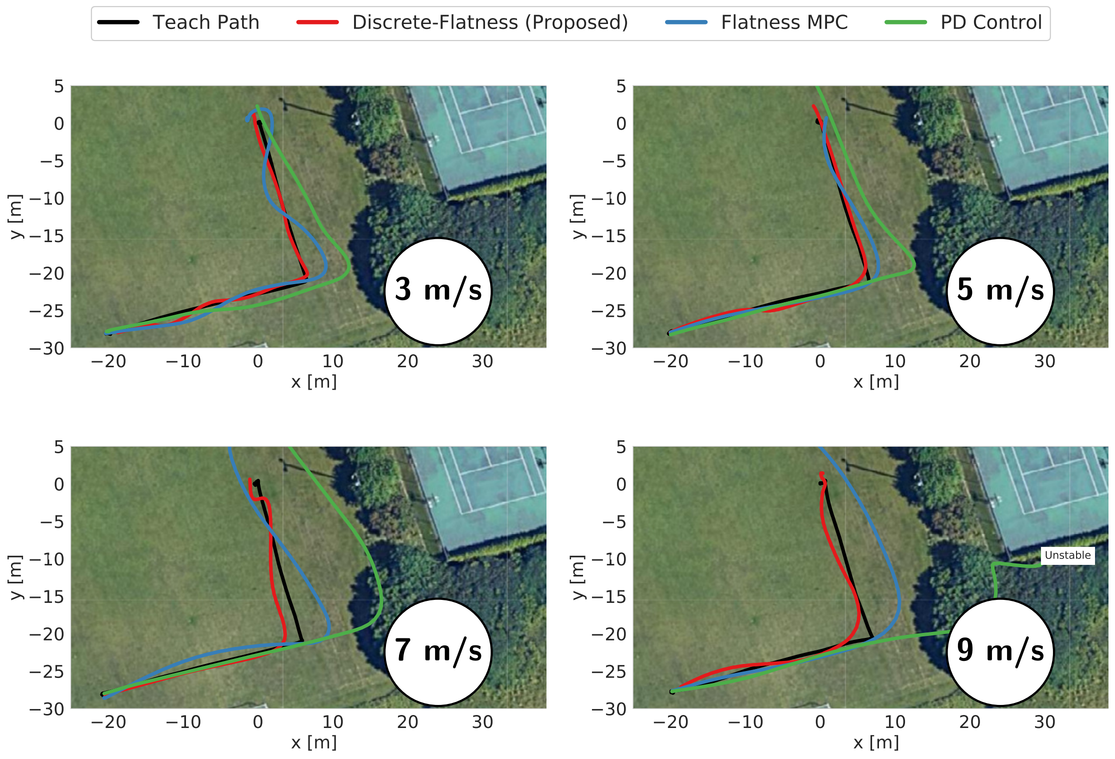


Figure 8.5: Visualization of path flown when repeating the *teach* path (black) under vision-based navigation using different controllers with a desired speed of (a) 3 m/s (b) 5 m/s (c) 7 m/s and (d) 9 m/s. The L-path starts at the lower left corner (at approximately $(-20, -30)$) and ends at the origin. We show one trial (out of three used in Fig. 8.4) 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. 8.4) by turning before the corner even at high speeds.

as the output reference moves to the next straight line segment of the path.

## 8.5.2 Parameters

We consider a lookahead of 2 s for *Discrete-Flatness* and 1.5 s 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.

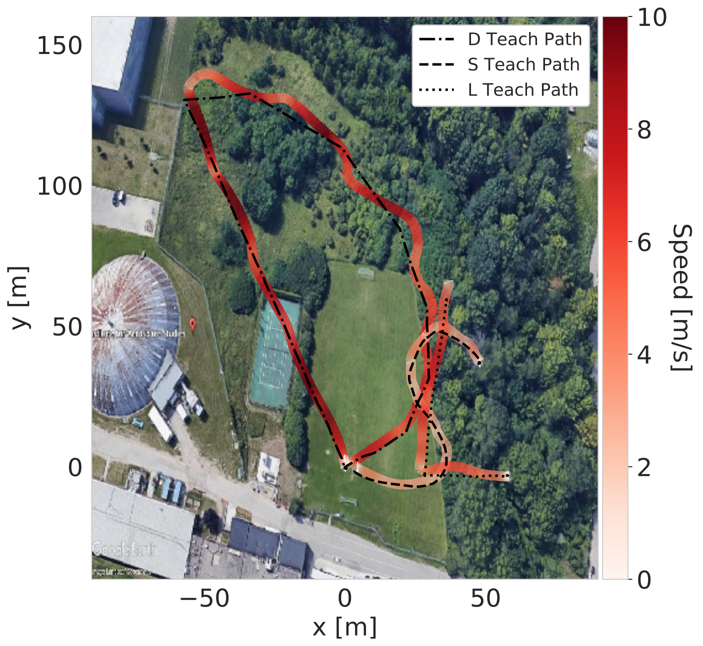


Figure 8.6: Visualization of 3 additional paths flown using vision-based navigation and the proposed *Discrete-Flatness* control achieving speeds up to 10 m/s. We fly three paths: 1) a D-shaped path (at 10 m/s), 2) an S-shaped path (at 3 m/s) and 3) an L-shaped path (at 8 m/s). This spells out the abbreviation of our lab "Dynamic System Lab" or DSL.

### 8.5.3 Results

We consider a simple L-shape teach path, black in Fig. 8.5, at a fixed altitude of 10 m above ground. In Fig. 8.4, 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. 8.4, *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. 8.5) 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

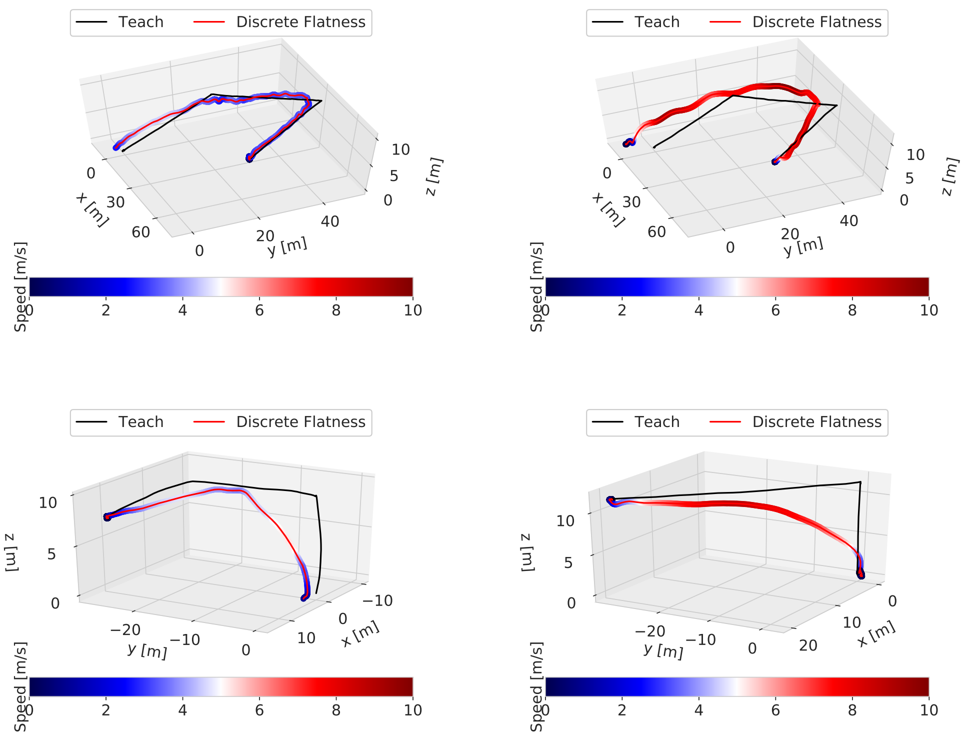


Figure 8.7: Visualization of various 3D paths flown using vision-based navigation and the proposed *Discrete-Flatness* control.

highlighted in Fig. 8.5.

**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. 8.6 with 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. 6.13. We have extended the proposed *Discrete-Flatness* control to 3-D and show outdoor trajectories at various speeds using vision-in-the-loop in Fig. 8.7.

## 8.6 Summary

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 full state estimation. This is particularly relevant to future work on learning-based control that could improve performance by simultaneously learning $\tilde{\Psi}^{-1}(\cdot)$ in (8.12) and $\tilde{\Psi}(\cdot)$ in (8.13) using *only input and output data*. More traditional learning-based controllers require full-knowledge of the state to learn the dynamics model. Furthermore, this overcomes limitations when the state estimator and controller use different learned dynamics models. Another potential challenge is that the window size $r$ determined from first principles may differ for the actual system. In future work, we propose developing a learning-based controller by varying the window size and observing the effect on the learning performance. Future work is also required to extend the proposed control design to account for different input and output rates. Video: `https://tinyurl.com/flyoutthewindow`

# Chapter 9

# CLOUD: Canadian Longterm Outdoor UAV Dataset

## 9.1 Overview

Unmanned Aerial Vehicles (UAVs) are mechanically simple and highly maneuverable which makes them well-suited to a wide range of applications including patrol-and-inspection, search-and-rescue and delivery operations. The limitations of traditional GPS-based navigation to dropout, jamming and interference has motivated relying on vision-based navigation as a suitable lightweight alternative. Despite the significant recent success in autonomous high-speed visual navigation, see, for example [Foehn et al., 2020], longterm robust visual navigation in outdoor environments is still an open challenge.

The Canadian Longterm Outdoor UAV Dataset (CLOUD) contains over 30 km of visual-inertial data collected at three different locations across Canada. Specifically, these locations include paths at Suffield, Montreal and the University of Toronto Institute for Aerospace Studies (UTIAS) in Toronto as shown in Fig. 9.1.

While outdoor UAV visual-inertial datasets exist, see for example [Majdik et al., 2017], they tend to be quite limited in scope. Specifically, they cover one location at a single time of day during a single season of the year. We present this dataset for two main reasons:

- To challenge the robustness of state-of-the-art visual navigation in different environments and to significant lighting and seasonal changes;
- To enable the use of rendered satellite images for UAV navigation.

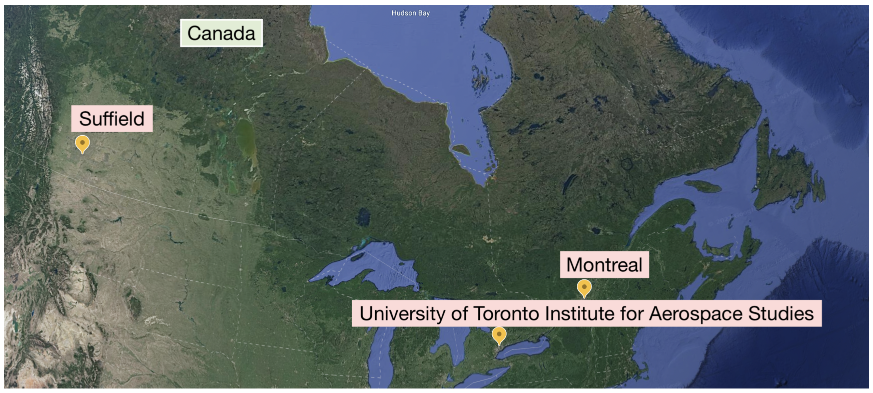


Figure 9.1: Our data is collected at three different locations across Canada.

CLOUD contains a variety of environmental, lighting and seasonal conditions which makes it well-suited for research in robust UAV visual localization. Moreover, many other potential use cases exist including satellite image UAV navigation, experience-based localization and simultaneous localization and mapping.
The data in CLOUD has been used throughout this thesis. We list a few relevant publications:

- Our CLOUD data, specifically the data in Table 9.2, has been used in Visual Teach and Repeat (VT&R) experiments in [Warren et al., 2019] to demonstrate the closed-loop autonomous performance of a visual navigation pipeline for emergency return in the event of GPS failure.

- Our CLOUD data has been used in [Greeff et al., 2020(b)] along with a feature-based visual localization algorithm to develop a simple perception model that is used in a perception-aware model predictive controller.

- Our CLOUD data, specifically the data in Table 9.7, has been used in visual localization using Google Earth images. The satellite images used for reconstruction to generate the Google Earth images may have been collected many years ago. This can result in significant differences between a live UAV image and the rendered Google Earth images. Our data has been used in [Patel et al., 2020] and [Bianchi et al., 2021] to try to address this challenge of robustly and accurately localizing live UAV images to Google Earth images.

Table 9.1: Comparison of our dataset with related visual-inertial UAV datasets.

| Dataset | Location | No. Env. | Cam. Rate (Hz) | IMU Rate (Hz) | Max Speed (m/s) | Sat. Im. | Total Km |
|---|---|---|---|---|---|---|---|
| `Euroc` | Indoor | 2 | 20 | 200 | 2.3 | No | 0.9 |
| `Blackbird` | Indoor | 3 | 120 | 100 | 7 | No | 60.0 |
| `Zurich Urban` | Outdoor | 1 | 20 | 10 | 3.9 | Yes | 2.0 |
| `UPenn Fast` | Outdoor | 1 | 40 | 200 | 17.5 | No | 2.4 |
| `UZH-FPV Racing` | Both | 2 | 30/50 | 500/1000 | 12.8/23.4 | No | 6.17 |
| `CLOUD (Ours)` | Outdoor | 3 | 15 | 400 | 14.3 | Yes | 36.58 |

## 9.2 Related Work

While indoor visual-inertial UAV datasets can provide very accurate ground truth data, their usefulness to longterm visual outdoor navigation may be quite limited as they generally consist of shorter flights, lack significant seasonal and lighting changes, and other disturbances commonly prevalent in outdoor applications. The EUROC MAV dataset, see [Burri et al., 2016], contains 11 trajectories flown indoors with a maximum speed of 2.3 m/s. The Blackbird Dataset, see [Antonini et al., 2020], is the largest indoor UAV dataset and contains 168 flights with over 17 trajectories up to velocities of 7 m/s. While it contains a variety of environments, these environments are rendered using photo-realistically generated images.

Many current UAV visual-inertial outdoor datasets target visual navigation during fast aggressive maneuvers, see [Sun et al., 2018] and [Delmerico et al., 2020]. The UPenn Fast dataset, used in [Sun et al., 2018], comprises of the same path flown at 4 different speeds achieving a maximum speed of 17.5 m/s. The UZH-FPV Racing dataset, see [Delmerico et al., 2020], significantly extends this and provides multiple outdoor aggressive trajectories with speeds up to 23.4 m/s. These datasets are well-suited to test aggressive visual-navigation. However, their application to test or benchmark robust longterm visual navigation is limited as they do not provide multiple trials over various lighting and seasonal conditions. Furthermore, they currently contain data from a single environment.

Our Canadian Longterm Outdoor UAV Dataset (CLOUD) was collected from 2018-2020 in three distinct environments in Canada and across multiple seasons. To our knowledge, our dataset is currently the largest UAV visual outdoor dataset with over 30 km of visual-inertial flight data. We show a comparison to related datasets in Table 9.1. We include multiple trials of a given path and our data spans both rural and urban locations.

Moreover, similar to [Majdik et al., 2017], we also provide satellite images (generated using Google Earth) for the flown paths. Given the range of the data provide (different environments, lighting conditions, seasonal conditions), our dataset is also well-suited to enable and test UAV navigation using satellite images, see [Patel et al., 2020] and [Bianchi et al., 2021].

## 9.3 Datasets

### 9.3.1 Suffield Dataset

The Suffield dataset contains 15 trials (composed of both a Teach and Repeat run) collected at Suffield, Alberta between 14 - 15 June, 2018. The maximum speed during the trials varies between 3 m/s and 15 m/s. The Suffield dataset contains approximately 14 km of flight data. The scenes comprise of mainly grassland with some shipping containers. The data is well-suited for high-speed visual navigation.

### 9.3.2 Montreal Dataset

The Montreal dataset contains 10 trials (composed of both a Teach and Repeat run) collected at Silo 5 near downtown Montreal between 9 - 11 September, 2018. The Montreal dataset contains approximately 7 km of flight data. The scenes comprise of urban structure, including flying near various building structures. A section of the path includes scenes with the St Lawrence River. We also provide Google Earth images taken along the path. This data is well-suited for inspection and patrol applications. We show example images and the Google Earth view in Fig. 9.2.

### 9.3.3 UTIAS Dataset

The UTIAS dataset contains 5 different paths (composed of various trials) collected around the University of Toronto Institute for Aerospace Studies between 25 October 2018 - December 2020. The UTIAS dataset contains approximately 13 km of flight data where scenes comprise of mainly natural vegetation with some building structure. Notably, this dataset contains scenes flown in snow, windy conditions, and under extreme sun glare. We also provide Google Earth images taken along the various flight paths.

#### Shorter UTIAS Datasets

Multiple trials are collected for the first three paths (UTIAS Straight, UTIAS circle, UTIAS field). They are generally shorter paths with simple configurations (i.e., flying a straight line, in a u-shape and in a circle). These datasets are useful for quick benchmarking and testing because they comprise of simple flight paths but include challenging feature-sparse scenes as highlighted in Fig. 9.3.

#### UTIAS Day Datasets

UTIAS Day was collected within a single day from morning to evening and is ideally suited to test robustness of UAV visual navigation to daily lighting changes. RBG images and RTK-GPS are provided for this dataset.

#### UTIAS Winter Datasets

UTIAS Winter was collected over multiple weeks during the winter months in Toronto, Canada and is ideally suited to test robustness of UAV visual navigation to rapid seasonal changes as shown in Fig. 9.4. RBG images and RTK-GPS are provided for this dataset.

## 9.4 Hardware

Our dataset was captured using a DJI M600 Pro as shown in Fig. 6.1. A StereoLabs camera is attached using a DJI Ronin-Mx gimbal to the M600 platform. We additionally store logged GPS data, raw IMU data and the gimbal orientation. We collect (grey-scale or RGB) camera images (from the left camera) at 15 Hz. For the data in *UTIAS Suffield*, *UTIAS Montreal* and shorter *UTIAS* datasets we provide grey-scale images at 15 Hz. For the data in *UTIAS Day* and *UTIAS Winter* we provide colour images at 15 Hz and RTK-GPS.

## 9.5 Data Description

### 9.5.1 Coordinate Frames

Fig. 9.5 illustrates the coordinate frames applicable to our dataset. All data is described in one of these frames. The rigid transformations (described in the form of a vector followed by a quaternion) from the vehicle frame $\mathcal{F}_{\text{vehicle}}$ to the base frame

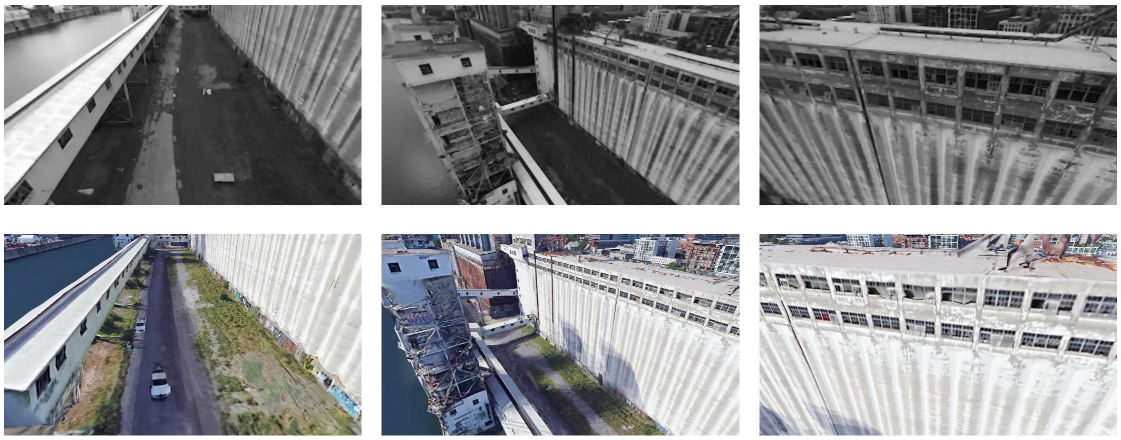
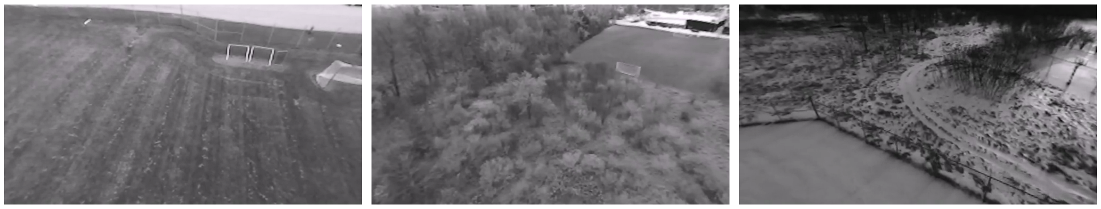


Figure 9.2: Example of live images (top) and Google Earth view (bottom) from Montreal dataset.



(a) UTIAS Field (b) UTIAS Circle (c) UTIAS Straight

Figure 9.3: Example images from shorter paths around UTIAS. These datasets are useful for quick benchmarking and testing as they include challenging feature-sparse scenes.

$\mathcal{F}_{\text{base}}$ and from the camera frame $\mathcal{F}_{\text{camera}}$ to the gimbal frame $\mathcal{F}_{\text{gimbal}}$ can be found at https://www.dynsyslab.org/cloud-dataset/. We also include the transformation from the gimbal (when it is in the neutral position) $\mathcal{F}_{\text{gimbal}}$ to the vehicle $\mathcal{F}_{\text{vehicle}}$. The key co-ordinates frames are the ground $\mathcal{F}_{\text{ground}}$, vehicle $\mathcal{F}_{\text{vehicle}}$ and base $\mathcal{F}_{\text{base}}$ frames:

- *Ground Frame(s)* - There are two ground frames, i.e., East-North-Up (ENU) and North-East-Down (NED).

- *Vehicle and Base Frames* - The vehicle $\mathcal{F}_{\text{vehicle}}$ and base $\mathcal{F}_{\text{base}}$ frames are both Forward-Left-Up (FLU). The vehicle frame $\mathcal{F}_{\text{vehicle}}$ is aligned at approximately the center-of-gravity of the M600 while the base frame $\mathcal{F}_{\text{base}}$ is shift vertically to align approximate at the base of the legs of the M600 UAV.

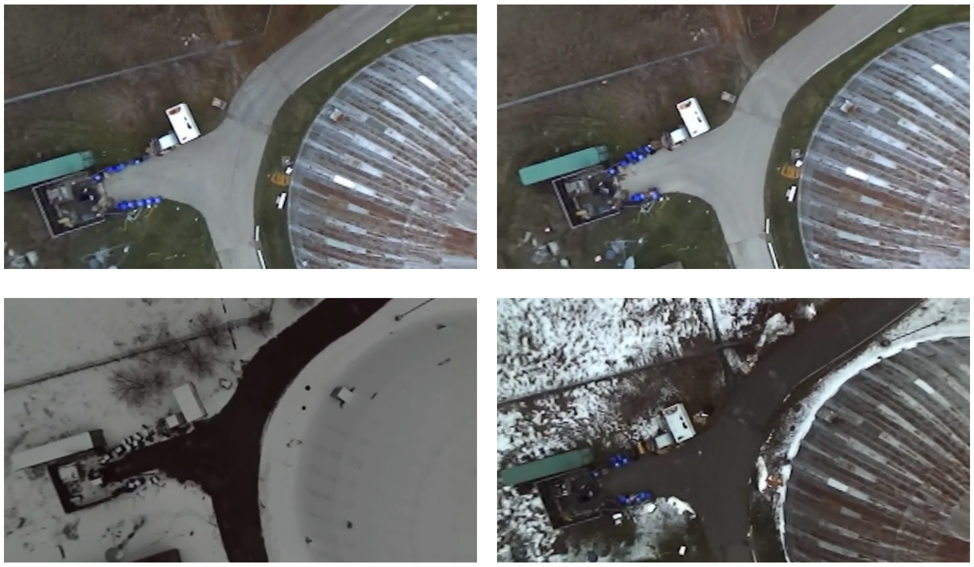


Figure 9.4: Example images from *UTIAS Winter* dataset highlight potential use-case for demonstrating robustness of visual navigation to rapid seasonal changes.

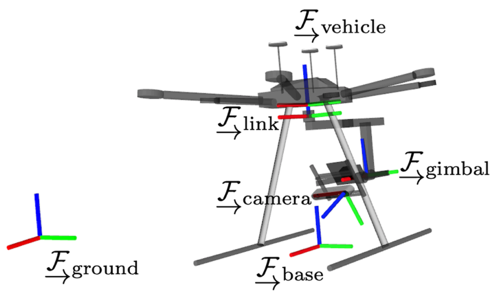

$\underrightarrow{\mathcal{F}}_{\text{vehicle}}$
$\underrightarrow{\mathcal{F}}_{\text{link}}$
$\underrightarrow{\mathcal{F}}_{\text{gimbal}}$
$\underrightarrow{\mathcal{F}}_{\text{camera}}$
$\underrightarrow{\mathcal{F}}_{\text{ground}}$
$\underrightarrow{\mathcal{F}}_{\text{base}}$

Figure 9.5: Overview of coordinate frames.

### 9.5.2 File Structure

The dataset is split into trial folders (which can be downloaded separately). For the *Suffield*, *Montreal* and shorter *UTIAS* datasets each trial contains a teach and repeat run. The repeat run is the same path as the teach run but traversed in the reverse direction (often autonomously using vision) shortly after the teach path ends. For the *UTIAS Day* and *UTIAS Winter* only a teach folder is provided. For trials where we have collected Google Earth Images along the path, see Tables 9.2-9.8, we also provide a folder with the associated Google Earth Images and their GPS locations and orientation.

The images subfolder contain images obtained from the left camera. Images are named in the order they were obtained. In each of the run folders we provide the associated timestamp for each image, GPS measurements, attitude, acceleration, velocity, gimbal angles and gimbal state. For more detail see the next section *Interpreting files.*

### 9.5.3 Interpreting files

All timestamps are Unix timestamps given in nanoseconds. These may be converted to UTC via standard datetime library functions.

#### Velocity Stream

The estimated velocity is provide as vector in the ground frame (ENU) $\mathcal{F}_{\text{ground}}$ at a rate of 50 Hz. It is provided in a human-readable, comma-delimited text file *velocities.txt* where each row provides the timestamp followed by the velocity vector.

#### Attitude Stream

The raw attitude is provide as quaternion from the vehicle frame $\mathcal{F}_{\text{vehicle}}$ to the ground frame (ENU) $\mathcal{F}_{\text{ground}}$ at a rate of 100 Hz. It is provided in a human-readable, comma-delimited text file *attitude.txt* where each row provides the timestamp followed by the attitude quaternion.

#### Acceleration Stream

The raw acceleration is provide as vector in the vehicle frame $\mathcal{F}_{\text{vehicle}}$ at a rate of 400 Hz. It is provided in a human-readable, comma-delimited text file *acceleration.txt* where each row provides the timestamp followed by the acceleration vector.

#### GPS Stream

The raw GPS is provide as latitude, longitude and altitude (in m with respect to the WGS 84 ellipsoid) at a rate of 50 Hz. It is provided in a human-readable, comma-delimited text file *gps.txt* where each row provides the timestamp followed by the GPS measurement.

#### Gimbal State Stream

The estimated gimbal state is provide as vector and quaternion which described the transformation from the base frame $\mathcal{F}_{\text{base}}$ to the camera frame $\mathcal{F}_{\text{camera}}$ at a rate of

50 Hz. It is provided in a human-readable, comma-delimited text file *gimbalstate.txt* where each row provides the timestamp followed by the vector and quaternion.

### Gimbal Angle Stream

The raw gimbal angles are provide as Euler angles that describe the rotation of from the gimbal frame $\mathcal{F}_{\text{gimbal}}$ to the $\mathcal{F}_{\text{ground}}$ (NED) frame at a rate of 50 Hz. The Euler angles are in the standard ZYX convention (roll, pitch, yaw). They are given in degrees. The roll and pitch angles are given relative to the global NED frame, however the yaw angle is given relative to the Vehicle/Base frames. Unfortunately, this is the convention used in the DJI ROS SDK. However, for further convenience our estimated gimbal state (position and orientation) relative to the Base frame $\mathcal{F}_{\text{base}}$ is provided in the *Gimbal State Stream*.

### Image IDs Stream

The images ids are also human-readable and comma-delimited with each row representing the timestamp followed by the associated image ID (as found in the images folder). Images are collected at approximately 15 Hz.

## 9.6 Summary of Trials

Table 9.2: Summary of Suffield Dataset.

| Trial | Date (yy-mm-dd) | Teach Start Time | Teach Max Speed | Teach Sat. Images | Repeat Start Time | Repeat Max Speed | Repeat Sat. Images |
|---|---|---|---|---|---|---|---|
| 1 | 18-06-14 | 11:39 | 3.51 m/s | No | 11:41 | 4.43 m/s | No |
| 2 | 18-06-14 | 11:47 | 7.55 m/s | No | 11:49 | 7.64 m/s | No |
| 3 | 18-06-14 | 11:55 | 8.60 m/s | No | 11:57 | 8.69 m/s | No |
| 4 | 18-06-14 | 12:02 | 10.58 m/s | No | 12:04 | 10.68 m/s | No |
| 5 | 18-06-14 | 12:09 | 12.63 m/s | No | 12:11 | 12.16 m/s | No |
| 6 | 18-06-14 | 12:22 | 15.06 m/s | No | 12:23 | 13.27 m/s | No |
| 7 | 18-06-14 | 12:38 | 3.51 m/s | No | 12:41 | 3.43 m/s | No |
| 8 | 18-06-14 | 13:05 | 7.57 m/s | No | 13:07 | 7.56 m/s | No |
| 9 | 18-06-14 | 13:12 | 10.64 m/s | No | 13:14 | 10.81 m/s | No |
| 10 | 18-06-14 | 13:18 | 14.28 m/s | No | 13:20 | 10.62 m/s | No |
| 11 | 18-06-14 | 13:32 | 7.62 m/s | No | 13:34 | 7.70 m/s | No |
| 12 | 18-06-14 | 13:40 | 7.72 m/s | No | 13:42 | 7.62 m/s | No |
| 13 | 18-06-14 | 13:49 | 7.68 m/s | No | 13:51 | 7.62 m/s | No |
| 14 | 18-06-15 | 10:34 | 3.62 m/s | No | 10:37 | 4.07 m/s | No |
| 15 | 18-06-15 | 11:13 | 3.50 m/s | No | 11:15 | 3.82 m/s | No |

Table 9.3: Summary of Montreal Dataset.

| Trial | Date (yy-mm-dd) | Teach Start Time | Teach Max Speed | Teach Sat. Images | Repeat Start Time | Repeat Max Speed | Repeat Sat. Images |
|---|---|---|---|---|---|---|---|
| 1 | 18-09-11 | 14:07 | 2.47 m/s | Yes | 14:10 | 2.63 m/s | No |
| 2 | 18-09-11 | 14:22 | 3.47 m/s | No | 14:25 | 3.53 m/s | No |
| 3 | 18-09-11 | 14:39 | 3.52 m/s | No | 14:41 | 3.79 m/s | No |
| 4 | 18-09-11 | 16:18 | 3.52 m/s | No | 16:21 | 2.81 m/s | No |
| 5 | 18-09-11 | 16:32 | 3.60 m/s | No | 16:35 | 2.76 m/s | No |
| 6 | 18-09-11 | 16:46 | 3.61 m/s | No | 16:49 | 2.35 m/s | No |
| 7 | 18-09-10 | 10:11 | 2.43 m/s | No | 10:14 | 2.44 m/s | No |
| 8 | 18-09-10 | 12:47 | 2.59 m/s | No | 12:50 | 2.89 m/s | No |
| 9 | 18-09-10 | 15:16 | 2.63 m/s | No | 15:20 | 2.68 m/s | No |
| 10 | 18-09-09 | 16:48 | 2.56 m/s | No | 16:51 | 2.57 m/s | No |

Table 9.4: Summary of UTIAS Field Dataset.

| Trial | Date | Teach | | | Repeat | | |
|---|---|---|---|---|---|---|---|
| | (yy-mm-dd) | Start Time | Max Speed | Sat. Images | Start Time | Max Speed | Sat. Images |
| 1 | 19-08-16 | 11:16 | 5.60 m/s | Yes | 11:17 | 6.26 m/s | No |
| 2 | 19-08-16 | 11:23 | 5.57 m/s | No | 11:24 | 6.93 m/s | No |
| 3 | 19-08-16 | 11:35 | 5.63 m/s | No | 11:36 | 8.81 m/s | No |
| 4 | 19-08-16 | 11:42 | 5.62 m/s | No | 11:43 | 10.84 m/s | No |
| 5 | 19-08-16 | 11:44 | 5.62 m/s | No | 11:50 | 11.89 m/s | No |

Table 9.5: Summary of UTIAS Circle Dataset.

| Trial | Date | Teach | | | Repeat | | |
|---|---|---|---|---|---|---|---|
| | (yy-mm-dd) | Start Time | Max Speed | Sat. Images | Start Time | Max Speed | Sat. Images |
| 1 | 18-10-25 | 13:40 | 3.37 m/s | Yes | 11:42 | 3.71 m/s | No |
| | 18-10-26 | | | | 15:30 | 3.55 m/s | No |
| | 18-10-29 | | | | 09:25 | 4.31 m/s | No |
| | 18-10-30 | | | | 16:42 | 3.62 m/s | No |
| | 18-10-30 | | | | 17:52 | 4.08 m/s | No |
| | 18-11-05 | | | | 13:46 | 3.79 m/s | No |
| | 18-11-08 | | | | 16:20 | 4.56 m/s | No |
| | 18-11-12 | | | | 12:10 | 3.64 m/s | No |
| 2 | 18-11-08 | 14:37 | 3.49 m/s | No | 14:42 | 3.92 m/s | No |
| | 18-11-08 | | | | 14:49 | 4.73 m/s | No |
| | 18-11-08 | | | | 14:57 | 7.20 m/s | No |
| | 18-11-08 | | | | 15:09 | 8.67 m/s | No |

Table 9.6: Summary of UTIAS Straight Dataset.

| Trial | Date | Teach | | | Repeat | | |
|---|---|---|---|---|---|---|---|
| | (yy-mm-dd) | Start Time | Max Speed | Sat. Images | Start Time | Max Speed | Sat. Images |
| 1 | 19-02-11 | 16:08 | 3.37 m/s | Yes | 16:27 | 3.38 m/s | No |
| | 19-02-11 | | | | 16:30 | 3.24 m/s | No |
| | 19-02-11 | | | | 16:34 | 3.36 m/s | No |
| | 19-02-11 | | | | 16:41 | 3.35 m/s | No |
| | 19-02-11 | | | | 16:45 | 3.40 m/s | No |

Table 9.7: Summary of UTIAS Day Dataset.

| Trial | Date | Teach | | | Repeat | | |
|---|---|---|---|---|---|---|---|
| | (yy-mm-dd) | Start Time | Max Speed | Sat. Images | Start Time | Max Speed | Sat. Images |
| 1 | 19-08-01 | 10:37 | 3.75 m/s | Yes | | | |
| 2 | 19-08-01 | 11:56 | 3.79 m/s | No | | | |
| 3 | 19-08-01 | 14:35 | 3.68 m/s | No | | | |
| 4 | 19-08-01 | 17:50 | 3.61 m/s | No | | | |
| 5 | 19-08-01 | 20:24 | 3.68 m/s | No | | | |

Table 9.8: Summary of UTIAS Winter Dataset.

| Trial | Date | Teach | | | Repeat | | |
|---|---|---|---|---|---|---|---|
| | (yy-mm-dd) | Start Time | Max Speed | Sat. Images | Start Time | Max Speed | Sat. Images |
| 1 | 20-11-05 | 15:55 | 3.56 m/s | Yes | | | |
| 2 | 20-11-16 | 15:49 | 3.53 m/s | No | | | |
| 3 | 20-11-23 | 14:08 | 3.46 m/s | No | | | |
| 4 | 20-12-10 | 11:58 | 3.50 m/s | No | | | |

# Chapter 10

# Summary and Future Work

This chapter summarizes the key contributions of this thesis. We explain the relationship between the chapters and refer to the related publications. At the end of this chapter we provide further insights into research paths for future work.

## 10.1 Contributions and Publications

Our first motivation was for multirotor UAVs to achieve safe high-performance fast flight by accounting for nonlinearities and unknown dynamics in *computationally tractable control* algorithms that can be used in real-time operation in a high-frequency feedback loop.

A core idea behind this thesis is to exploit a structural property of many nonlinear models, including *multirotors*, known as differential flatness, see [Fliess et al., 1995] and [Mellinger et al., 2011]. Intuitively, differential flatness allows us to separate the nonlinear model into a linear dynamics component and a nonlinear transformation. A common control approach for differentially flat systems is to try to cancel the nonlinear term (using feedback linearization) and design a controller using the linear dynamics component which is computationally efficient to design. However, in reality, performance and safety are limited by the mismatch between the nominal model (for example, used to cancel the nonlinearity) and the actual system dynamics (which may include unknown disturbances and changing dynamics). In **Part I: Exploiting Flatness Structure**, we explored control design that exploits differential flatness to develop computationally tractable control that is able to achieve high performance and safety despite the mismatch between the nominal model and actual system.

**Contribution I:** In Chapter 3, we utilized the differential flatness property to couple a feedback model predictive controller that uses the linear dynamics with feedforward linearization to account for nonlinearity. The proposed Flatness-Based Model Predictive Control (FMPC) approach has two main advantages. Firstly, by solving a convex quadratic program in linear model predictive control, we achieved a computational benefit over nonlinear model predictive control which requires solving a non-convex nonlinear program. Secondly, by using feedforward linearization, we still account for nonlinearities and demonstrate, practically, that we obtain improved robustness to model parameter uncertainty and input time delays over the more common feedback linearization method.

The contributions of this chapter are three-fold.

- Firstly, we presented a novel *Flatness-based Model Predictive Control* (FMPC) architecture that coupled feedback MPC with feedforward linearization. We demonstrated that feedback MPC and feedforward linearization have a symbiotic relationship. Feedforward linearization allows us to use a simplified linear model in MPC, while using MPC as our feedback controller allows us to satisfy the conditions for feedforward linearization.
- Secondly, we implemented our FMPC architecture on a multirotor UAV, accounting for inner-loop dynamics and known input time delays.
- Finally, we demonstrate experimental results for FMPC as an outer-loop controller on a multirotor UAV with improved trajectory tracking performance in many cases over nonlinear model predictive control (NMPC) and linear model predictive control (LMPC).

The publication associated with this chapter is:

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

Our method in Chapter 3 is model-based and closed-loop performance is limited to the accuracy of the prior model and does not improve with online-data. Consequently, in Chapters 4 - 5, we explored learning-based control as it can often outperform conventional model-based techniques in the presence of model uncertainties by using online system data. However, most state-of-the-art learning-based nonlinear trajectory tracking controllers still lack any formal guarantees.

**Contribution II:** In Chapter 4, we exploit the property of differential flatness to design an online, robust learning-based controller to achieve both high tracking performance and probabilistically guarantee a uniform ultimate bound on the tracking error. This chapter has three key contributions:

- We present a novel approach that uses a Gaussian Process to improve feedback linearization and quantify how well we are able to linearize the system.
- We proved that theoretically our quantified uncertainty can be combined with a standard robust LQR to probabilistically guarantee an ultimate bound on the tracking error.
- We show through simulations how our proposed approach results in improved tracking performance over related learning-based controllers that use differential flatness.

The publication associated with this chapter is:

- M. Greeff and A. P. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using Gaussian Processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121-1126, 2020.

Our flatness-based robust learning control approach in Chapter 4 has three main limitations: (i) it relies on robust LQR and cannot be combined with alternatives controllers, for example, linear model predictive control; (ii) the method is computationally intractable for high-rate feedback as it takes on average approximately 0.3 s to compute each iteration and (iii) it cannot account for state and input constraints.

**Contribution III:** In Chapter 5, we learn a filter that can augment any controller for control-affine differentially flat systems to efficiently certify robot tracking convergence and input constraints in the presence of model uncertainty.

The three key contributions of this work are:

- We provide a novel filter that can augment any controller for control-affine differentially flat systems to achieve high tracking performance while certifying robot tracking convergence and input constraints despite model uncertainty.
- We show that for control-affine systems the filter comprises of an optimization problem that is not only convex but can be solved efficiently as a second-order cone program (SOCP).

- We demonstrate, in simulation, a significant reduction in average computation over related methods, while still achieving high tracking performance. This makes our proposed approach suitable for online and onboard implementation in high-rate feedback loops, for example, on autonomous UAVs.

The publication associated with this chapter is:

- M. Greeff, A. W. Hall, and A. P. Schoellig, "Learning a stability filter for uncertain differentially flat systems using Gaussian Processes," in *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 789-794, 2021.

The second challenge with real-world environments is that we may not be able to always operate with reliable GPS. This has motivated developing *vision-based navigation* that relies primarily on lightweight, inexpensive onboard camera sensors. One such vision-based approach uses a Visual Teach and Repeat (VT&R) framework that allows the UAV to repeat a previously taught path by matching current visual features to those in the locally metric map created during teach, see [Gao et al., 2020] or [Warren et al., 2019]. In **Part II: Autonomous Vision-Based Flight**, we considered the Visual Teach and Repeat (VT&R) framework and focused on the *controller* required to autonomously repeat a previously taught path.

**Contribution IV:** Many state-of-the-art controllers are perception-agnostic and tend to assume that the action computed by the controller has no effect on the ability of vision-based navigation to determine the UAV's location (or localization). In Chapter 7, we present a perception-aware model predictive controller that accounts for its effect on the visual localization capabilities of our system in Chapter 6.

There are three points of novelty to the proposed perception-aware MPC:

- We develop and validate a simple geometric perception model (for nominal lighting conditions - i.e., little difference in lighting conditions between teach and repeat) using over 12 km of data for our visual localization system in Chapter 6.

- We show how to integrate this perception model in a chance constraint, such that localization is guaranteed, in MPC and how to convert it to a deterministic nonlinear constraint.

- Using real-world perception data, we provide experimental simulation results demonstrating the value of our perception-aware MPC in terms of reliably (without losing localizability) but optimally self-regulating speed along a path compared to similar perception-agnostic controllers.

The publication associated with this chapter is:

- 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, pp. 9412-9419, 2020.

The main limitation with the perception-aware MPC formulation in Chapter 7 is that it relies on traditional model predictive control which assumes a perfect full-state estimate (including position, velocity and acceleration) to make accurate predictions.

**Contribution V:** In Chapter 8, we present an alternative predictive controller that does not rely on the full-state estimate for vision-based multirotor flight. An accurate full-state estimate is often challenging due to typically noisy IMU measurements, an infrequent position update from the vision system and an imperfect motion model used to obtain high-rate state estimates required by control. Our controller avoids inaccurate velocity and acceleration estimates and instead relies on only a window of outputs, specifically position and yaw, and previously sent inputs.

The contributions of this chapter 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.

The publication associated with this chapter is:

- M. Greeff, S. Zhou, and A. P. Schoellig, "Fly out the window: Exploiting discrete-time flatness for fast vision-based multirotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5023-5030, 2022.

## 10.2 Future Work

### Part I: Exploiting Flatness Structure

**Certifying Differential Flatness:** In Chapters 3 - 5, we rely on the assumption that the system is differentially flat in a known and measured flat output. This assumption is based on the derivation of differential flatness for inaccurate first-principle models. For example, we assume that our multirotor system is differentially flat (in position and yaw) based on a first-principle model of the system. There is still an open question as to whether this property remains to be true despite unknown disturbances and/or unmodelled dynamics. Future work could develop techniques to certify when the property of differential flatness, currently derived from inaccurate first-principle models, can be safely exploited by using data from physical robotic systems. Further extensions could also investigate data-driven approaches to learn or estimate likely flat outputs of the system.

**Long-Term and Short-Term Learning:** In Chapters 4 - 5, we develop learning-based techniques to achieve high-performance while certifying safety (through guaranteeing tracking convergence) by exploiting differential flatness. However, we did not differentiate between long-term learning and online short-term learning. An interesting next step would be to explore the role of long-term learning and online short-term learning in differential flatness-based control architectures. For example, one idea is to use long-term learning to learn and certify differential flatness and update the linear dynamics model while short-term learning is used to compensate for the nonlinear term (similar to our approaches in Chapters 4 - 5).

**State Constraints:** Future work could combine flatness-based model predictive control in Chapter 3 with the learning-based stability filter in Chapter 5. Such an approach would benefit from using a linear model predictive controller to consider constraints while using data from the system to improve performance. In our linear model predictive controller in Chapter 3 we approximated state constraints to constrain the transformed state (i.e., the flat output and higher derivatives). Future work, could explore how to approximate state constraints when transforming them into convex constraints on the transformed state such they are guaranteed to be satisfied.

## Part II: Autonomous Vision-Based Flight

**Perception-Aware Models:** In Chapter 7, we presented a perception model to capture the limitations of feature-based localization approaches. However, in recent years there has been significant progress in alternative more robust localization methods such as using mutual information, as in [Patel et al., 2020], or kernel-based approaches, as in [Bianchi et al., 2021]. Unlike feature-based approaches, these methods have been successfully used to localize using satellite images. Future work should investigate developing models that capture the limitations, accuracy and variability of these current state-of-the-art localization approaches. These models could be used in planners and robust controllers to actively plan paths where the UAV is likely to be able localize well.

**Learning and Adaptive Discrete-Time Flatness Control:** In Chapter 8, we presented a predictive controller by exploiting discrete-time flatness to bypass a noisy full state estimate. The work in this chapter has been model-based. As such, future work could explore learning-based control that could improve performance by simultaneously learning both the input-to-output mappings and the output-to-input mappings in our discrete-time framework by using *only input and output data*. Most learning-based controllers, see Chapter 4 for example, require full-knowledge of the state to learn the dynamics model. Furthermore, this could potentially overcome limitations when the state estimator and controller use different learned dynamics models. Another interesting direction is to explore the role of adaptive control in discrete-time flatness control to compensate for both dynamic disturbances, such as wind, and visual odometry biases, such as a scale bias. Another potential challenge with our proposed discrete-time flatness control approach is that we use a model to determine the window size $r$ of output (specifically flat outputs) and input (our sent commands) data samples required for control. In practice the window size $r$ determined from first principle models may differ for the actual system. In future work, we propose developing a learning-based controller by varying the window size and observing the effect on the control performance. Our controller in Chapter 8 required the output estimation rate to be the same as the commanded input rate. This meant that we relied on an additional estimator (specifically STEAM) between the relative position estimate from visual odometry and the higher-rate estimates sent to the controller. Future work is also required to extend the proposed control design to account for different input and output rates.

# Bibliography

A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Autonomous Robots*, pp. 1-16, 2016.

P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery," in *Proc. IEEE Aerospace Conference*, pp. 1-8, 2008.

J. Han and Y. Chen, "Multiple UAV formations for cooperative source seeking and contour mapping of a radiative signal field," *Journal of Intelligent & Robotic Systems*, pp. 1-10, 2013.

M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11773-11780, 2014.

K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, "Robust model predictive flight control of unmanned rotorcrafts," *Journal of Intelligent & Robotic Systems*, pp. 1-27, 2015.

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

Y. Wang and S. P. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267-278, 2010.

M. Diehl, H. J. Ferreau, N. Haverbeke, *Efficient numerical methods for nonlinear MPC and moving horizon estimation.* In Nonlinear model predictive control,

Springer, Berlin, Heidelberg, pp. 391-417, 2009.
B. Houska, H. Ferreau, and M. Diehl, "ACADO toolkit - An open source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298-312, 2011.

J.A. Primbs and V. Nevistic, "MPC extensions to feedback linearizable systems," in *Proc. IEEE American Control Conference (ACC)*, pp. 2073-2077, 1997.

V. Nevistic and M. Morari, "Robustness of MPC-based schemes for constrained control of nonlinear systems," in *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 5823-5828, 1997.

M.V. Khotare, V. Nevistic, and M. Morari, "Robust constrained model predictive control for nonlinear systems: a comparative study," in *Proc. IEEE Conference of Decision and Control (CDC)*, pp. 2884-2889, 1995.

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

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

V. Hagenmeyer and E. Delaleau, "Exact feedforward linearization based on differential flatness," *International Journal of Control*, vol. 76, no. 6, pp. 537-556, 2003.

V. Hagenmeyer, S. Streif, and M. Zeitz, "Flatness-based feedforward and feedback linearization of the ball&plate lab experiment," in *Proc. IFAC-Symposium on Nonlinear Control Systems (NOLCOS)*, pp. 233-238, 2004.

J. Deng, V. M. Becarra, and R. Stobart, "Input constraints handling in an MPC/feedback linearization scheme," *International Journal of Applied Mathematics and Computer Science*, vol. 19, no. 2, pp. 219-232, 2009.

K. Margellos and J. Lygeros, "A simulation based MPC technique for feedback linearizable systems with input constraints," in *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 7539-7544, 2010.

M.J. Kurtz and M.A. Henson, "Feedback linearizing control of discrete-time nonlinear systems with input constraints," *International Journal of Control*, vol. 70, no. 4, pp. 603-616, 2010.

M. Fliess and R. Marquez, "Continuous-time linear predictive control and flatness: a module-theoretic setting with examples," *International Journal of Control*, vol. 73, no. 7, pp. 606-623, 2000.

V. Hagenmeyer and E. Delaleau, "Continuous-time non-linear flatness-based predictive control: an exact feedforward linearization setting with an induction drive example," *International Journal of Control*, vol. 81, no. 10, pp. 1645-1663, 2008.

M.W. Mueller and R. D' Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. IEEE European Control Conference (ECC)*, pp. 1383-1389, 2013.

F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 491-496, 2016.

J. Umlauft, L. Pöhler, and S. Hirche, "An uncertainty-based control Lyapunov approach for control-affine systems modelled by Gaussian processes," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 483-488, 2018.

C. J. Ostafew, A. P. Schoellig and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile path-tracking," *International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547-1563, 2016.

F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *Proc. IEEE European Control Conference (ECC)*, pp. 2496-2501, 2015.

M. Greeff and A. P. Schoellig, “Flatness-based model predictive control for quadrotor trajectory tracking,” in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 6740-6745, 2018.

T. Westenbroek, D. F. Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, “Feedback linearization for unknown systems via reinforcement learning,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1364-1371, 2020.
J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche, “Feedback linearization using Gaussian processes,” in *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 5249-5255, 2017.

M. K. Helwa, A. Heins, and A. P. Schoellig, “Provably robust learning-based approach for high-accuracy tracking control of Lagragian systems,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1587-1594. 2019.

M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modelling and control.* Hoboken, NJ, USA: Wiley, 2006.

C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning.* Cambridge, MA: MIT Press, 2006.

A. Pollastri and V. Tulli, “The distribution of the absolute value of the ratio of two correlated normal random variables,” *Statistica Applicazioni*, vol. 13, pp. 107-119, 2015.

G. G. Rigatos, *Nonlinear Control and Filtering Using Differential Flatness Approaches.* Switzerland: Springer, 2015.

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information theoretic regret bounds for Gaussian process optimization in the bandit setting,” *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250-3265, 2012.

C. D. McKinnon and A. P. Schoellig, “Learning probabilistic models for safe predictive control in unknown environments,” in *Proc. IEEE European Control*

*Conference (ECC)*, pp. 2472-2479, 2019.

J. Umlauft, L. Pöhler and S. Hirche, "An uncertainty-based control Lyapunov approach for control-affine systems modelled by Gaussian process," *IEEE Control Systems Letters*, vol. 2, pp. no. 3, pp. 483-488, 2018.

M. Greeff and A. P. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using Gaussian processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121-1126, 2020.

L. Zheng, R. Yang, J. Pan, H. Cheng and H. Hu, "Learning-based safety-stability-driven Control for safety-critical systems under model uncertainties," in *Proc. International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1112-1118, 2020.

E. Sontag. "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems & Control Letters*, vol. 13, pp. 117-123, 1989.

A. Taylor, V. Dorobantu, H. Le, Y. Yue, and A. Ames. "Episodic learning with control lyapunov functions for uncertain robotic systems," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 6878-6884, 2019.

F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects," in *Proc. IEEE American Control Conference (ACC)*, pp. 3683-3690, 2021.
Y. Nesterov and A. Nemirovski, *Interior-point polynomial algorithms in convex programming*, Siam studies in applied mathematics: 1994.

D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A. A. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4093-4099, 2020.

D. Thakur, G. Loianno, L. Jarin-Lipschitz, A. Zhou, and V. Kumar, "Autonomous inspection of a containment vessel using a micro aerial vehicle", in *Proc. IEEE Inter-*

*national Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1-7, 2019.

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.

P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "AlphaPilot: autonomous drone racing," *Autonomous Robots*, vol. 46, pp. 307-320, 2022.

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.

Y. Song, M. Steinweg, E. Kaufmann and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205-1212, 2021.

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. IEEE International. Conference on Robotics and Automation (ICRA)*, pp. 690-696, 2019.

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

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.

L. Jarin-Lipschitz, R. Li, T. Nguyen, V. Kumar and N. Matni, "Robust, perception-based control with quadrotors," in *Proc. IEEE International Conference on*

*Intelligent Robots and Systems (IROS)*, pp 7737-7743, 2020.

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. IEEE International Conference on Decision and Control (CDC)*, pp. 2183-2190, 2021.

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

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.

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

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.

B. Kolar, J. Diwold and M. M. Schöberl, “Necessary and sufficient conditions for difference flatness,” *IEEE Transactions on Automatic Control*, 2022, *In Press.*

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

M. Alsalti, J. Berberich, V. G. Lopez, F. Allgöwer, M. A. Müller, “Data-based system analysis and control of flat nonlinear systems,” in *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 1484-1489, 2021.

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. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 157-164, 2015.

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.

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.

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.

DARPA, “Fast Lightweight Autonomy (FLA),” 2015, solicitation number DARPA-BAA-15-16.

K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, Ke Sun, A. Zihao Zhu, J.A. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. Jose Taylor, and V. Kumar, “Fast, autonomous flight in GPS-denied and cluttered environments,” *Journal of Field Robotics*, vol. 35, no. 1, pp. 101-120, 2018.

M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, “Fast autonomous flight in warehouses for inventory applications,” *IEEE Robotics and Automation Letters*, vol. 3, pp. 3121-3128, 2018.

F. Gao, L. Wang, K. Wang, W. Wu, B. Zhou, L. Han and S. Shen, “Optimal trajectory generation for quadrotor teach-and-repeat,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1493-1500, 2019.

D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “PAMPC: Perception-aware model predictive control for quadrotors,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1-8, 2018.

B. Patel, M. Warren, and A. P. Schoellig, “Point me in the right direction: improving visual localization on UAVs with active gimballed camera pointing,” in

*Proc. Conference on Computer and Robot Vision (CRV)*, pp. 105-112, 2019.

W. Churchill, C. H. Tong, C. Gurau, I. Posner, and P. Newman, "Know your limits: embedding localiser performance models in teach and repeat maps," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4238-4244, 2015.

C. Gurau, C. H. Tong, and I. Posner, "Fit for purpose? predicting perception performance based on past experience," in *Proc. International Symposium on Experimental Robotics (ISER)*, pp. 454-464, 2016.

C. Gurau, D. Rao, C. H. Tong, and I. Posner, "Learn from experience: probabilistic prediction of perception performance to avoid failure," *The International Journal of Robotics Research*, pp. 1-15, 2017.

P. T. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, pp. 534-560, 2010.

M. Paton, K. Mactavis, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: multi-experience localization for long-term visual teach and repeat," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1918-1925, 2016.

B. Patel, T. D. Barfoot and A. P. Schoellig, "Visual localization with Google earth images for robust global pose estimation of UAVs," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6491-6497, 2020.

M. Bianchi and T. D. Barfoot, "UAV localization using autoencoded satellite images," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1761-1768, 2021.

M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157-1163. 2016.

A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The Blackbird UAV dataset," The International Journal of Robotics Research, vol. 39, no. 10-11, pp. 1346-1364, 2020.

A. L. Majdik, C. Till, and D. Scaramuzza, “The Zurich urban micro aerial vehicle dataset,” *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269-273, 2017.

K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar. “Robust stereo visual inertial odometry for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965-972, 2018.

J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler and D. Scaramuzza, ”Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6713-6719, 2019.

P. Billingsley, *Probability and measure*, John Wiley and Sons, 1995.

Stanford Artificial Intelligence Laboratory et al. *Robot Operating System [Internet]*. 2018. Available from: https://www.ros.org.

A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, “Learning High-Speed Flight in the Wild,” *Science Robotics*, vol. 6, no. 59, 2021.

H. Sira-Ramirez and S. K. Agrawal, *Differentially Flat Systems.* Crc Press, 2018.

G. Campion, G. Bastin, and B. Dandrea-Novel, “Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47-62, 1996.

R. M. Murray and S. S. Sastry, “Nonholonomic Motion Planning: Steering using Sinusoids,” *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700-716, 1993.

S. Jiang and K. Song, “Differential Flatness-Based Motion COntrol of a Steer-And-Drive Omnidirectional Mobile Robot,” in Proc. IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1167-1172, 2013.

B. Yüksel, G. Buondonno and A. Franchi, “Differential Flatness Control of Protocentric Aerial Manipulators with Any Number of Arms and Mixed Rigid-/Elastic-Joints,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 561-566.

V. Hagenmeyer and E. Delaleau, “Robustness analysis of exact feedforward linearization based on differential flatness,” *Automatica*, vol. 39, no. 11, pp. 1941-1946, 2003.