# Motion planning of quadcopters for enhanced autonomy in complex environments

by

Xiangyu Wu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2022

Motion planning of quadcopters for enhanced autonomy in complex environments

## Abstract

Motion planning of quadcopters for enhanced autonomy in complex environments

by

Xiangyu Wu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Assistant Professor Mark W. Mueller, Chair

Flight range and time, as well as autonomous flights in complex environments, are two challenges preventing quadcopters from being more widely used in the industry. This thesis reduces the effect of these two problems via several motion planning methods.

This thesis is composed of three parts. In the first part, two methods are proposed to improve the flight time (endurance) and distance (range) of quadcopters. The first method does so by finding the optimal flight speed using extremum seeking control. The second method extends the first method by finding the optimal flight sideslip as well as speed, and adds a step size adapter to the extremum seeking controller to improve its convergence speed. Both methods do not require the power consumption modeling of the quadcopters and can thus adapt to changing payloads and disturbances.

In the next part, we focus on the problem of fast collision avoidance flight in cluttered environments. First, we propose a computationally efficient memoryless planner for fast outdoor flights, using a depth camera for sensing obstacles and using the visual inertial odometry (VIO) for state estimation. Since the VIO may function poorly in areas with very few visual features or when the flight is overly aggressive. We then take the state estimation quality of the VIO into account during the trajectory planning. This perception-aware planner is able to guide the vehicle to areas with more VIO features and avoid overly aggressive trajectories. It improves the VIO's accuracy and reduces its failure rate, while only slightly reducing the flight speed.

Finally, an inertial navigation motion planning strategy is introduced. This state estimation method only requires the inertial measurement unit (IMU) and can be used as a backup when other methods fail. By breaking a long trajectory into multiple short "hopping trajectories" and introducing zero velocity updates when the vehicle is stationary on the ground, the state estimation variance can be drastically reduced and can be used in closed-loop control of quadcopters.

To my family.

# Contents

## Acknowledgments

I would like to first and foremost thank my parents and fiancee Lingxi Li for their love and encouragement not only during my study at UC Berkeley, but also during other important stages of my life. The invaluable time spent with you makes my life bright and colorful. Thanks for your support for the pursuit of my own path and I would not have reached so far without your selfless support.

I am particularly grateful for my Ph.D. advisor Professor Mark Mueller, who is an incredible mentor with immense knowledge, brilliant ideas, and intense enthusiasm. I really appreciate your careful and meticulous guidance in my study throughout the past five years, and I will strive to uphold the same level of excellence you instilled in the rest of my career.

I genuinely thank Professor Mark Mueller, Professor Kameshwar Poolla, and Professor Pieter Abbeel for serving on my dissertation committee. Meanwhile, I am thankful to Professor Anil Aswani and Professor Koushil Sreenath for serving on my qualifying exam committee.

Thanks to my colleagues at UC Berkeley: Nathan Bucki, Clark Zha, Karan Jain, Daisy Zhang, Ting-Hao Wang, Youngsang Suh, Andrea Tagliabue, Jun Zeng, Shuxiao Chen, Jaeseung Byun, Junseok Lee, Ryan Dimick, Ean Hall, Dennis Schradick, Zheng Jia, Conrad Holda, Saman Fahandezh-Saadi, Trey Fortmuller, Joey Kroeger, Natalia Perez, and Karan Mahesh. I learned a lot from you and thanks to your help with my research at Berkeley. I will cherish the enjoyable time we worked together.

Finally, the works in this thesis would not have been possible without funding from the following sources: the Graduate Division Block Grant, the J.K. Zee Fellowship, the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge, the Army Research Laboratory agreement no. W911NF-20-2-0105, the AFRI Competitive Grant no. 2020-67021-32855/project accession no. 1024262 from the USDA National Institute of Food and Agriculture, and China High-Speed Railway Technology Co., Ltd. The experimental testbed at the HiPeRLab is the result of the contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

# Chapter 1

# Introduction

A quadcopter is an aerial vehicle propelled by four motors attached to a rigid frame, an example of which is shown in Fig. 1.1. Generally, the propellers used have the same pitch angle. Two of them are clockwise and two of them are counter-clockwise. The thrust and torque of the vehicle can be controlled by changing the speed of each motor. A tutorial to the modeling, state estimation, and control of quadcopters can be found at [1].

Quadcopters are useful for a wide range of applications such as aerial photography [2] inspection [3], and transportation [4] thanks to their simple design and high maneuverability. However, limited flight time and reliable autonomous flights in complex environments are still challenges limiting their operations. In this thesis, we propose several methods to enhance

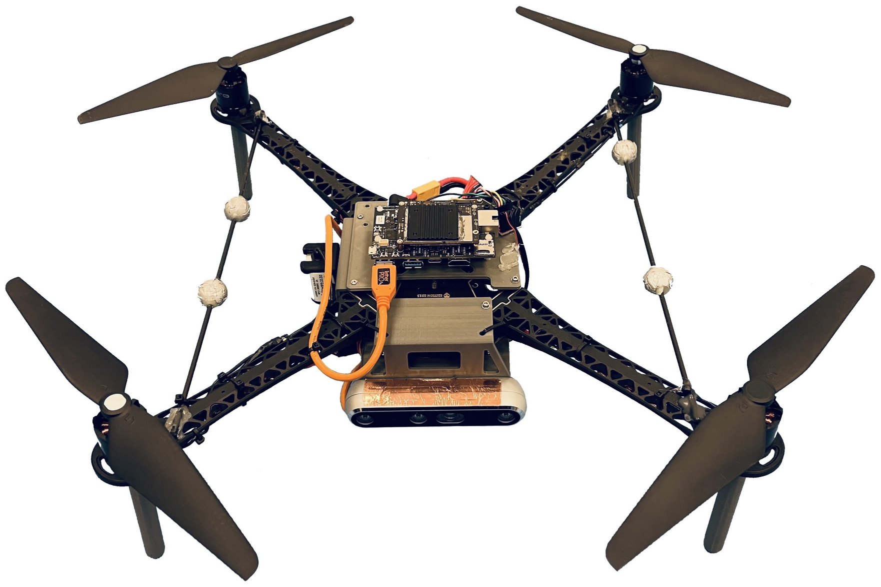


Figure 1.1: A custom-built quadcopter with motion caption maker balls and a stereo camera.

their autonomy via motion planning, and the following chapters are organized as follows:

In Chapter 2, we address limited flight time and endurance problem by finding the most energy-efficient flight speed. The proposed method is based on extremum seeking control, and is able to find the flight speed which maximizes a quadcopter's flight time (endurance) or flight distance (range) while moving along a given path, using on-board power measurement. It does not require any model of the power consumption of the system, can be executed online, and guarantees adaptation to unknown payloads. In indoor experiments we show that hovering is not the most energy-efficient loitering strategy, and demonstrate our method's ability to adapt to different aerodynamic disturbances, such as payloads, while flying at the optimal range velocity along a circular path. The method may be especially useful in applications where a quadcopter carries an unknown payload, allowing it to adapt flight speed for improved flight range.

In Chapter 3, we improve our method in Chapter 2. We propose a method for finding both the optimal speed and the sideslip angle of a multicopter when flying a given path to achieve either the longest flight distance or time. In contrast, in Chapter 2 only the flight speed is optimized. Flight speed and sideslip are chosen as optimization variables because they are often free variables in multicopter path planning, and can be changed without changing the mission. Our method is based on a novel multivariable extremum seeking controller with adaptive step size, which is inspired by recent work from the machine learning community on stochastic optimization. It converges faster than the standard extremum seeking controller with constant step size used in the previous chapter. In addition, like the method proposed in Chapter 2, this method does not require a power consumption model of the vehicle and is computationally efficient to run on an onboard embedded computer. In both indoor and outdoor experiments, the method is shown to converge with different payloads and in the presence of wind disturbances.

In Chapter 4 we investigate the problem of fast flight of the quadcopter in outdoor cluttered environments. We introduce a collision avoidance planner based on a recent planner named RAPPIDS (rectangular pyramid partitioning using integrated depth sensors). The RAPPIDS motion planner generates collision-free flight trajectories at high speed with low computational cost using only the latest depth image. In Chapter 4 we improve its performance by taking the following issues into account. (a) Changes in the dynamic characteristics of the multicopter that occur during flight, such as changes in motor input/output characteristics due to battery voltage drop. (b) The noise of the inertial navigation unit, which can cause unwanted control input components. (c) Planner utility function which may not be suitable for the cluttered environment. In this chapter we introduce solutions to each of the above problems and propose an improved RAPPIDS planner for the successful operation of quadcopters in a challenging outdoor cluttered environment. At the end of this chapter, we validate the planner's effectiveness by presenting the flight experiment results in a forest environment.

In Chapter 5, we take the state estimation quality of the vehicle from VIO (visual inertial odometry) into account in its trajectory planning. The research in this chapter is inspired by our research in Chapter 4, where VIO divergence is a major cause of flight failure. VIO is widely used for the state estimation of multicopters, but it may function poorly in environments with few visual features or in overly aggressive flights. The perception-aware planner proposed in this chapter is able to fly the vehicle to a goal position at high speed, avoiding obstacles in the environment while achieving good VIO state estimation accuracy. It samples a group of minimum jerk trajectories and finds collision-free trajectories among them, which are then evaluated based on their speed to the goal and perception quality. Both the features' motion blur and their locations are considered for the perception quality. The best trajectory from the evaluation is tracked by the vehicle and is updated in a receding horizon manner when new images are received from the camera. It can run in real time on a small embedded computer on board. We validated its effectiveness through experiments in indoor and outdoor environments. Compared to the perception-agnostic planner in the previous chapter, this planner kept more features in the camera's view and made the flight less aggressive, making the VIO more accurate. It also reduced VIO failures, which occurred for the perception-agnostic planner but not for this perception-aware planner

In Chapter 6, we introduce an inertial navigation motion planning strategy, which can be used for the state estimation of a quadcopter if other sensors fail. In some challenging environments, such as inside buildings on fire, the main sensors (e.g. cameras, LiDARs and GPS systems) used for multicopter localization can become unavailable. In contrast, direct integration of the inertial navigation sensors (the accelerometer and rate gyroscope), is usually unaffected by external disturbances. However, the rapid error accumulation quickly makes a naive application of such a strategy feasible only for very short durations. In this chapter, we propose a motion strategy for reducing the inertial navigation state estimation error of multicopters. Our strategy breaks a long-duration flight into multiple short-duration hops between which the vehicle remains stationary on the ground. When the vehicle is stationary, zero-velocity pseudo-measurements are introduced to an extended Kalman Filter (EKF) to reduce the state estimation error, which greatly reduces the inertial navigation's uncertainty. Indoor and outdoor experiments are conducted to show this strategy's effectiveness.

Finally, Chapter 7.2 summarizes the methods and results presented in this thesis.

The materials in this thesis are based on the following papers:

- Andrea Tagliabue, Xiangyu Wu, and Mark W. Mueller. "Model-free Online Motion Adaptation for Optimal Range and Endurance of Multicopters". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5650–5656

- X. Wu and M. W. Mueller. "In-flight range optimization of multicopters using multivariable extremum seeking with adaptive step size". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1545–1550

- Xiangyu Wu et al. "Model-free online motion adaptation for energy efficient flights of multicopters". In: *arXiv preprint arXiv:2108.03807* (2021)

- Junseok Lee[1], Xiangyu Wu[1], Seung Jae Lee and Mark W. Mueller. "Autonomous

[1]Junseok Lee and Xiangyu Wu contributed equally to this article. Names are in alphabetical order.

flight through cluttered outdoor environments using a memoryless planner". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 1131–1138

- Xiangyu Wu et al. "Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry". In: *arXiv preprint arXiv:2204.03134* (2022)
- Xiangyu Wu and Mark W. Mueller. "Using multiple short hops for multicopter navigation with only inertial sensors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 8559–8565
- Jiaming Zha, Xiangyu Wu, Ryan Dimick, and Mark W. Mueller. "A collision-resilient aerial robot design with an icosahedron tensegrity shell" . (in preparation)

# Chapter 2

# Flight speed adaptation for energy efficient flight

Limited flight time and distance constitutes a major bottleneck for the widespread adoption of multicopters. In this chapter, we introduce an approach that allows a quadcopter to find the flight speed which maximizes its flight time (endurance) or flight distance (range) while moving along a given path, using on-board power measurement. The proposed strategy is based on Extremum Seeking Control and (a) does not require any model of the power consumption of the system, (b) can be executed on-line, and (c) adapts to unknown payloads. We show in the experiments that hovering is not the most energy-efficient loitering strategy, and we demonstrate the proposed method's ability to adapt to different aerodynamic disturbances, such as payloads, while flying at the optimal range speed along a circular path. The method may be especially useful in applications where a quadcopter carries an unknown payload, allowing it to adapt for improved range.

Note that the material in this chapter is based on the following previously published work.

- Andrea Tagliabue, Xiangyu Wu, and Mark W. Mueller. "Model-free Online Motion Adaptation for Optimal Range and Endurance of Multicopters". In: *2019 International Conference on Robotics and Automation (ICRA).* 2019, pp. 5650–5656

## 2.1 Introduction

Multicopters are gaining increasing interest as tool for critical, real-world, outdoor applications such as search and rescue [10], inspection [11] and transportation [12]. Due to the relatively simple and inherently redundant mechanical design, their popularity is also raising for manned transportation [13] and space exploration [14], [15] applications. However, the limited flight time and distance of most of the available platforms [16] severely constraint their range of applications.

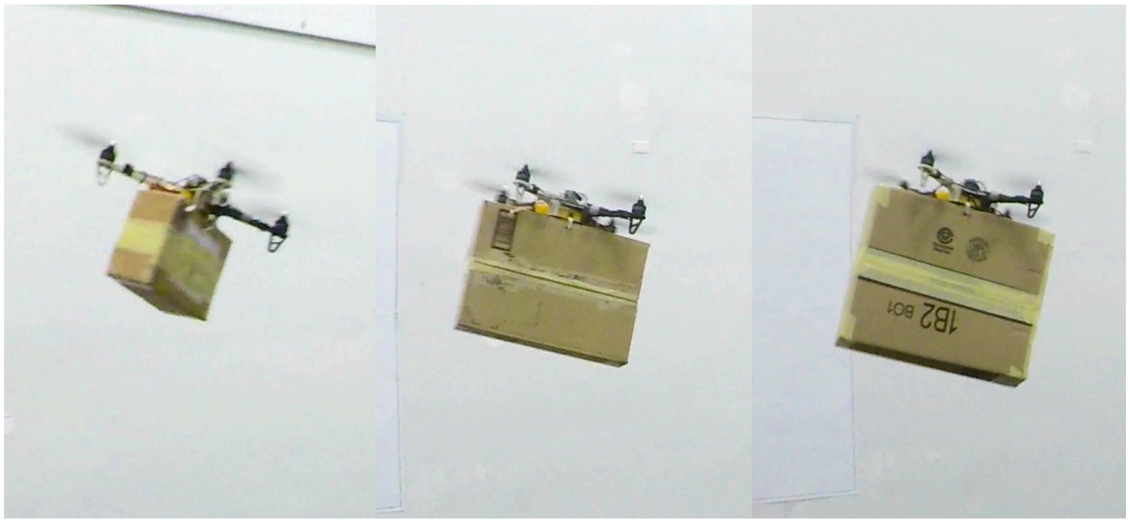


Figure 2.1: A quadcopter carrying different payloads with similar mass but different size. The proposed control scheme allows to find the speed that maximizes the flight distance of the vehicle along a given path. It adapts to unknown disturbances, such as the aerodynamic interference from a payload.

A possible solution to limited flight autonomy is the deployment of novel designs, such as VTOL platforms [17], [18], tethered multicopters [19] and hybrid solutions [20]. For existing platforms, efficiency can be improved via hardware optimization (e.g. by reducing the weight) or via algorithm-based optimization.

Algorithm-based optimization offers multiple opportunities for the improvement of efficiency of aerial machines, as it is easy to implement, economic to deploy, and can be used to complement mechanical designs, gaining insights for novel hardware platforms. Algorithmic improvements can be achieved via a model-based or a model-free approach. A model-based approach (e.g. [21], [22], [23]) allows to fully exploit the capabilities of the system, but relies on the ability to derive and identify an adequate model of the power consumption of a multicopter. Such a model is usually focused on capturing the electrical power losses [24], [25], [26], or the aerodynamic power losses [27] [28], [29] of the robot. A model-free approach (e.g. [30]), instead, allows to better take into account hard-to-model, less known effects, such as changes in performance due to aging of the components, or changes in the aerodynamic due to payloads.

In this chapter we present an on-line, model-free, adaptive approach to find the velocity which maximizes the total flight time (endurance or loitering time) or flight distance (range) of a quadcopter, using Extremum Seeking (ES) control. ES control is peak-finding technique and it is thoroughly described in e.g. [31]. It has found a relatively wide usage in robotics, as detailed in the literature survey [32]. Its applications include aeronautics, where it is employed to increase power efficiency in formation flight [33]. In our chapter, ES control is used to minimize derived cost functions which express the endurance and range of the robot as a function of its velocity, given a fixed energy budget. The proposed scheme autonomously sets the reference velocity along a predefined path according to the chosen cost function. By flying along a circular path with a quadcopter, we show that the algorithm allows to find the optimal, non-zero, loitering velocity. We demonstrate that our approach successfully adapts the reference velocity to the optimal range velocity in multiple flight scenarios, such as the transportation of the different payloads shown Fig. 2.1. In addition, we compare our experimental results with the predictions from a derived dynamic model of the power consumption of a quadcopter.

## 2.2 Dynamic Model of a Quadcopter UAV

In this section we derive the dynamic model of a quadcopter that takes into account the electrical power consumption, as measured at the terminals of the on-board battery. Such a model allows to justify some of the counterintuitive properties of the system observed in our experimental results (e.g. power consumption is not monotonically increasing w.r.t. velocity), and allows to make predictions on the behavior of the system beyond the capabilities of our experimental testbed.

### Reference frame definition

As shown in Fig. 2.2, we define two sets of coordinate frames: an inertial frame $I$ and a non-inertial frame $B$, attached to the Center of Mass (COM) of the quadcopter.

### Quadcopter dynamics

The quadcopter is modeled as a rigid body with six degrees of freedom. Its translational and rotational dynamics are described by the following set of Newton-Euler equations:

$$m\ddot{\boldsymbol{x}} = m\boldsymbol{g} + \boldsymbol{R}\sum \boldsymbol{f}_i + \boldsymbol{f}_d \tag{2.1}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\![\boldsymbol{\omega}]\!] \tag{2.2}$$

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} + \sum \boldsymbol{\tau}_i \tag{2.3}$$

The vector $\boldsymbol{x}$ and its derivatives express the vehicle's translational position, velocity and acceleration in the inertial reference frame $I$, while $\boldsymbol{\omega}$ and its derivative define the angular velocity and acceleration in the body-fixed frame $B$. Each propeller $i$ ($i = 1, ..., 4$), produces a thrust force $\boldsymbol{f}_i = (0, 0, f_i)$ and a torque $\boldsymbol{\tau}_i = (0, 0, \tau_i)$, expressed in $B$. We additionally introduce the rotation matrix $\boldsymbol{R}$, which relates the body-fixed and world-fixed frames, and the gravity vector $\boldsymbol{g} = (0, 0, -g)$, expressed in $I$. The vector $\boldsymbol{f}_d$, also expressed in $I$, represents the drag force modelled as an isotropic drag, which is the sum of a linear and a quadratic

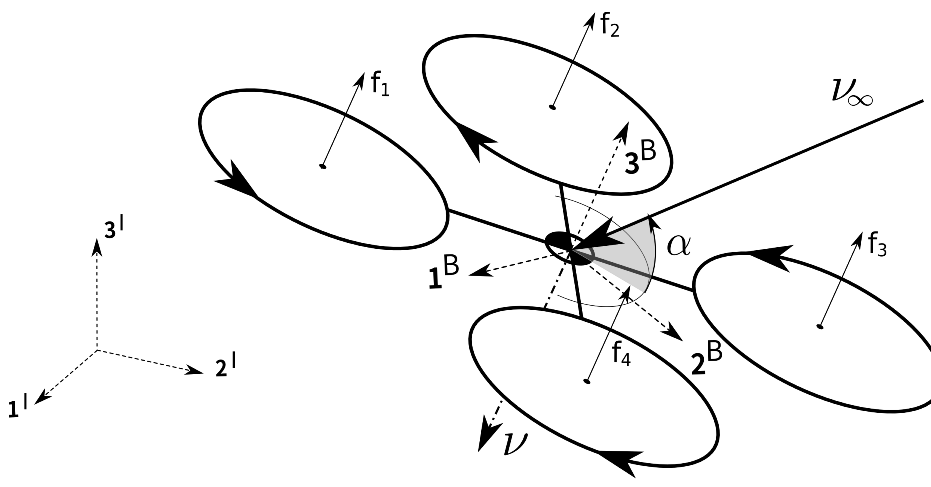


Figure 2.2: Coordinate frame definition. $I$ represents the inertial reference frame and $B$ the quadcopter reference frame. We additionally show the thrust force of the $i$-th propeller $\boldsymbol{f}_i$, the freestream velocity $\boldsymbol{\nu}_\infty$, the induced velocity $\boldsymbol{\nu}$ and the angle of attack $\alpha$, shown positive in the diagram.

term (see e.g. [34]):

$$\boldsymbol{f}_d = -\left(\mu_1 \nu_\infty + \mu_2 \nu_\infty^2\right) \frac{\boldsymbol{\nu}_\infty}{\nu_\infty} = f_d \frac{\boldsymbol{\nu}_\infty}{\nu_\infty}, \quad \nu_\infty = \|\boldsymbol{\nu}_\infty\| \tag{2.4}$$

where $\boldsymbol{\nu}_\infty$ corresponds to the freestream velocity expressed in $I$. The scalars $\mu_1$ and $\mu_2$ represent the drag coefficients and can be identified experimentally. We additionally assume that the drag force acts on the COM of the multicopter and thus no torques are produced.

## Power losses

Following [16], [21], [30], we assume that the total power consumption $p$ (measured at the terminals of the battery) is proportional to the *aerodynamic induced power* [29] $p_{\text{induced}}$:

$$p = \frac{1}{\eta} p_{\text{induced}} \tag{2.5}$$

where $\eta$ lumps the conversion losses in the energy flow from the battery to the propellers and can be obtained experimentally. Assuming constant altitude and forward flight and given $f_{\text{thrust}} = f_1 + ... + f_4$, $p_{\text{induced}}$ is computed as [29]:

$$p_{\text{induced}} = \sum_{i=1}^{4} p_{\text{induced},i} = \sum_{i=1}^{4} \kappa \left(\nu + \nu_\infty \sin\alpha\right) f_i = \kappa \left(\nu + \nu_\infty \sin\alpha\right) f_{\text{thrust}} \tag{2.6}$$

where $\nu$ represents the induced velocity applied by the propeller to the surrounding air. The angle of attack $\alpha$ is defined as the angle between $\boldsymbol{\nu}_\infty$ and the plane given by $\mathbf{1}^B$ and $\mathbf{2}^B$, as represented in Fig. 2.2. For simplicity we have assumed the angle of attack $\alpha$, the induced velocity $\nu$ and the freestream flow $\boldsymbol{\nu}_\infty$ to be the same for every propeller, neglecting effects such as changes in freestream velocity due to non-zero angular rates $\boldsymbol{\omega}$. The scalar $\kappa$ is an empirical correction factor, and can be lumped in the conversion efficiency factor $\eta$. The induced velocity $\nu$ is implicitly defined as [29]:

$$\nu = \frac{\nu_h^2}{\sqrt{(\nu_\infty \cos\alpha)^2 + (\nu_\infty \sin\alpha + \nu)^2}}. \tag{2.7}$$

The induced velocity at hover $\nu_h$ is obtained from:

$$\nu_h = \sqrt{\frac{mg/4}{2\rho\pi r^2}} \tag{2.8}$$

where $\rho$ is the density of the air and $r$ is the radius of the propellers. Equation (2.7) can be solved for $\nu$ using numerical techniques such as the Newton-Raphson [35].

## 2.3 Model-Free Adaptive Control

In this section we introduce the model-free approach which allows to identify optimal range and endurance velocities of a multicopter, along a predefined reference path. Small changes in the vehicle's velocity can affect the power consumption and thus the autonomy of the vehicle, as illustrated in Fig. 2.3. Identification via model-based approaches is not always possible, as optimal range and endurance speed depend on multiple electromechanical and aerodynamic properties of the robot, like the thrust to lift ratio of propellers, the drag on the fuselage (e.g. due to payloads), the efficiency of the electric motors and the efficiency of the electronic speed controllers. Unknown or un-modeled disturbances, including wind, aging of the components or payloads, can further affect these optimal operating points. An on-line, adaptive approach seems therefore especially suitable for this task. The proposed scheme based on Extremum Seeking control is detailed in the following paragraphs. A main assumption of our method is that information about the instantaneous power consumption of the vehicle is available, for example by sensing the voltage and current at the battery.

### Extremum Seeking Controller

As shown in Fig. 2.4, ES control allows to find an unknown, time-varying plant operating point $r^*(t)$ which maximizes or minimizes a given plant output $q(t)$. The optimal setpoint is found by applying a small periodic perturbation $a\sin(\omega_d t)$ to the current reference setpoint $\hat{r}(t)$ and by monitoring the changes of the plant's output at the given disturbance frequency $\omega_d$. The scalar $a$ defines the magnitude of the perturbation, while the disturbance frequency

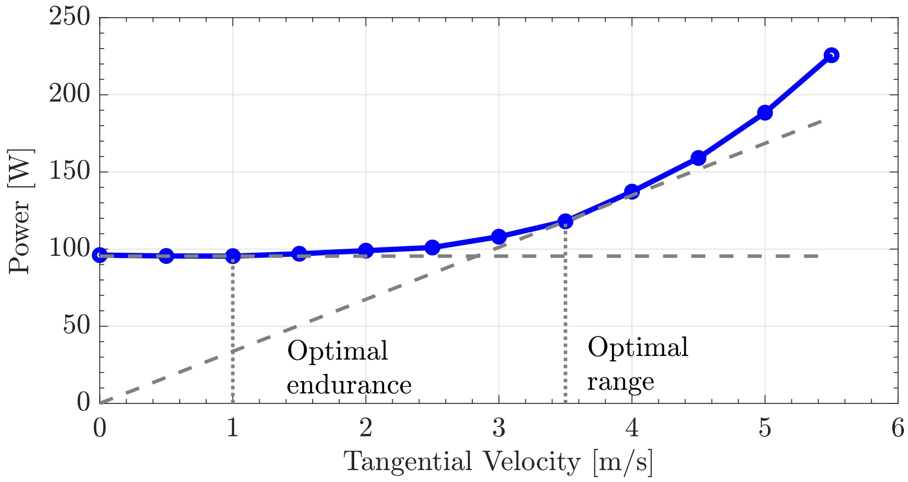


Figure 2.3: Power-velocity curve obtained by flying a quadcopter at different velocities along a horizontal circular path of radius $r = 1.7$ m. We have highlighted the optimal endurance velocity, corresponding to the speed that minimizes the electrical power consumption, and the optimal range velocity, corresponding to the speed that minimizes the ratio between power and velocity. According to the proposed model, the power consumption at optimal endurance velocity further decreases by flying along a straight path.

$\omega_d$ is set at a value sufficiently small, so that the plant can be considered a static map. If perturbed plant's input $r(t) = \hat{r}(t) + a\sin(\omega_d t)$ and output $q(t)$ are in phase (i.e. input grows, output grows) then the reference setpoint $\hat{r}(t)$ is decreased (assuming that we are minimizing the cost function). If they are out of phase, $\hat{r}(t)$, which corresponds to the current estimate of the optimal operating point, is increased. The persistent nature of the input perturbation $a\sin(\omega_d t)$ allows to adapt to time-varying systems. A proof of convergence and further details are provided in [31].

In the context of maximizing the range or endurance of a quadcopter, we employ an ES controller to set the reference tangential speed of the vehicle along a desired path. We define suitable cost functions which relates range (distance flown) and endurance (time flown) to the speed of the vehicle and its instantaneous power consumption. The estimate of the optimal reference tangential velocity, output of the ES controller, is then used to parametrize the desired path into a trajectory, fed as input to the position and attitude controller of the vehicle. A diagram of the system architecture can be found in Fig. 2.5.

## Cost function derivation

In this section we derive two cost functions, one which relates the velocity of flight of a multicopter with its range distance and one with its endurance time. We assume that it is given a constant energy budget, such as the energy stored in the on-board battery

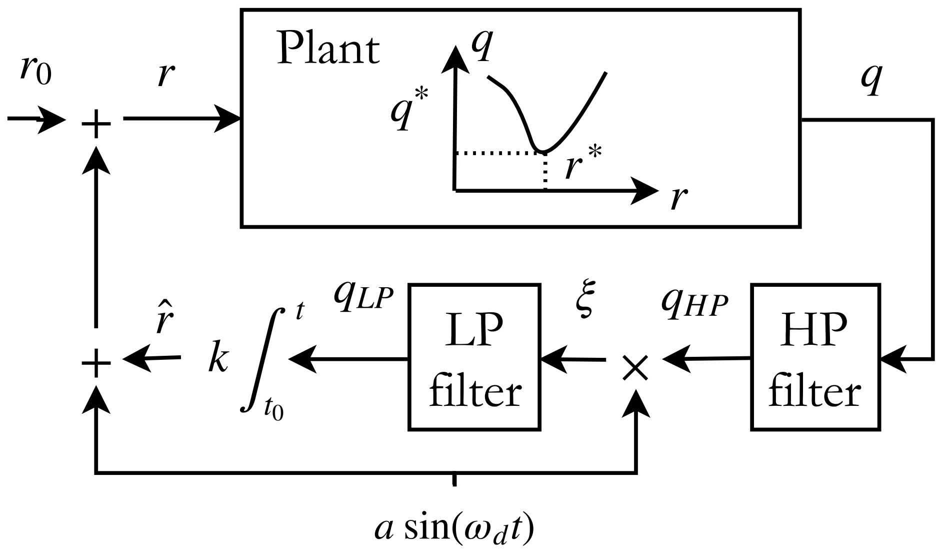


Figure 2.4: Block diagram of a feedback scheme based on Extremum Seeking (ES) control. The scalar $r_0$ represents the plant's initial setpoint. The frequency of the HP and LP filters is set, respectively, to $\omega_{\mathrm{HP}}$ and $\omega_{\mathrm{LP}}$. The scalar $k$ is a tuning parameter of the controller: setting $k > 0$ allows to maximize the output of the plant, while $k < 0$ minimizes the output.

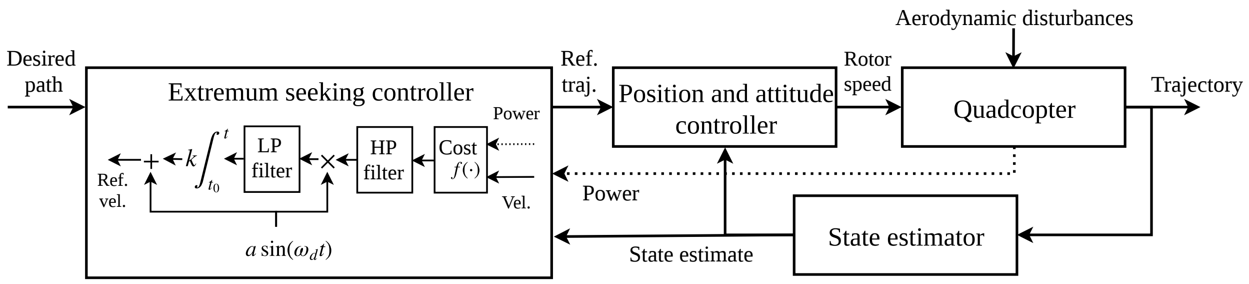


Figure 2.5: System diagram employed for on-line optimization of the range or endurance of a quadcopter UAV. Given a desired path, such as a circle of constant altitude and radius, the Extremum Seeking (ES) controller generates a reference trajectory which is tracked by the UAV via the position and attitude controller. The reference trajectory is generated by defining a reference velocity along the desired path. The ES controller automatically sets the reference velocity which minimizes a given cost function $f(\cdot)$, according to the information provided by the motion capture system about the estimated velocity of the UAV and the current and voltage measurements from the sensors mounted on-board.

$E \in [E_{\text{empty}}, E_{\text{full}}]$. We additionally assume that the vehicle (a) is moving at steady state, with constant ground velocity $v_{\text{ground}}$, (b) is using a constant power $p$, and (c) is maintaining a constant altitude and orientation along the reference path.

### Endurance mode

The endurance time $t_{\text{endurance}}$ is defined as:

$$t_{\text{endurance}} := \int_{t_0}^{t_{\text{end}}} dt = \int_{E_{\text{empty}}}^{E_{\text{full}}} \frac{1}{p} dE = \frac{1}{p}\Delta E \tag{2.9}$$

by considering that $t_0$ and $t_{\text{end}}$ represent initial and final time of the mission, corresponding respectively with the full and empty states of the on-board battery $E_{\text{full}}$ and $E_{\text{empty}}$, with $\Delta E = E_{\text{full}} - E_{\text{empty}}$. From (2.9), and given that the total energy in the battery $\Delta E$ is constant, follows that:

$$\max(t_{\text{endurance}}) \Leftrightarrow \max\left(\frac{1}{p}\right) \Leftrightarrow \min(p) \tag{2.10}$$

### Range mode

The range distance $d_{\text{range}}$ is defined as:

$$d_{\text{range}} := \int_{t_0}^{t_{\text{end}}} v_{\text{ground}}\ dt = \int_{E_{\text{empty}}}^{E_{\text{full}}} \frac{v_{\text{ground}}}{p} dE = \frac{v_{\text{ground}}}{p}\Delta E \tag{2.11}$$

From (2.11) follows that:

$$\max(d_{\text{range}}) \Leftrightarrow \max\left(\frac{v_{\text{ground}}}{p}\right) \Leftrightarrow \min\left(\frac{p}{v_{\text{ground}}}\right). \tag{2.12}$$

### Stability and performance considerations

**Stability** convergence of the ES control scheme is guaranteed only if the magnitude of the additive disturbance $a\sin(\omega_d t)$ on the reference velocity is sufficiently small [31], and $\omega_d$ is sufficiently smaller than the natural frequency of the quadcopter. Tuning of the controller requires then to identify three different time scales in the controlled system:

1. A “fast” time scale, defined by the dominant (slowest) dynamics of the plant. In our case this value is set to the dominant pole of the position controller of the quadcopter.
2. A “medium” time scale, which corresponds to the frequency $\omega_d$ of the disturbance of the ES controller.
3. A “slow” time scale, which correspond to the dominant frequency of the high-pass and low-pass filters employed in the ES controller.

**Performance** while stability requirements significantly limit the convergence speed of the controller, they imply that the action of the persistent disturbance output by the ES controller has little effect on the worsening of the power consumption, as the accelerations introduced by the periodic perturbation are small. Furthermore, we observe that due to the persistent input perturbation, which guarantees time-varying adaptation, the controller only converges to a neighborhood of the optimal setpoint. Convergence time, in addition, is limited by the speed of the dominant frequency of the plant (for the time scale separation requirement).

## 2.4 Experimental results

In this section we present the experimental results to validate the effectiveness of the proposed on-line, optimal velocity finding approach. The results show that the algorithm is able to find the optimal range and endurance velocities despite unknown disturbances and starting from different initial velocities. In addition, we present how to identify the parameters of the power model proposed in Section 2.2 and validate the model.

### Experimental setup

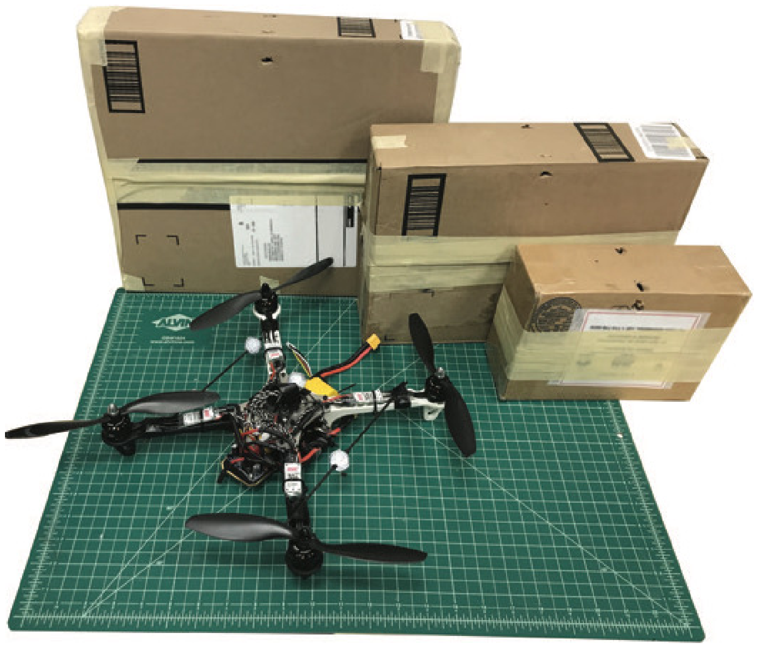


Figure 2.6: Hardware used for the experimental results.

The vehicle used throughout the experimental results is a custom-built quadcopter, shown in Fig. 2.6, where we also show the payloads used for validation of the proposed approach. The on-board attitude controller runs at 500 Hz on a Bitcraze Crazyflie [36] electronic board with a modified version of the PX4 firmware [37]. Position and other controllers run off-board, sending commands to the vehicle via a radio link at 50 Hz. The experiments are executed indoor, using a commercial motion capture system for the localization of the vehicle. Due to the size of the flight space, we fly along circular paths with a maximum radius of approximately 2.15 m.

## Model identification and validation

In this section we validate the modeling assumptions presented in Section 2.2. We identify the model parameters and compare the predicted and measured power consumption by flying along a horizontal circular trajectory at different velocities. We assume that the efficiency

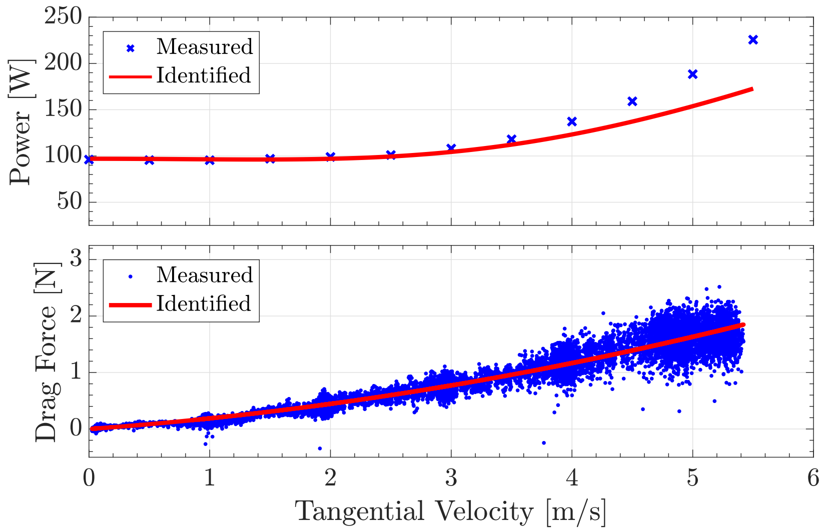


Figure 2.7: *(Top:)* Power-velocity curve measured and predicted by our model. *(Bottom)* Drag force as a function of the velocity and model prediction with the identified parameters. The identification setup has been obtained by flying a quadcopter without payload along a horizontal circular path of radius $r = 1.7$ m.

of the powertrain and propellers is lumped in the parameter $\eta$, which is identified as:

$$\eta = \frac{\hat{p}_h}{mg\nu_h} \tag{2.13}$$

where $\hat{p}_h$ corresponds to the measured electrical power consumption at hover, obtained via the on-board voltage and current sensor. The induced velocity at hover $\nu_h$ is computed according to (2.8). The drag coefficients are identified by flying at different velocities along a circular trajectory with constant altitude and are obtained according to:

$$\hat{f}_{\text{thrust}} = -\frac{m}{\cos\phi\cos\theta} g \tag{2.14}$$

$$\hat{f}_{\text{drag}} = (\hat{f}_{\text{thrust}} \boldsymbol{R3}^I) \cdot \boldsymbol{e}_{\nu_\infty} - m\dot{\nu}_\infty. \tag{2.15}$$

where for simplicity and to reduce the effects of noise we have assumed that the vehicle only moves horizontally. The angles $\phi$ and $\theta$ correspond, respectively, to the roll and pitch of the vehicle. From the identification experiment, we obtain that the powertrain efficiency $\eta \approx 0.310$, the linear drag coefficient $\mu_1 \approx 0.153$ $\mathrm{N\,m^{-1}\,s}$ and the quadratic drag coefficient $\mu_2 \approx 0.035$ $\mathrm{N\,m^{-2}\,s^2}$. A comparison of the estimated and measured power consumption, as well as the measured and identified drag, is shown in Fig. 2.7, where we fly along a circular trajectory of radius 1.7 m at high speed. We remark that the validation dataset is different from the one used for identification of the parameters of the model. We can observe that the model predicts the power consumption well up to about 3.5 $\mathrm{m\,s^{-1}}$, and then tends to underestimate the power demand, potentially due to the simple modeling assumption regarding the electrical power losses.

## Optimal range and endurance velocities along a circular path

In this part we present the experimental results from the online peak-finding scheme based ES control, which is used to find the optimal range and endurance velocities of a quadcopter flying a circular path. As described in Section 2.3, the magnitude of the reference disturbance $\omega_d$ is set to 0.2 $\mathrm{rad\,s^{-1}}$, which is about one decade slower than the closed-loop dynamic of the position controller, whose dominant frequency is set to 2.0 $\mathrm{rad\,s^{-1}}$. We empirically found that a good value for the amplitude of the disturbance $a$ is 0.15 $\mathrm{m\,s^{-1}}$. If faster convergence speed is required and a larger disturbance can be tolerated, the magnitude of disturbance $a$ can be slightly increased.

### Optimal range velocity

Convergence to optimal range velocities is demonstrated by flying a quadcopter of mass 0.665 kg with different payloads along a horizontal circular path of radius 1.7 m. The employed cost function corresponds to minimize the expression derived in (2.12), where the power measurement $p$ is provided by on-board sensors and the velocity $v_{\text{ground}}$ is obtained from the output of the state estimator. The ES controller is tuned so that the gain $k$ is set to $-1$, and the cutoff frequencies of the HP filter $\omega_{\text{HP}}$ and LP filter $\omega_{\text{LP}}$ are set to 0.1 $\mathrm{rad\,s^{-1}}$.

Experiments were executed using different cardboard boxes acting as payloads, and with no payload. The box was oriented so that its largest surface was facing the direction of motion of the vehicle. By varying the size of the box while maintaining approximately the

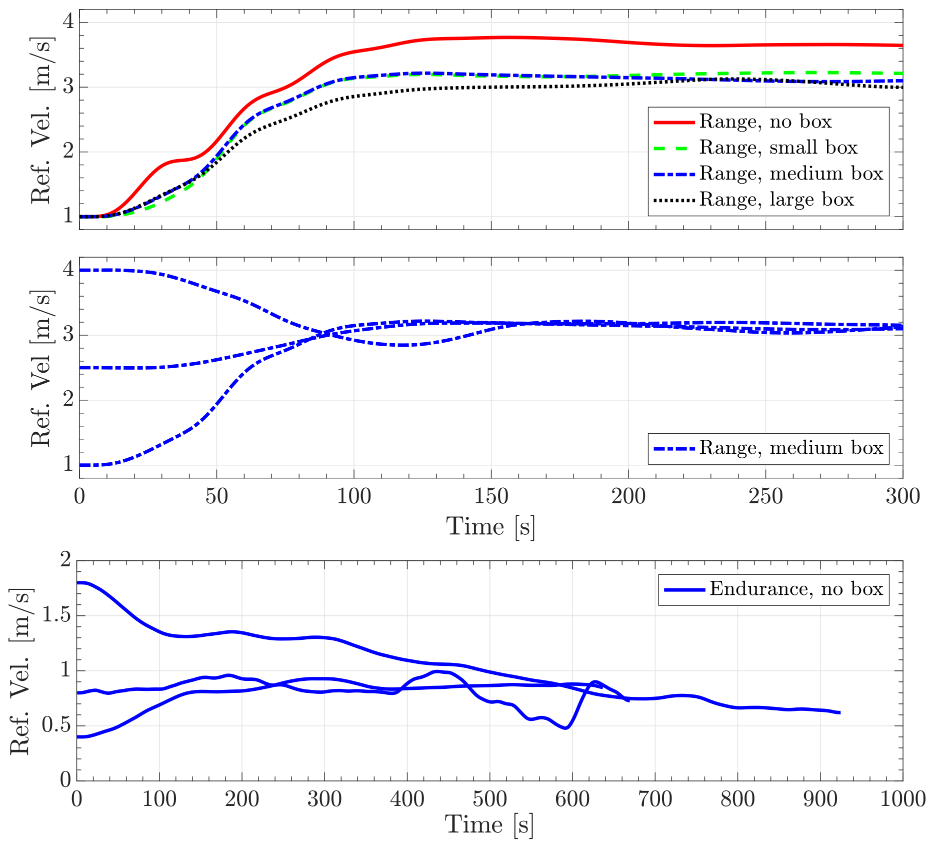


Figure 2.8: Experimental results of the convergence of the Extremum Seeking (ES) controller. In the first plot we observe that the optimal range velocity differs due to the aerodynamic properties of the transported payload ($\approx 3.0\ \mathrm{m\,s^{-1}}$ for large box, $\approx 3.1\ \mathrm{m\,s^{-1}}$ for medium, $\approx 3.2\ \mathrm{m\,s^{-1}}$ for small, and $\approx 3.6\ \mathrm{m\,s^{-1}}$ with no box). In the second and third plot we observe that optimal range and endurance velocity converge despite the different initial values. The optimal endurance velocity is non-zero due to the effect of the induced power consumption. The changes in power and energy per meter for the different scenarios are represented in Fig. 2.10 and Fig. 2.9 respectively.

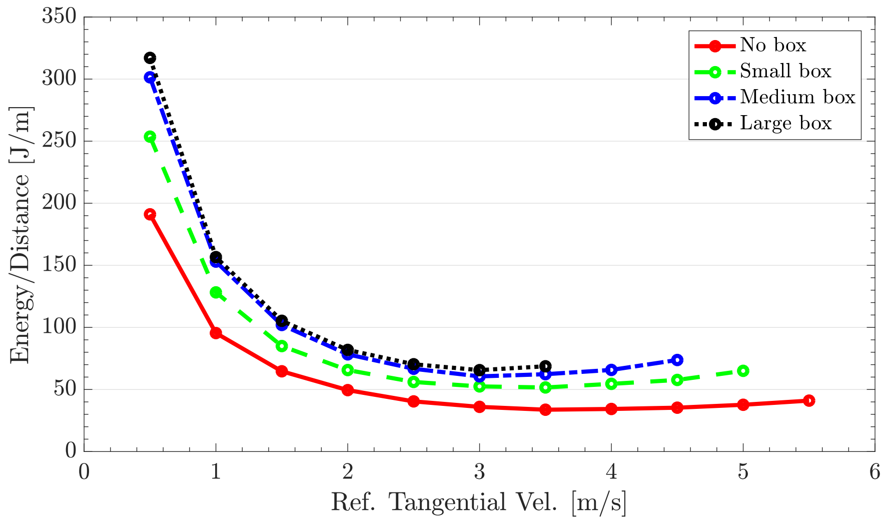


Figure 2.9: Cost function, expressed as energy/disitance, used by the ES controller to find the optimal range velocity of a quadcopter. The cost function presents a minimum at ≈ 3.0 $\mathrm{m\,s^{-1}}$ when a cardboard box is attached to the quadcopter, and a minimum at ≈ 3.5 when no payload is attached. These results are in agreement with the convergence velocities shown in Fig. 2.8 (first and second rows).

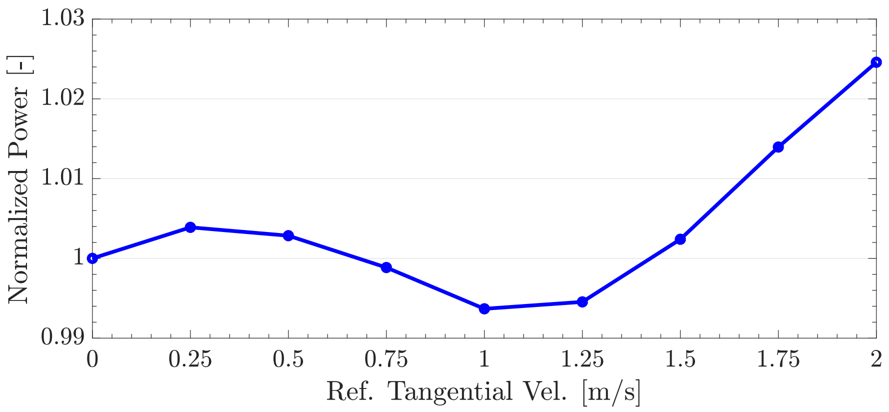


Figure 2.10: Normalized power of the vehicle flying at different tangential velocities (with no box), which corresponds to cost function used by the ES controller to find the optimal endurance velocity.

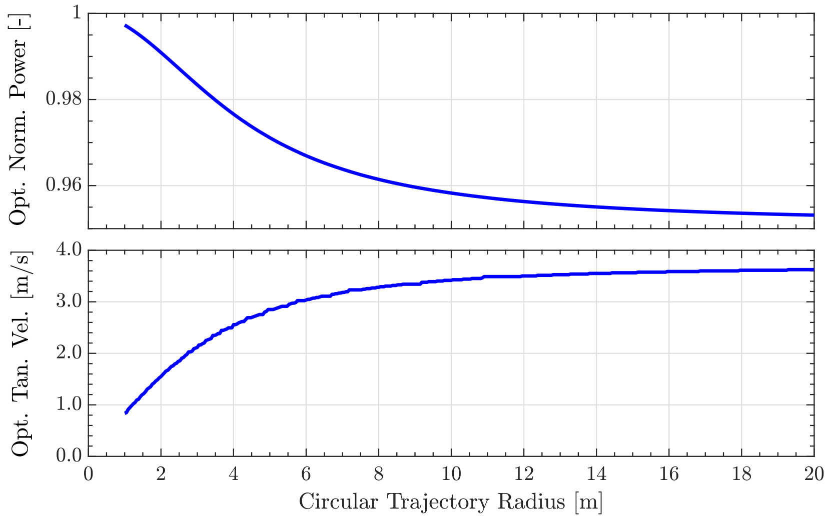


Figure 2.11: Optimal endurance velocity and hover-normalized power consumption as a function of the trajectory radius for a quadcopter of mass $m = 0.695$ kg and identified drag coefficients, estimated according to the proposed lumped model. We can observe that the proposed model predicts a minimum power consumption of approximately 95% of the value at hover for circular paths of radius larger than approx. 10 m.

same weight of 0.2 kg, we are able to show convergence to different velocities. Fig. 2.1 shows photographs of the vehicle with the boxes, as used in the experiment.

The experimental results are shown in the first and second rows of Fig. 2.8, where we display the estimate of the optimal reference tangential velocity, as computed by the ES controller. In order to verify that the reference velocity converges to the optimal value, in a separate experiment we fly for 60 s at different tangential velocities, along a circular path of the same radius as before, and we record the value assumed by the employed cost function. The results are shown in Fig. 2.9.

By comparing the convergence velocities of the ES controller in the first row Fig. 2.8 with data in Fig. 2.9, we observe that the proposed method is able to find the optimal range velocities despite the difference in payloads. In addition, the second row of Fig. 2.8 shows that the method is able to find the optimal range velocity despite starting from different initial velocities.

### Optimal endurance velocity

Similar to the optimal range case, convergence to the optimal endurance velocity is shown by flying along a circular path of radius 2.15 m at a fixed altitude, using a quadcopter of 0.695

kg and no box attached. The employed cost function corresponds to minimizing the power measured on-board, as derived in (2.10). The gain $k$ of the ES controller is set to $-1$, while $\omega_{\text{HP}} = \omega_{\text{LP}} = 0.02$ rad/sec. The convergence results are displayed in the third row of Fig. 2.8, where we plot the estimate of the optimal reference tangential velocity, output of the ES controller (without sinusoidal disturbance). For comparison, Fig. 2.10 represents the value of the employed cost function, obtained by flying for 20 s at different tangential velocities along the same circular path. In this case we observe that the minimum of the cost function, which corresponds to a reduction of approximately 1% of the power at hover, is reached for a non-zero reference tangential velocity, corresponding to about 1 $\text{m}\,\text{s}^{-1}$. Such effect is due to the reduction in induced power consumption for increasing freestream velocity, as detailed by [29] and also observed by [16], [21], and justifies why hovering is not the optimal loitering strategy. From Fig. 2.8 (third row) we can observe that the reference tangential velocity successfully converges to the minimum of the cost function measured in Fig. 2.10. From the derived model we can additionally observe that increasing the radius of the circular path helps to increase the autonomy of the robot. As shown in Fig. 2.11, our model predicts on improvement of approx. 5% of the power consumption at hover for paths with radius larger than 10 m.

## 2.5 Conclusion

In this chapter we have presented a method to find the velocity which maximizes the flight distance (range) or time (endurance) of a multicopter, given a desired path. Experiments show that the proposed approach is able to converge to the optimal velocity independent of the initial speed of the robot. By varying the aerodynamic drag of the vehicle with different payloads, we additionally show that our method allows to adapt the optimal range velocity to unknown disturbances. From our modeling efforts and experimental results we observed that, for circular paths with sufficiently large radius, the total power consumption as a function of the freestream velocity of the multicopter is not monotonically increasing, but presents a minimum for non-zero velocity. This means that the optimal loitering strategy is not hovering, but rather flying with some velocity along a straight line or along a circular trajectory of sufficiently large radius. Our experiments achieved a repeatable improvement w.r.t. the electrical power consumption at hover (Fig. 2.10), while flying along a circular path. Our model predicts further improvements as the radius of the path increases (Fig. 2.11), but we could not verify the consumption along circular paths with larger radius due to the limited space available.

## Acknowledgements

The research in this chapter was supported by the Zeno Karl Schindler Foundation (ZKSF) Master Thesis Grant, Swiss Friends of the United States Association (SFUSA) Study Grant and the CHSR Corporation. The experimental testbed at the HiPeRLab is the result of contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

# Chapter 3

# Fast adaptation of speed and sideslip for energy efficient flight

This chapter is a follow up of the previous chapter, and extends the adaptation of the flight speed to both the speed and sideslip of the vehicle. By adding a step size adapter to standard extremum seeking controller, the convergence speed is also improved.

In this chapter propose a method for finding the optimal speed and sideslip angle of a multicopter flying a given path to achieve either the longest flight distance or time. Since flight speed and sideslip are often free variables in multicopter path planning, they can be changed without changing the mission. The proposed method is based on a novel multivariable extremum seeking controller with adaptive step size, which is inspired by recent work from the machine learning community on stochastic optimization. It (a) does not require a power consumption model of the vehicle, (b) is computationally efficient and runs on low-cost embedded computers in real-time, and (c) converges faster than the standard extremum seeking controller with constant step size. We prove the stability of this approach and validate it through both indoor and outdoor experiments. The method is shown to converge with different payloads and in the presence of wind.

Note that the material in this chapter is based on the following papers. It is primarily based upon the latter paper, which is an extension of the former paper.

- X. Wu and M. W. Mueller. "In-flight range optimization of multicopters using multivariable extremum seeking with adaptive step size". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1545–1550

- Xiangyu Wu et al. "Model-free online motion adaptation for energy efficient flights of multicopters". In: *arXiv preprint arXiv:2108.03807* (2021)

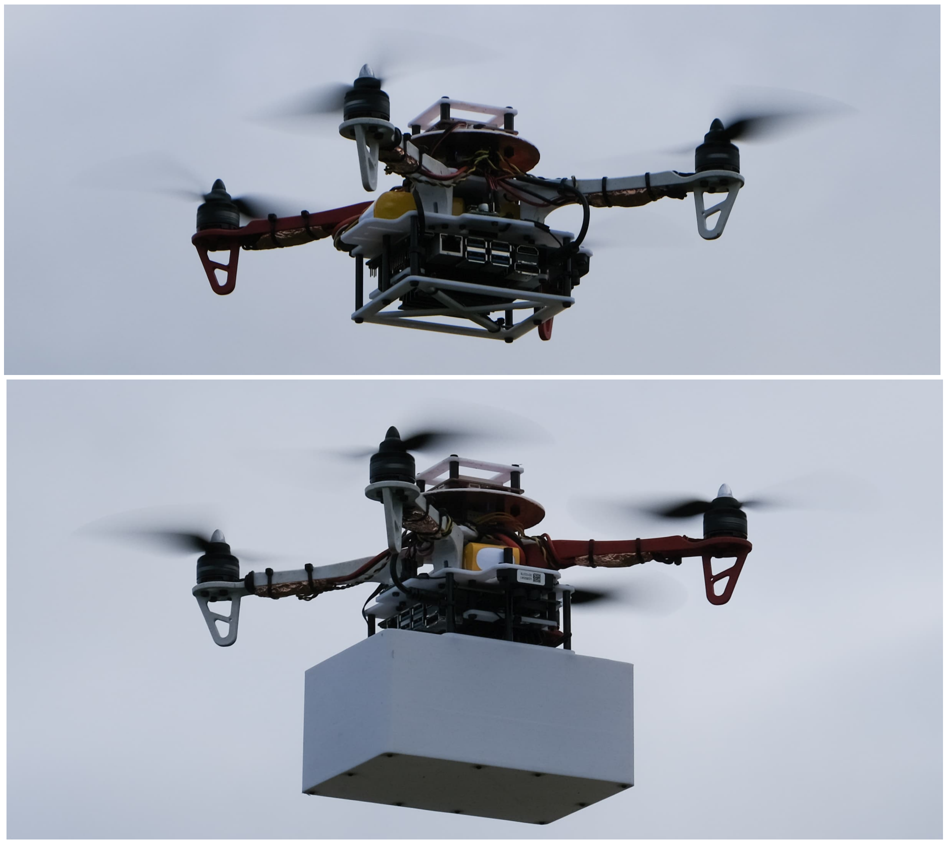


Figure 3.1: A quadcopter with and without a box payload with unknown aerodynamics effects, as used in the outdoor experiments.

## 3.1 Introduction

Multicopters are used in a wide range of applications such as aerial photography [2], transportation [4], search and rescue [10], inspection [3], and agriculture [38], thanks to their low cost, ease of control, and high maneuverability. However, a primary limitation for current vehicles is their limited flight endurance and range [16].

One way to improve the limited flight range or endurance problem is through energy-efficient mechanical design. For example, in [39] a triangular quadcopter with one large rotor for lifting and three small rotors for control was proposed, which has the advantage of combining the energy efficiency of the large rotor and the fast control response of the small rotors. In [40], the authors designed a quadcopter with slightly tilted motors which has a better control authority over the yaw. This results in a lower variance in motor forces for yaw control. Because a motor's power is a convex function of its thrust, this design helps to reduce the total power consumption of the motors. Hybrid quadcopters which are able to do both aerial and ground locomotion, were introduced in [20] and [14]: when the vehicles operate in the ground locomotion mode on a flat ground, they only need to overcome the rolling resistance and use much less power compared to flying. A hybrid power system for multicopters consisting of a lithium battery, a fuel cell, and a hydrogen tank was introduced in [41], which enables longer flight time compared to traditional battery-only power systems, thanks to the higher specific energy of hydrogen compared with the lithium battery. In [42], an in-flight battery switching system was proposed, which enables a small quadcopter to dock an additional battery to a large quadcopter and increases its flight time.

Another category of methods focus on developing algorithms to reduce the power consumption of existing multicopters. By planning energy-efficient trajectories or by implementing energy-aware control algorithms, these approaches do not require design changes to existing hardware and are thus economical to deploy. For example, in [23] the authors proposed a method for finding the minimum-energy trajectory between a predefined initial and final state of a quadcopter, by solving an optimal control problem of the angular accelerations of the four propellers. This approach was extended in [43], where the fixed end-time trajectory optimization was extended to both free and fixed end-time solved with an indirect projected gradient algorithm to improve the numerical accuracy. Simulation results were shown to validate the effectiveness of the methods in both papers. In [22], the task of reaching a goal in a set of candidate goals while using the least amount of energy was investigated. The energy-efficient path planning algorithm was based on model predictive control and disturbance from wind was considered. The authors showed that their method was able to reach the goal which required the least amount energy in simulations and indoor experiments. In [44] and [45], the authors proposed energy-aware coverage path planning methods for photogrammetric sensing of large areas using multicopters. The methods find the optimal speed along the coverage path to minimize the energy usage during the mission. Outdoor experiments were conducted to validate their methods.

A necessary condition for model-based methods to perform well is accurate power consumption modeling. Power consumption models of multicopters can be derived by analyzing their electric and aerodynamic properties. For example, [24] [46] introduced power consumption models of the battery, electric speed controller and motor, and [29, Chapter 5] introduced the aerodynamic power consumption of the propeller based on the momentum theory. Besides, some researchers proposed data-driven models by selecting variables that affect the power consumption (e.g. the vehicle's speed and acceleration, wind speed, and payload weight) as inputs and finding their relationship to power consumption through experimental data [44] [47].

However, there are often hard-to-model effects on the vehicle's power consumption, such as changes in vehicle components' performance (e.g. batteries and motors) due to aging and temperature changes. In addition, the change in payload size, shape, or weight in applications such as package delivery and spraying (e.g., pesticide or fertilizer at farms) often requires reidentification of parameters in the power consumption model, which is time-consuming. The imperfections in the energy model could potentially be compensated using online data-driven methods. For example, in [48] the authors used an Extended Kalman Filter and in

[49] the authors used Gaussian processes to estimate the correction terms in the vehicle's dynamics equations, which improved the control accuracy of the quadcopters. However, to the best of our knowledge, no such methods have been developed for the energy efficient flight of quadcopters yet, and their effectiveness and computational efficiency are thus still an open question.

The aforementioned difficulties in quadcopter energy consumption modeling motivates us to propose a model-free method for finding the flight speed and sideslip angle (i.e., angle between the forward direction of the vehicle and the relative wind) which achieve the longest flight time (endurance) or flight range given a predefined path. The method is based on a novel multivariable extremum seeking controller and does not require power consumption models of the multicopter.

Extremum seeking control is a model-free adaptive control technique for finding the local minimizer of a given, potentially time-varying, cost function by applying a persistently exciting periodic perturbation to a set of chosen inputs, and monitoring the corresponding output changes. A survey of the development of this control method can be found at [50]. It has applications in areas such as maximizing the energy generation of wind turbines [51] and photovoltaic power plants [52], and maximizing the pressure rise in axial flow compressors [53]. Its applications in robotics can be found in a literature survey [32]. A common problem of extremum seeking controllers is their slow convergence speed, and we propose a novel multivariable extremum seeking controller with adaptive step size to improve it. In addition to the flight speed, it could also simultaneously find the optimal flight sideslip angle to achieve the longest flight range or endurance (time).

The major contributions of this chapter are as follows:

1. We present a model-free adaptive method to find the flight speed and sideslip angle of multicopters that achieve the longest flight range or endurance.

2. The method is based on a novel multivariable extremum seeking controller with adaptive step size, which is computationally efficient and converges faster than the standard extremum seeking.

3. We give a stability proof for the proposed controller via averaging and singular perturbation analysis.

4. We validate the effectiveness of the proposed method in extensive outdoor experiments. The experiments demonstrate the proposed method's faster convergence compared with the standard method, robustness to payloads and wind disturbances.

5. Outdoor experiments and analysis about the proposed method's performance under wind disturbances.

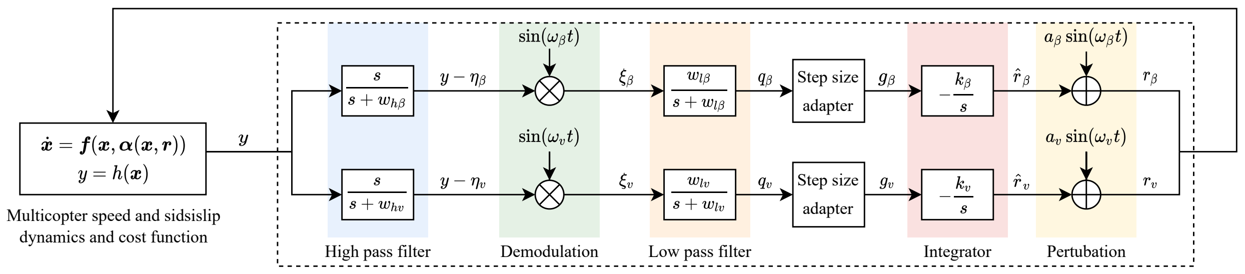


Figure 3.2: Block diagram of the adaptive step size multivariable extremum seeking controller (in the dashed rectangle). The goal of the controller is to find the optimal sideslip $r_\beta$ and $r_v$ to minimize the cost function $y = (h \circ l)(\boldsymbol{r})$. The frequencies of the high pass and low pass filters are set, respectively, to $\omega_{hv}$ and $\omega_{lv}$ for speed, and $\omega_{h\beta}$ and $\omega_{l\beta}$ for sideslip. The scalar $k_v$ and $k_\beta$ are related to the step size of the extremum seeking controller and both of them should be positive numbers to minimize the cost function. The standard extremum seeking controller with sinusoidal perturbations does not have the step size adapter and the outputs of the low pass filters directly go to the integrator, while the remaining structure of the algorithm is exactly the same. The step size adapter is detailed in Section 3.3.

## 3.2 Problem statement

In this chapter, we propose a method to find the most energy-efficient flight speed and sideslip angle to mitigate the common problem of the limited flight range and endurance of multicopters.

We choose to optimize these two variables because they affect the vehicle's power consumption and are typically additional (redundant) degrees of freedom in a multicopter's flight, where the flight missions require the vehicle to track specified geometric paths. Because the multicopter is usually not axisymmetric (especially when carrying payloads), flying with different sideslip angles affects the drag force faced by the vehicle and leads to different power consumption. The sideslip angle can be changed by changing the yaw angle. The flight speed also affects the power consumption of the vehicle: when the flight speed increases, the power consumption first decreases and then increases, which can be explained by momentum theory [29, Chapter 2.14]. This predicts that the maximum flight endurance is achieved by flying at a suitable flight speed, rather than hovering.

When our goal is to achieve the longest flight endurance (time), we want to minimize the consumed energy for a given time. As a result, the cost function for the optimal endurance flight is defined as the instantaneous electric power $p_e$. When the goal is to achieve the longest flight range (distance), we want to minimize the energy consumed for a given distance. Thus, the cost function for the optimal range flight is instantaneous electric power over speed $p_e/v$ (i.e. energy over distance), where $v$ denotes the speed of the vehicle.

A model-free optimization method is preferable, which can handle hard-to-model effects (e.g., components aging and temperature change) and payload changes. This motivates us to use an extremum seeking controller to find the optimal flight speed and sideslip angle. The required inputs to the extremum seeking controller are the instantaneous energy cost and a user-defined geometric path. Its outputs are the vehicle's reference speed and sideslip angle commands, which are then used to convert the geometric path into a reference trajectory to be tracked by the low-level controllers.

## 3.3 Model-free speed and sideslip adaptation

In this section, we introduce the novel multivariable extremum seeking controller with adaptive step size. It is able to achieve faster convergence than the standard extremum seeking controller with a fixed step size, by taking a smaller step size when the estimated gradient has a large magnitude or variance and vice versa. Vector variables and functions that map to vectors are written in boldface.

### Extremum seeking controller with adaptive step size

A block diagram of the proposed adaptive-step-size, multivariable, extremum seeking controller is shown in Fig. 3.2. We define the state variables of the multicopter (relevant to our problem) as $\boldsymbol{x} = [v, \beta]^T$, where $v$ and $\beta$ are the speed and sideslip of the vehicle, respectively. The outputs of the extremum seeking controller are defined as $\boldsymbol{r} = [r_v, r_\beta]^T$, where $r_v$ is the reference flight speed and $r_\beta$ is the reference flight sideslip. We assume a smooth control law $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{r})$, so that the closed-loop dynamics of the speed and sideslip are represented by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{r})). \tag{3.1}$$

The cost function is represented by

$$y = h(\boldsymbol{x}). \tag{3.2}$$

We make the following assumptions about the closed-loop vehicle dynamics and the cost function:

**Assumption 1.** There exists a smooth function $\boldsymbol{l} : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{r})) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{l}(\boldsymbol{r})$.

**Assumption 2.** For each reference input $\boldsymbol{r}$, the controller ensures that the equilibrium $\boldsymbol{x} = \boldsymbol{l}(\boldsymbol{r})$ is locally exponentially stable uniformly in $\boldsymbol{r}$.

**Assumption 3.** The cost function (described in Section 3.2) has a local minimum at $\boldsymbol{r}^* = [r_v^*, r_\beta^*]^T$, such that

$$\nabla(h \circ \boldsymbol{l})(\boldsymbol{r}^*) = 0, \quad \nabla^2(h \circ \boldsymbol{l})(\boldsymbol{r}^*) > 0. \tag{3.3}$$

### Gradient estimation

The extremum seeking controller approximates the gradient of the cost function and integrates the negative of the estimated gradient to minimize the cost [54]. To approximate the gradient of the cost function, sinusoidal perturbations

$$\boldsymbol{p}(t) = [a_v \sin(\omega_v t), a_\beta \sin(\omega_\beta t)]^T \tag{3.4}$$

are added to the speed setpoint $\hat{r}_v$ and sideslip setpoint $\hat{r}_\beta$, where $a_v$ and $a_\beta$ are the speed and sideslip perturbation magnitudes and the $\omega_v$ and $\omega_\beta$ are the speed and sideslip perturbation frequencies.

The cost function's value $y$ consists of low-frequency components ($\eta_v$ and $\eta_\beta$) and high-frequency components ($y - \eta_v$ and $y - \eta_\beta$). The cost is first high pass filtered to remove the low-frequency components and retain only the cost changes because of the perturbations. These values are then multiplied elementwise with the demodulation signals

$$\boldsymbol{d}(t) = [\sin(\omega_v t), \sin(\omega_\beta t)]^T, \tag{3.5}$$

where the demodulation signals' frequencies $w_v$ and $w_\beta$ are the same as their corresponding perturbation frequencies. We denote the results of the multiplications as $\xi_v$ and $\xi_\beta$. If the cost function's value change is in phase with the perturbations, which means that the cost value increases as the inputs' values increase, $\xi_v$ and $\xi_\beta$ will be positive. If they are out of phase, the outputs will be negative. After this, $\xi_v$ and $\xi_\beta$ are sent to low pass filters, whose outputs are approximations of the cost function's gradient, denoted by $q_v$ and $q_\beta$.

### Step size adapter

The difference between the proposed extremum seeking controller and the standard multivariable extremum seeking controller [55] is the step size adapters, which are defined as follows:

$$\dot{m}_v = \gamma_v(q_v^2 - m_v), \quad \dot{m}_\beta = \gamma_\beta(q_\beta^2 - m_\beta), \tag{3.6}$$

$$g_v = \frac{q_v}{\sqrt{m_v + \epsilon}}, \quad g_\beta = \frac{q_\beta}{\sqrt{m_\beta + \epsilon}}, \tag{3.7}$$

where $m_v, m_\beta$ are estimates of the second moments of the output of the low pass filters $q_v$ and $q_\beta$, and $\epsilon$ is a small positive constant preventing dividing by zero. Equations in (3.6) are essentially first-order low-pass filters for $q_v^2$ and $q_\beta^2$, and $\gamma_v$ and $\gamma_\beta$ denote their cut-off frequencies respectively. The idea is motivated by the adaptive moment estimation algorithm (Adam) [56], which is commonly used in the stochastic optimization of objective functions in machine learning, such as training neural networks [57, 58].

The adapters take in the output of the low pass filters $q_v$ and $q_\beta$ (the gradient estimates), and outputs $g_v$ and $g_\beta$. They are then passed to the integrators to perform gradient descent. The effective step size for gradient descent is $k_v g_v / q_v$ for the speed optimization and $k_\beta g_\beta / q_\beta$

for the sideslip optimization, and the step size adapters change them by changing $g_v$ and $g_\beta$. The second moments of the initial outputs from the low pass filters are used to initialize $m_v$ and $m_\beta$ in (3.6).

In (3.7), by dividing $q_v$ and $q_\beta$ with the square root of their corresponding second moments, the outputs $g_v$ and $g_\beta$ of the adapters will be approximately bounded by $\pm 1$, since $|\mathbb{E}[q_l]|/\sqrt{\mathbb{E}[q_l^2]} \leq 1$ ($\mathbb{E}$ denotes expected value, and $q_l$ being either $q_v$ or $q_\beta$). As a result, the descent rates for speed and slideslip are bounded by $k_v$ and $k_\beta$. This can be understood as establishing a trust region around the current parameter value, beyond which the current gradient estimation can be inaccurate. In addition, the adapters output small values when the gradient estimates have large uncertainty ($m_v$ and $m_\beta$ are large) and vice versa, which makes the controller more robust to noise.

## Stability Analysis

In this section, we present the stability proof of the novel multivariable extremum seeking controller with adaptive step size through averaging and singular perturbation analysis. A similar methodology was used in [55] to prove the stability of a single variable standard extremum seeking controller and was used in [59] to prove the stability of a multivariable Newton-based extremum seeking controller.

### System dynamics

By substituting the setpoint $\boldsymbol{r}$ with $\hat{\boldsymbol{r}} + \boldsymbol{p}(t)$, the closed-loop dynamics of the vehicle in (3.1) can be rewritten as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{r}} + \boldsymbol{p}(t))). \tag{3.8}$$

The proposed extremum seeking controller's dynamics in Fig. 3.2 can be summarized as

$$\begin{aligned}
&\dot{\hat{r}}_v = -k_v \frac{q_v}{\sqrt{m_v + \epsilon}}, \quad \dot{\hat{r}}_\beta = -k_\beta \frac{q_\beta}{\sqrt{m_\beta + \epsilon}}, \\
&\dot{q}_v = -\omega_{lv} q_v + \omega_{lv}(y - \eta_v) \sin w_v t, \\
&\dot{q}_\beta = -\omega_{l\beta} q_\beta + \omega_{l\beta}(y - \eta_\beta) \sin w_\beta t, \\
&\dot{\eta}_v = -\omega_{hv} \eta_v + \omega_{hv} y, \quad \dot{\eta}_\beta = -\omega_{h\beta} \eta_\beta + \omega_{h\beta} y, \\
&\dot{m}_v = \gamma_v(-m_v + q_v^2), \quad \dot{m}_\beta = \gamma_\beta(-m_\beta + q_\beta^2).
\end{aligned} \tag{3.9}$$

The parameters for the extremum seeking controller are selected as

$$\begin{aligned}
&\omega_v = \omega \omega_v^{'} = O(\omega), \quad \omega_\beta = \omega \omega_\beta^{'} = O(\omega), \\
&\omega_{hv} = \omega \delta w_{hv}^{'} = O(\omega\delta), \quad \omega_{h\beta} = \omega \delta w_{h\beta}^{'} = O(\omega\delta), \\
&\omega_{lv} = \omega \delta w_{lv}^{'} = O(\omega\delta), \quad \omega_{l\beta} = \omega \delta w_{l\beta}^{'} = O(\omega\delta), \\
&k_v = \omega \delta k_v^{'} = O(\omega\delta), \quad k_\beta = \omega \delta k_\beta^{'} = O(\omega\delta), \\
&\gamma_v = \omega \delta \gamma_v^{'} = O(\omega\delta), \quad \gamma_\beta = \omega \delta \gamma_\beta^{'} = O(\omega\delta),
\end{aligned} \tag{3.10}$$

where $\delta$ and $\omega$ are small positive constants, and $\omega_v^{'}$, $\omega_\beta^{'}$, $\omega_{hv}^{'}$, $\omega_{h\beta}^{'}$, $\omega_{lv}^{'}$, $\omega_{l\beta}^{'}$, $k_v^{'}$, $k_\beta^{'}$, $\gamma_v^{'}$ and $\gamma_\beta^{'}$ are positive constants. In addition, for this multivariable extremum seeking controller to work for both the speed and the sideslip angle simultaneously, their perturbation frequencies $\omega_v$ and $\omega_\beta$ should be distinct.

For the following averaging and singular pertubration analysis, we use the time scale $\tau = \omega t$. In addition, we define

$$\begin{aligned} \tilde{r}_v = \hat{r}_v - r_v^*, \quad \tilde{r}_\beta = \hat{r}_\beta - r_\beta^*, \\ \tilde{\eta}_v = \eta_v - (h \circ \boldsymbol{l})(\boldsymbol{r}^*), \quad \tilde{\eta}_\beta = \eta_\beta - (h \circ \boldsymbol{l})(\boldsymbol{r}^*). \end{aligned} \tag{3.11}$$

Then, the system dynamics in (3.8) and (3.9) with small perturbations can be rewritten as:

$$\omega \frac{d\boldsymbol{x}}{d\tau} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{r^*} + \tilde{\boldsymbol{r}} + \boldsymbol{p}(\tau))), \tag{3.12}$$

$$\frac{d}{d\tau}\begin{bmatrix} \tilde{r}_v \\ \tilde{r}_\beta \\ q_v \\ q_\beta \\ \tilde{\eta}_v \\ \tilde{\eta}_\beta \\ m_v \\ m_\beta \end{bmatrix} = \delta \begin{bmatrix} (-k_v^{'} q_v)/\sqrt{m_v + \epsilon} \\ (-k_\beta^{'} q_\beta)/\sqrt{m_\beta + \epsilon} \\ \omega_{lv}^{'}(y - (h \circ \boldsymbol{l})(\boldsymbol{r}^*) - \tilde{\eta}_v) \sin w_v^{'}\tau - \omega_{lv}^{'} q_v \\ \omega_{l\beta}^{'}(y - (h \circ \boldsymbol{l})(\boldsymbol{r}^*) - \tilde{\eta}_\beta) \sin w_\beta^{'}\tau - \omega_{l\beta}^{'} q_\beta \\ -\omega_{hv}^{'}\tilde{\eta}_v + \omega_{hv}^{'}(y - (h \circ \boldsymbol{l})(\boldsymbol{r}^*)) \\ -\omega_{h\beta}^{'}\tilde{\eta}_v + \omega_{h\beta}^{'}(y - (h \circ \boldsymbol{l})(\boldsymbol{r}^*)) \\ \gamma_v^{'}(-m_v + q_v^2) \\ \gamma_\beta^{'}(-m_\beta + q_\beta^2) \end{bmatrix} \tag{3.13}$$

where $\tilde{\boldsymbol{r}} = [\tilde{r}_v, \tilde{r}_\beta]^T$, $\bar{\boldsymbol{p}}(\tau) = \boldsymbol{p}(t/\omega)$.

## Averaging analysis

We first freeze the dynamics of the vehicle (3.8) at its equilibrium point $\boldsymbol{x} = \boldsymbol{l}(\boldsymbol{r}^* + \tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau))$, substitute it into (3.13) and get the reduced system

$$\frac{d}{d\tau}\begin{bmatrix} \tilde{r}_v \\ \tilde{r}_\beta \\ q_v \\ q_\beta \\ \tilde{\eta}_v \\ \tilde{\eta}_\beta \\ m_v \\ m_\beta \end{bmatrix} = \delta \begin{bmatrix} (-k_v^{'} q_v)/\sqrt{m_v + \epsilon} \\ (-k_\beta^{'} q_\beta)/\sqrt{m_\beta + \epsilon} \\ \omega_{lv}^{'}(v(\tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) - \tilde{\eta}_v) \sin w_v^{'}\tau - \omega_{lv}^{'} q_v \\ \omega_{l\beta}^{'}(v(\tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) - \tilde{\eta}_\beta) \sin w_\beta^{'}\tau - \omega_{l\beta}^{'} q_\beta \\ -\omega_{hv}^{'}\tilde{\eta}_v + \omega_{hv}^{'} v(\tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) \\ -\omega_{h\beta}^{'}\tilde{\eta}_v + \omega_{h\beta}^{'} v(\tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) \\ \gamma_v^{'}(-m_v + q_v^2) \\ \gamma_\beta^{'}(-m_\beta + q_\beta^2) \end{bmatrix}, \tag{3.14}$$

where $v(\tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) = (h \circ \boldsymbol{l})(\boldsymbol{r}^* + \tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau)) - (h \circ \boldsymbol{l})(\boldsymbol{r}^*)$. From Assumption 3 we have that:

$$v(0) = 0,\ \nabla v(0) = 0,\ \nabla^2 v(0) > 0. \tag{3.15}$$

To provide compact notations, we denote $\nabla^2 v(0) = H$ for later discussion. The least common period of sinusoidal functions with frequencies of $\omega_v^{'}$ and $\omega_\beta^{'}$ is defined as $\Pi$. We first prove the stability of the reduced system using averaging analysis:

**Proposition 1.** *For the reduced system* (3.14)*, under Assumption 3, there exists* $\bar{a}$ *and* $\bar{\delta}$ *such that for all* $\|\boldsymbol{a}\| \in (0, \bar{a})$*,* $\delta \in (0, \bar{\delta})$*, the reduced system dynamics* (3.14) *have a unique exponentially stable periodic solution of period* $\Pi$*, which for all* $\tau > 0$

$$\begin{aligned}
&\left|\tilde{r}_v^{\Pi}(\tau)\right| \leq O(\delta + \|\boldsymbol{a}\|^2),\ \left|\tilde{r}_\beta^{\Pi}(\tau)\right| \leq O(\delta + \|\boldsymbol{a}\|^2),\\
&\left|\tilde{\eta}_v^{\Pi}(\tau)\right| \leq O(\delta + \|\boldsymbol{a}\|^2),\ \left|\tilde{\eta}_\beta^{\Pi}(\tau)\right| \leq O(\delta + \|\boldsymbol{a}\|^3),\\
&\left|q_v^{\Pi}\right| \leq O(\delta),\ \left|q_\beta^{\Pi}\right| \leq O(\delta),\ \left|m_v^{\Pi}\right| \leq O(\delta),\ \left|m_\beta^{\Pi}\right| \leq O(\delta).
\end{aligned} \tag{3.16}$$

*Proof.* The reduced system (3.14) is in the form where the averaging method is applicable [60, Chapter 10.4] ($\delta$ is a small positive parameter). Its corresponding averaged system dynamics can be described as follows,

$$\frac{d}{d\tau}\begin{bmatrix} \tilde{r}_v^a \\ \tilde{r}_\beta^a \\ q_v^a \\ q_\beta^a \\ \tilde{\eta}_v^a \\ \tilde{\eta}_\beta^a \\ m_v^a \\ m_\beta^a \end{bmatrix} = \delta \begin{bmatrix} (-k_v^{'} q_v^a)/\sqrt{m_v^a + \epsilon} \\ (-k_\beta^{'} q_\beta^a)/\sqrt{m_\beta^a + \epsilon} \\ \omega_{lv}^{'} \frac{1}{\Pi}\int_0^{\Pi}(v(\tilde{\boldsymbol{r}}^{\boldsymbol{a}} + \bar{\boldsymbol{p}}(\sigma)) \sin\omega_v^{'}\sigma d\sigma - \omega_{lv}^{'} q_v^a \\ \omega_{l\beta}^{'} \frac{1}{\Pi}\int_0^{\Pi}(v(\tilde{\boldsymbol{r}}^{\boldsymbol{a}} + \bar{\boldsymbol{p}}(\sigma)) \sin\omega_\beta^{'}\sigma d\sigma - \omega_{l\beta}^{'} q_\beta^a \\ -\omega_{hv}^{'}\tilde{\eta}_v^a + \omega_{hv}^{'}\frac{1}{\Pi}\int_0^{\Pi} v(\tilde{\boldsymbol{r}}^{\boldsymbol{a}} + \bar{\boldsymbol{p}}(\sigma))d\sigma \\ -\omega_{h\beta}^{'}\tilde{\eta}_\beta^a + \omega_{h\beta}^{'}\frac{1}{\Pi}\int_0^{\Pi} v(\tilde{\boldsymbol{r}}^{\boldsymbol{a}} + \bar{\boldsymbol{p}}(\sigma))d\sigma \\ \gamma_v^{'}(-m_v^a + q_v^{a2}) \\ \gamma_\beta^{'}(-m_\beta^a + q_\beta^{a2}) \end{bmatrix}, \tag{3.17}$$

where the superscript $a$ denotes the variables of the averaged system, and $\Pi$ is the least common period of sinusoidal functions with frequencies of $\omega_v^{'}$ and $\omega_\beta^{'}$.

The equilibrium point of the averaged system (3.17) is denoted as $[\tilde{r}_v^{a,e}, \tilde{r}_\beta^{a,e}, q_v^{a,e}, q_\beta^{a,e}, \tilde{\eta}_v^{a,e},$

$\tilde{\eta}_\beta^{a,e}, m_v^{a,e}, m_\beta^{a,e}]^T$ which satisfies:

$$q_v^{a,e} = q_\beta^{a,e} = 0, \tag{3.18}$$

$$m_v^{a,e} = m_\beta^{a,e} = 0, \tag{3.19}$$

$$\int_0^\Pi (v(\tilde{\boldsymbol{r}}^{\boldsymbol{a,e}} + \bar{\boldsymbol{p}}(\sigma)) \sin \omega_v^{'} \sigma d\sigma = 0, \tag{3.20}$$

$$\int_0^\Pi (v(\tilde{\boldsymbol{r}}^{\boldsymbol{a,e}} + \bar{\boldsymbol{p}}(\sigma)) \sin \omega_\beta^{'} \sigma d\sigma = 0, \tag{3.21}$$

$$\tilde{\eta}_v^{a,e} = \tilde{\eta}_\beta^{a,e} = \frac{1}{\Pi} \int_0^\Pi v(\tilde{\boldsymbol{r}}^{\boldsymbol{a,e}} + \bar{\boldsymbol{p}}(\sigma)) d\sigma, \tag{3.22}$$

where the superscript $e$ denotes the variables for the equilibrium point. We consider $\tilde{r}_v^{a,e}$ and $\tilde{r}_\beta^{a,e}$ as perturbations with second-order Taylor series expansion over $a_v$ and $a_\beta$,

$$\tilde{r}_v^{a,e} = b_{1,v} a_v + b_{2,v} a_\beta + b_{3,v} a_v^2 + b_{4,v} a_v a_\beta + b_{5,v} a_\beta^2 + O(\|\boldsymbol{a}\|^3), \tag{3.23}$$

$$\tilde{r}_\beta^{a,e} = b_{1,\beta} a_v + b_{2,\beta} a_\beta + b_{3,\beta} a_v^2 + b_{4,\beta} a_v a_\beta + b_{5,\beta} a_\beta^2 + O(\|\boldsymbol{a}\|^3), \tag{3.24}$$

where $b_{i,v}$ and $b_{i,\beta}$ $(i = 1, .., 5)$ are constant numbers. By substituting (3.23), (3.24) into (3.20), (3.21), integrating and equating the like powers of $a_v$ and $a_\beta$, we can find that the first-order coefficients and second-order coefficients for the mixing terms are zero, and $\tilde{r}_v^{a,e}$ and $\tilde{r}_\beta^{a,e}$ can be written as:

$$\tilde{r}_v^{a,e} = b_{3,v} a_v^2 + b_{5,v} a_\beta^2 + O(\|\boldsymbol{a}\|^3), \tag{3.25}$$

$$\tilde{r}_\beta^{a,e} = b_{3,\beta} a_v^2 + b_{5,\beta} a_\beta^2 + O(\|\boldsymbol{a}\|^3). \tag{3.26}$$

In addition, by substituting (3.25), (3.26) into (3.22) and integrating, we can get

$$\tilde{\eta}_v^{a,e} = \tilde{\eta}_\beta^{a,e} = \frac{1}{4}(H_{11} a_v^2 + H_{22} a_\beta^2) + O(\|\boldsymbol{a}\|^3). \tag{3.27}$$

At the equilibrium point of the averaged system in (3.17), the Hessian $J_r^{a,e}$ is a block-diagonal matrix as follows,

$$J_r^{a,e} = \delta \begin{bmatrix} A & 0_{4\times4} \\ B & -\text{diag}(\omega_{hv}^{'}, \omega_{h\beta}^{'}, \gamma_v^{'}, \gamma_\beta^{'}) \end{bmatrix}, \tag{3.28}$$

where $A, B \in \mathbb{R}^{4\times4}$,

$$A = \begin{bmatrix} 0 & 0 & -k_v^{'}/\sqrt{\epsilon} & 0 \\ 0 & 0 & 0 & -k_\beta^{'}/\sqrt{\epsilon} \\ A_{31} & A_{32} & -\omega_{lv}^{'} & 0 \\ A_{41} & A_{42} & 0 & -\omega_{l\beta}^{'} \end{bmatrix}, \tag{3.29}$$

$$B = \begin{bmatrix} B_{11} & B_{12} & 0 & 0 \\ B_{21} & B_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.30}$$

with expressions of two matrices,

$$\begin{bmatrix} A_{31} & A_{32} \end{bmatrix}^T = \frac{\omega_{lv}^{'}}{\Pi} \int_0^{\Pi} \frac{\partial v(\tilde{\boldsymbol{r}}^{a,e} + \bar{\boldsymbol{p}}(\sigma))}{\partial \tilde{\boldsymbol{r}}^{a,e}} \sin \omega_v^{'} \sigma d\sigma, \tag{3.31}$$

$$\begin{bmatrix} A_{41} & A_{42} \end{bmatrix}^T = \frac{\omega_{l\beta}^{'}}{\Pi} \int_0^{\Pi} \frac{\partial v(\tilde{\boldsymbol{r}}^{a,e} + \bar{\boldsymbol{p}}(\sigma))}{\partial \tilde{\boldsymbol{r}}^{a,e}} \sin \omega_\beta^{'} \sigma d\sigma, \tag{3.32}$$

$$\begin{bmatrix} B_{11} & B_{12} \end{bmatrix}^T = \frac{\omega_{hv}^{'}}{\Pi} \int_0^{\Pi} \frac{\partial v(\tilde{\boldsymbol{r}}^{a,e} + \bar{\boldsymbol{p}}(\sigma))}{\partial \tilde{\boldsymbol{r}}^{a,e}} d\sigma, \tag{3.33}$$

$$\begin{bmatrix} B_{21} & B_{22} \end{bmatrix}^T = \frac{\omega_{h\beta}^{'}}{\Pi} \int_0^{\Pi} \frac{\partial v(\tilde{\boldsymbol{r}}^{a,e} + \bar{\boldsymbol{p}}(\sigma))}{\partial \tilde{\boldsymbol{r}}^{a,e}} d\sigma. \tag{3.34}$$

Hence, the block-lower-triangular matrix $J_r^{a,e}$ in (3.28) is Hurwitz if and only if that all diagonal submatrices are Hurwitz. Since $\delta, \gamma_v^{'}$, $\gamma_\beta^{'}$, $\omega_{hv}^{'}$ and $\omega_{h\beta}^{'}$ are positive constants, it remains to prove $A$ as Hurwitz for stability.

With a first-order Taylor expansion we can get that

$$\begin{bmatrix} A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \omega_{lv}^{'} a_v & 0 \\ 0 & \omega_{l\beta}^{'} a_\beta \end{bmatrix} H + O(\|\boldsymbol{a}\|). \tag{3.35}$$

The characteristic polynomial of $A$ with roots $\lambda$ can be written by computing the determinant of $\lambda I - A$,

$$\begin{aligned} &\det(\lambda I - A) \\ &= \det \left( \lambda I \left( \lambda I + \delta \begin{bmatrix} \omega_{lv}^{'} & 0 \\ 0 & \omega_{l\beta}^{'} \end{bmatrix} \right) + \frac{\delta^2}{\sqrt{\epsilon}} \begin{bmatrix} A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix} \begin{bmatrix} k_v^{'} & 0 \\ 0 & k_\beta^{'} \end{bmatrix} \right) \\ &= \det \left( \lambda^2 I + \lambda \delta \begin{bmatrix} \omega_{lv}^{'} & 0 \\ 0 & \omega_{l\beta}^{'} \end{bmatrix} + \frac{\delta^2}{2\sqrt{\epsilon}} \begin{bmatrix} \omega_{lv}^{'} a_v & 0 \\ 0 & \omega_{l\beta}^{'} a_\beta \end{bmatrix} H \begin{bmatrix} k_v^{'} & 0 \\ 0 & k_\beta^{'} \end{bmatrix} + O(\delta^2 \|\boldsymbol{a}\|) \right), \end{aligned} \tag{3.36}$$

which can be expanded to a 4th order polynomial of $\lambda$. Under the assumptions that $\|\boldsymbol{a}\|$ is small and that the Hessian $H$ in (3.15) is positive, the roots of this 4th order polynomial can be shown have negative real parts using the Routh-Hurwitz criterion [61, Chap. 6.2], implying that $A$ is Hurwitz. Therefore, $J_r^{a,e}$ is proven as Hurwitz. The Hurwitz Jacobian $J_r^{a,e}$ indicates that the equilibrium point of the averaged system (3.17) is locally exponentially stable if $a_v$ and $a_\beta$ are sufficiently small. Then according to [60, chapter 10.4], the theorem is proved. □

This implies that the error terms $\tilde{r}_v^{\Pi}(\tau)$ and $\tilde{r}_\beta^{\Pi}(\tau)$ converge to an $O(\delta + \|\boldsymbol{a}\|^2)$ neighbourhood of zero. The flight speed and sideslip found by the extremum seeking controller are periodic and converge to an $O(\delta + \|\boldsymbol{a}\|^2)$ neighbourhood of their optimal values $r_v^*$ and $r_\beta^*$ (i.e. values that minimize the cost functions defined in Section 3.2).

## Singular perturbation analysis

We then analyze the full system (3.12) and (3.13). To provide compact notations, we define the state vector of the extremum seeking controller as $\boldsymbol{z} = [\tilde{r}_v, \tilde{r}_\beta, q_v, q_\beta, \tilde{\eta}_v, \tilde{\eta}_\beta, m_v, m_\beta]^T$, and write (3.13) as

$$\frac{d\boldsymbol{z}}{d\tau} = \delta \boldsymbol{E}(\tau, \boldsymbol{x}, \boldsymbol{z}). \tag{3.37}$$

By Proposition 1, there exists an exponentially stable periodic solution $\boldsymbol{z}^\Pi(\tau)$ such that

$$\frac{d\boldsymbol{z}^\Pi(\tau)}{d\tau} = \delta \boldsymbol{E}(\tau, \boldsymbol{L}(\tau, \boldsymbol{z}^\Pi(\tau)), \boldsymbol{z}^\Pi(\tau)). \tag{3.38}$$

where $\boldsymbol{L}(\tau, \boldsymbol{z}^\Pi(\tau)) = \boldsymbol{l}(\boldsymbol{r}^* + \tilde{\boldsymbol{r}} + \bar{\boldsymbol{p}}(\tau))$. To convert the system (3.12) and (3.37) into the standard singular perturbation form, we shift the state $\boldsymbol{z}$ to get $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{z}^\Pi(\tau)$ such that

$$\omega \frac{d\boldsymbol{x}}{d\tau} = \tilde{\boldsymbol{F}}(\tau, \boldsymbol{x}, \tilde{\boldsymbol{z}}), \tag{3.39}$$

$$\frac{d\tilde{\boldsymbol{z}}}{d\tau} = \delta \tilde{\boldsymbol{E}}(\tau, \boldsymbol{x}, \tilde{\boldsymbol{z}}). \tag{3.40}$$

where

$$\begin{aligned} \tilde{\boldsymbol{E}}(\tau, \boldsymbol{x}, \tilde{\boldsymbol{z}}) &:= \boldsymbol{E}(\tau, \boldsymbol{x}, \tilde{\boldsymbol{z}} + \boldsymbol{z}^\Pi(\tau)) - \boldsymbol{E}(\tau, \boldsymbol{L}(\tau, \boldsymbol{z}^\Pi(\tau)), \boldsymbol{z}^\Pi(\tau)) \\ \tilde{\boldsymbol{F}}(\tau, \boldsymbol{x}, \tilde{\boldsymbol{z}}) &:= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{r}^* + \tilde{\boldsymbol{r}} + \boldsymbol{p}(\tau))). \end{aligned}$$

The quasi-steady state is

$$\boldsymbol{x} = \boldsymbol{L}(\tau, \tilde{\boldsymbol{z}} + \boldsymbol{z}^\Pi(\tau)). \tag{3.41}$$

By substituting the quasi-steady state into (3.40) and we get the reduced model

$$\frac{d\tilde{\boldsymbol{z}}}{d\tau} = \delta \tilde{\boldsymbol{E}}(\tau, \boldsymbol{L}(\tau, \tilde{\boldsymbol{z}} + \boldsymbol{z}^\Pi(\tau)), \tilde{\boldsymbol{z}}), \tag{3.42}$$

which has an equilibrium at the origin $\tilde{\boldsymbol{z}} = 0$. The equilibrium has been shown to be exponentially stable in the proof of Proposition 1. In addition, we study the stability of the boundary layer model (in the time scale $t = \tau/\omega$)

$$\frac{d\boldsymbol{x}_b}{dt} = \tilde{\boldsymbol{F}}(\tau, \boldsymbol{x}_b + \boldsymbol{L}(\tau, \tilde{\boldsymbol{z}} + \boldsymbol{z}^\Pi(\tau)), \tilde{\boldsymbol{z}}) = \boldsymbol{f}(\boldsymbol{x}_b + \boldsymbol{l}(\boldsymbol{r}), \boldsymbol{\alpha}(\boldsymbol{x}_b + \boldsymbol{l}(\boldsymbol{r}), \boldsymbol{r})). \tag{3.43}$$

Since $\boldsymbol{f}(\boldsymbol{l}(\boldsymbol{r}), \boldsymbol{\alpha}(\boldsymbol{l}(\boldsymbol{r}), \boldsymbol{r})) = 0$ according to Assumption 1, $\boldsymbol{x}_b = 0$ is the equilibrium of the boundary layer model (3.43). By Assumption 2, this equilibrium is locally exponentially stable uniformly in $\boldsymbol{r}$.

Combining the exponential stability of the reduced model with the exponential stability of the boundary layer model, and using Tikhonov's theorem on the infinite interval [60, Chapter 11.3], we can conclude that the solution of (3.37) is $O(\omega)$-close to the solution of the reduced model (3.42). Using the results of Proposition 1, we can then conclude that the error terms $\tilde{r}_v^{\Pi}(\tau)$ and $\tilde{r}_\beta^{\Pi}(\tau)$ converge to an $O(\omega + \delta + \|\boldsymbol{a}\|^2)$ neighbourhood of zero.

In summary, the proposed extremum seeking controller is locally stable – starting from an initial condition near the cost function's local minimum, it will converge to a neighbourhood around that local minimum if the perturbation is sufficiently small and slow relative to the closed-loop dynamics of the vehicle, and if the Assumptions 1-3 hold.

## Extremum seeking parameter selection

The values of parameters of the standard extremum seeking controller and our proposed adaptive step size extremum seeking controller used throughout the experiments are shown in Table 3.2. The perturbation frequencies ($w_v$ and $w_\beta$), perturbation magnitudes ($a_v$ and $a_\beta$), gains for the integrator ($k_v$ and $k_\beta$), cutoff frequencies of high-pass ($w_{hv}$ and $w_{h\beta}$) and low-pass filters ($w_{lv}$ and $w_{l\beta}$) need to be selected properly to achieve good performance of the extremum seeking controllers. The guidelines for choosing them are detailed below:

### Perturbation frequencies

The perturbation frequencies must be slow compared with the closed-loop dynamics of the quadcopter ($\omega$ should be small as mentioned in the stability analysis), such that they can be well tracked by the vehicle. Mathematically, the perturbation frequency could be selected smaller than the dominant frequency of the vehicle's closed-loop dynamics. The perturbation frequencies can be increased to achieve a faster convergence rate [62], given they can be tracked well by the vehicle. In addition, the multivariable extremum seeking control requires distinct perturbation frequencies for the speed and sideslip angle.

### Perturbation magnitudes and integrator gains

Large values for the perturbation magnitudes will be helpful for faster convergence, but will increase the oscillation magnitudes. Large values for the integrator gains will also be helpful for faster convergence, but will make the controller more sensitive to disturbances. As a result, we can increase the perturbation magnitudes and integrator gains to obtain the fastest convergence speed for a permissible amount of oscillation and sensitivity.

### Cutoff frequencies of the high-pass and low-pass filters

The cutoff frequencies of the high-pass and low-pass filters should be designed based on their corresponding perturbation frequencies: the cutoff frequency of the high-pass filter should be set higher than the perturbation frequency ($w_{hv} \geq w_v$ and $w_{h\beta} \geq w_v$), and the cutoff frequency of the low-pass filter should be set lower than the perturbation frequency ($w_{lv} \leq w_\beta$ and $w_{l\beta} \leq w_\beta$), to prevent attenuation of measurements at the perturbation frequency. We set the cutoff frequencies of the high-pass and low-pass filters to be the same as their corresponding perturbation frequencies, which simplified the parameter tuning process and was found to work well in the experiments.

#### Step-size adapter cutoff frequency

The two parameters in the step size adapters $\gamma_v$ and $\gamma_\beta$ are cutoff frequencies for the low-pass filters of the square for estimated gradient $q_v^2$ and $q_\beta^2$. One could increase their values as long as the noises are sufficiently attenuated.

In general, the selection of the extremum seeking parameters is a tuning process, but the guidelines above are valuable for making parameter tuning effectively.

## 3.4 Indoor experimental results

Indoor experiments were conducted to evaluate the performance of the proposed adaptive step size, multivariable extremum seeking method. We show that the proposed method is able to find the optimal sideslip and speed for different payloads and can converge more than 30% faster than the standard extremum seeking controller. The experiment video can be found at `https://www.youtube.com/watch?v=xuT-eGATesA`.

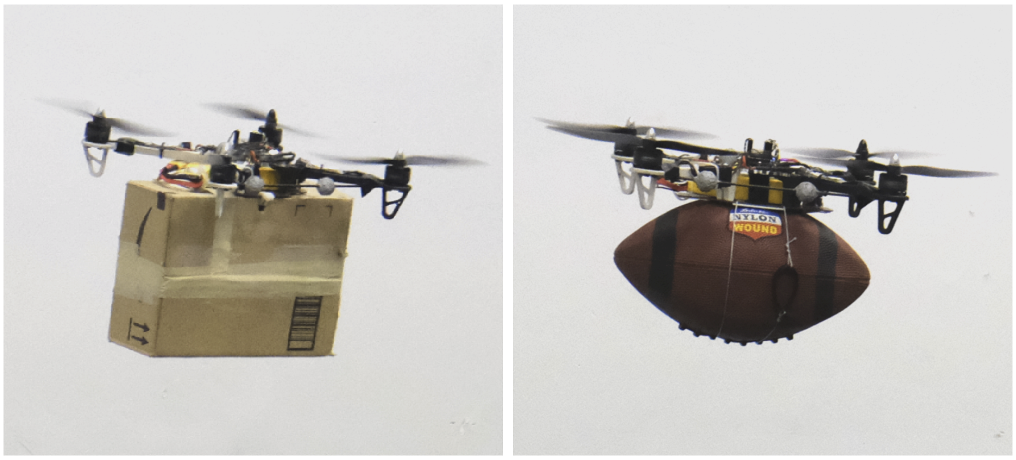


Figure 3.3: A quadcopter carrying a cardboard box payload and a football payload, used in the indoor experiments.

### Experimental setup

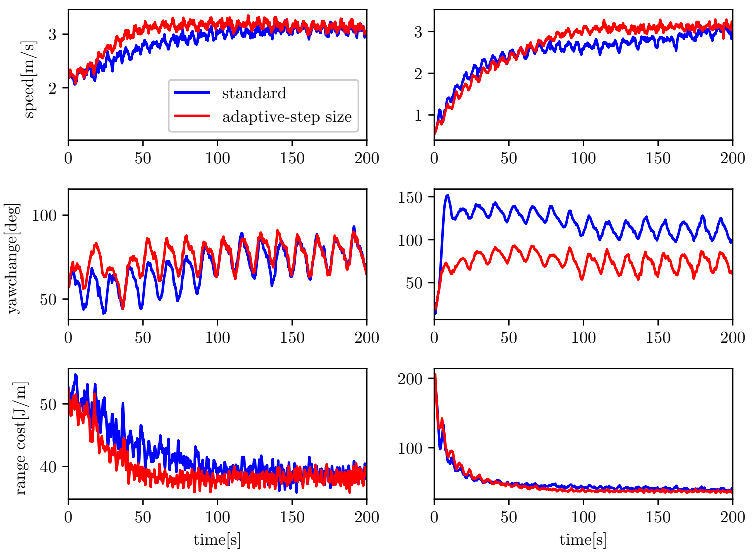
The quadcopter used in the indoor experiments are shown in Fig. 3.3. The vehicle weighs 660 grams without payload. The distance between the hubs of two diagonal motors is 330 mm and the propeller is 203 mm in diameter. We used two payloads during the experiment, one is a cardboard box that weighs 120 grams with a size of 255×180×85 mm; the other one is an American football that weighs 329 grams. A Crazyflie 2.0 [36] running a custom version of PX4 firmware was used as the low-level flight controller for the vehicle. The experiments were conducted in an indoor flight space with size of 7×6×5 m. A motion capture system was used for the state estimation of the vehicle and a voltage and current measurement module was connected to the battery to measure the instantaneous power consumption of the vehicle.

The value of parameters of the standard extremum seeking controller and the proposed adaptive step size extremum seeking controller used through out the experiments are shown in Table 3.2. The perturbation frequency of reference speed $w_v$ was set to 1 rad/s, and the perturbation frequency of reference sideslip $w_s$ was set to 0.5 rad/s. Multivariable extremum seeking control requires $w_v \neq w_s$ [54]. While increasing the perturbation frequencies is helpful to improve the convergence rate of the extremum seeking controller, one should make sure they are not too large for the vehicle to track. The cutoff frequencies of the high-pass and low-pass filters were set to be the same as their corresponding perturbation frequency. The magnitude of speed perturbation $a_v$ was set to be 0.15 m/s and the magnitude of sideslip was set to 7.5 degrees. These values need to be selected large enough to provide the extremum seeking controller with gradient information of the cost function and also make the neighborhood around the optimal value that the extremum seeking controller converges to small.

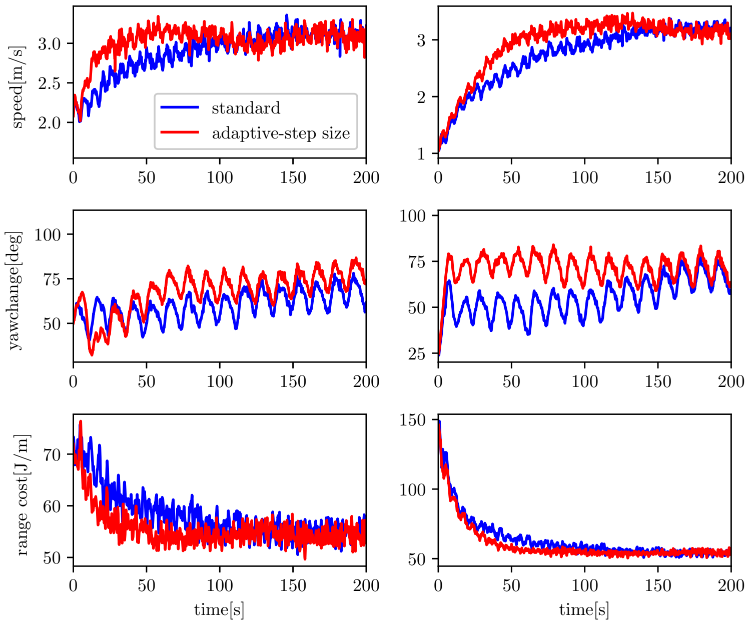
Table 3.1: Values of extremum seeking parameters for indoor experiments

| Parameter | Standard method | Proposed method |
|---|---|---|
| $a_v$ | 0.15 m/s | |
| $\omega_v, \omega_{hv}, \omega_{lv}$ | 1 rad/s | |
| $a_\beta$ | 7.5° | |
| $\omega_\beta, \omega_{h\beta}, \omega_{l\beta}$ | 0.5 rad/s | |
| $k_v$ | 0.025 | 0.1 |
| $k_\beta$ | 0.02 | 0.1 |
| $\gamma_v, \gamma_\beta$ | N/A | 0.5 rad/s |

To make a fair comparison, we kept all the control parameters for the two different methods to be the same except $k_v$ and $k_s$, since they have different meanings for the two methods: the $k_v$ and $k_s$ values are the step sizes for the standard method but are only part of the step sizes for the adaptive method, as shown in Section 3.3. They were empirically tuned in experiments for the two different methods to achieve the fastest convergence rate while guaranteeing the stability of the system (too large $k_v$ and $k_s$ will make the system unstable).



(a) Carrying a cardboard box payload.



(b) Carrying a football payload.

Figure 3.4: Comparison between the performance of the proposed method (in red) and the standard method (in blue) for searching the optimal range speed and sideslip of a quadcopter in the indoor experiments. The proposed method achieved a shorter convergence time.

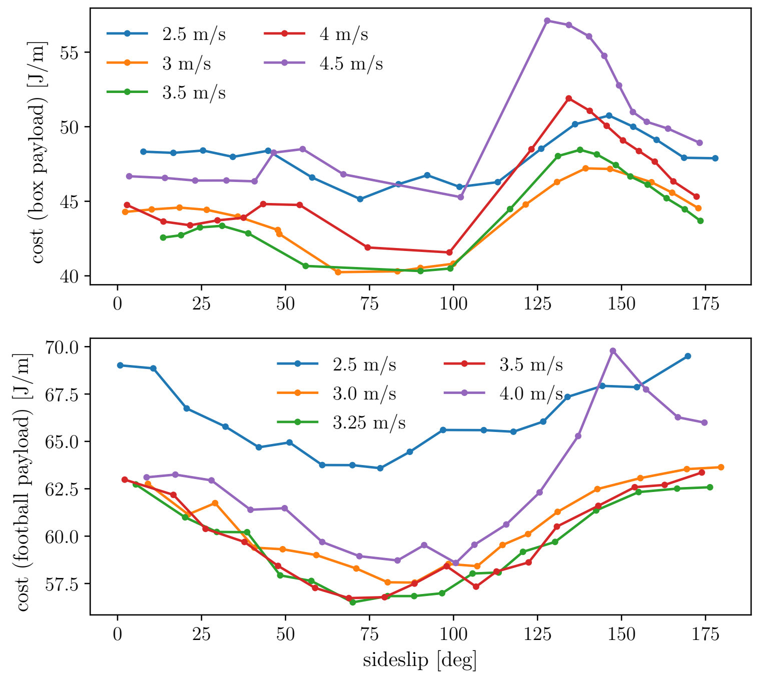


Figure 3.5: Ground truth values of the cost function with the cardboard box payload (the first row) and with the football payload (the second row) for the indoor experiments. Each data point in this figure is the average value of 15 seconds' experimental data with a frequency of 100 Hz. The cost function reaches the minimum value when the speed is 3.0 – 3.5 m/s and the sideslip is 65 – 100° for the cardboard box payload, and when the speed is about 3.25 m/s and the sideslip is 70° – 95° for the football payload.

## Optimal range speed and sideslip seeking

In the experiments, the quadcopter was commanded to fly along a circular path with constant height and a radius of 1.7 meters due to the space constraint. The cost function used is derived in (2.12), where the power measurement $p$ was provided by the onboard power module and the speed $v$ and was provided by the state estimator based on the motion capture system.

When the cardboard box was used as payload, the performance of the proposed and standard extremum seeking controller is compared in Fig. 3.4(a) in two tests with different initial conditions. In both tests, the proposed method converged in about 100 s, while the standard method took longer to converge: 150 s in the first test and 200 s in the second test. Both methods converged to speed and sideslip close to the optimal values shown in the first row of Fig. 3.5. When the football was used as payload, the performance of the two methods is compared in Fig. 3.4(b). In the first test, the proposed method converged in about 75 s and the standard method converged in about 125 s; in the second test, the proposed method converged in about 100 s and the standard method converge in about 150 s. Both methods converged to speed and sideslip close to the optimal values shown in the second row of Fig. 3.5.

The experiments have shown that the proposed method is able to find the optimal range speed and sideslip despite different payloads and initial conditions, and converges more than 33% faster compared to the standard method.

## 3.5 Outdoor experimental results

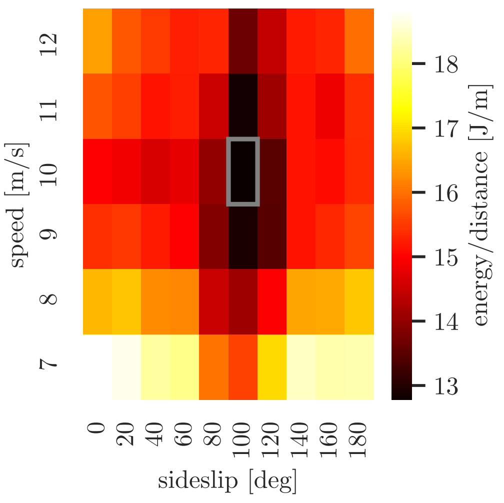
We then conducted outdoor experiments to further validate the effectiveness of the extremum seeking controller with adaptive step size to find the optimal flight speed and sideslip. The proposed method was shown to have better convergence speed than the standard extremum seeking control. It was also able to converge in the present of strong wind disturbances. The experiment video can be found at `https://youtu.be/aLds8LVfogk`.

The outdoor experiments extend the indoor experiments in the previous Section 3.4 in the following ways:

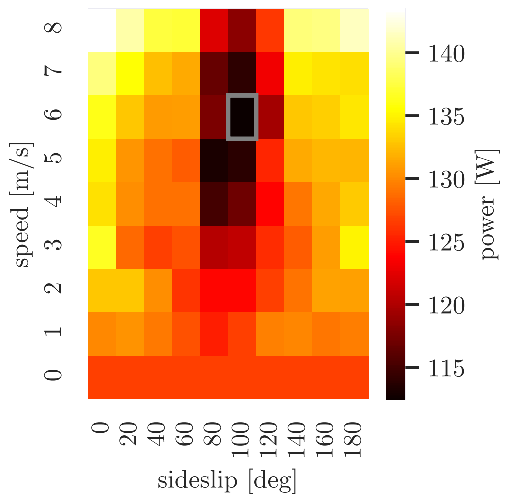
1. In the indoor experiments, a motion caption system was used to measure the vehicle's position and attitude at very high accuracy (about 1 mm error for position and 1 degree error for attitude) and at 200 Hz frequency. In contrast, in the outdoor experiments, a GPS was used for position estimation, whose accuracy was at meter level with a much lower frequency (10 Hz). As motion capture systems are not available in most of the real-world applications, this new sensor setup with GPS shows that our proposed method is able to perform well under much larger state estimation variances compared with indoor experiments.

2. Because of limited space, the multicopter was only able to fly a circular path of 1.7 m radius in previous indoor experiments. The centripetal force increased dramatically as the flight speed increased for such a small radius, contributing largely to the power consumption. Such an experimental setup is rare in real-world applications such as package delivery or surveillance, and made the vehicle's power consumption increase almost monotonically as the speed increased. In outdoor experiments, the centripetal force became much smaller due to much larger flight radius – a more realistic experiment setup. We were thus also able to find the speed and sideslip for optimal endurance flight, as the power as a function of speed and sideslip has a much deeper minimum.

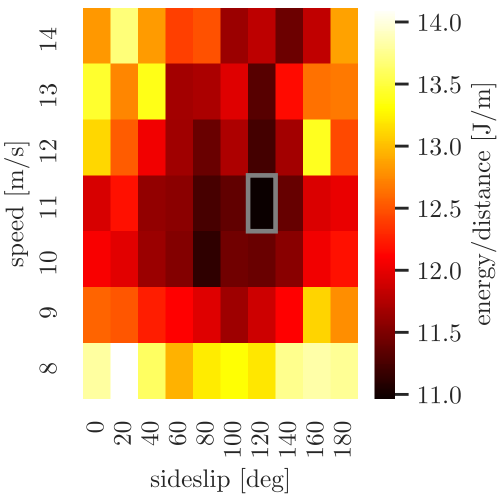
3. Experiments were conducted both on light wind and windy days, to see the effect of wind disturbances on the proposed method. Such real-world effects were not possible indoors.



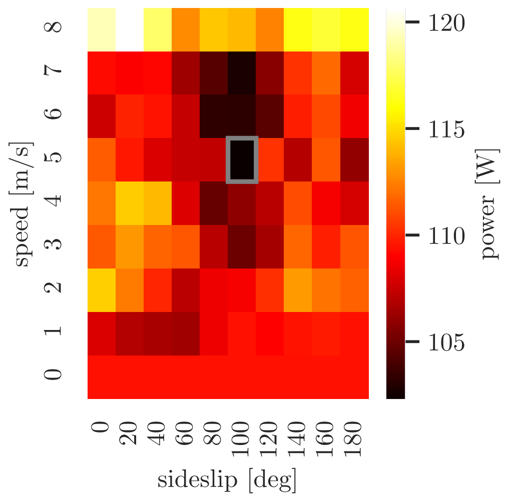
(a) Range cost with box payload.



(b) Endurance cost with box payload.



(c) Range cost with no box payload.



(d) Endurance cost with no box payload.

Figure 3.6: Ground truth data of the cost functions' values, with and without an additional box payload. Each square in the heat maps corresponds to 20 seconds' data collected at 50Hz. The optimal value in each case is encircled with a grey rectangle. (a) The range cost with the box payload reaches its minimum at about 10 m/s in speed and 100 degrees in sideslip. (b) The endurance cost with the box payload reaches its minimum at about 6 m/s and 100 degrees in sideslip. (c) The range cost with no box payload reaches its minimum at about 11 m/s and 120 degrees in sideslip. (d) The endurance cost with no box payload reaches its minimum at about 5 m/s and 100 degrees in sideslip.

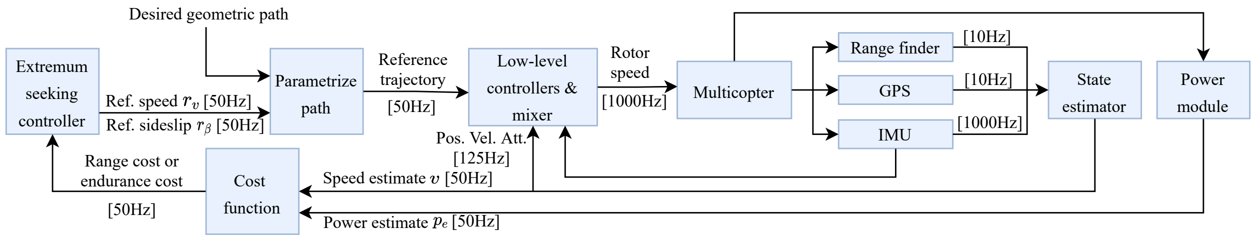


Figure 3.7: Control architecture for the model-free adaptive flight range or endurance optimization of a multicopter. The details of the extremum seeking controller block is shown in the dashed rectangle in Fig. 3.2. The extremum seeking controller runs on the onboard computer (Jetson Nano) while the low-level controllers and the state estimator run on the Pixracer flight controller.

## Experiment setup

The experiments were performed with a custom-built quadcopter with and without a plastic box payload (as shown in Fig. 3.1). The weight of the vehicle without the box payload was 0.9 kg, and the box weighs 0.1 kg and has a size of 180×115×80 mm. The distance between the hubs of the two diagonal motors is 330 mm and the propeller is 203 mm in diameter, same as the vehicle used in the indoor experiments. The extremum seeking controller was run on an onboard computer (Jetson Nano), and an mRo Pixracer R15 flight controller ran the standard PX4 firmware [63] including the state estimator and low-level controllers. The Jetson Nano and the Pixracer communicate through a UART link using mavros. The main reasons for running the extremum seeking controller on the onboard computer are for easier data logging and implementation. The computational power of micro controllers such as the Pixracer should also be able to run this algorithm, as it only requires several simple operations as shown in Fig. 3.2. Removing the onboard computer could further save the energy, at the cost of not being to log data as easily.

The experiments were conducted at a flat grass field at the Richmond Field Station, Richmond, CA (37.916588 N, -122.336667 E).

The control architecture for the vehicle is shown in Fig. 3.7. The extremum seeking controller (with or without adaptive step size) takes in the desired geometric path and instantaneous range cost or endurance cost. The power measurement $p_e$ comes from a power module (Holybro PM06 v2) connected to the battery, and the speed measurement $v$ comes from a state estimator based on a GPS (Zubax GNSS 2), a range finder for measuring the flight height (Beneware TFmini-S) and an IMU (Invensense MPU-9250). The extremum seeking controller outputs the reference tangential speed $r_v$ and sideslip $r_\beta$ along the desired path, which are used to parameterize the geometric path into a reference trajectory. The reference trajectory is then tracked by the low-level position and attitude controller, which is a cascaded PID controller.

The range of flight speed was 0-12 m/s when carrying the box payload and was 0-15 m/s without payload. The sideslip angle is a periodic variable, whose period is 180°, due to the vehicle and payload's rotational symmetry.

To make a fair comparison between the standard and the proposed extremum seeking controller, we kept all parameters for the two different methods to be the same except $k_v$ and $k_\beta$, since they have different meanings for the two methods: the $k_v$ and $k_\beta$ values are the step sizes for the standard method but are only part of the step sizes for the adaptive method, as shown in Section 3.3. They were empirically tuned in experiments for the two different methods to each achieve the fastest convergence rate in optimal range speed and sideslip searching when carrying a box payload 3.8(a).

Table 3.2: Values of extremum seeking parameters for outdoor experiments

| Parameter | Standard method | Proposed method |
|---|---|---|
| $a_v$ | 0.5 m/s | |
| $\omega_v, \omega_{hv}, \omega_{lv}$ | 1 rad/s | |
| $a_\beta$ | 10° | |
| $\omega_\beta, \omega_{h\beta}, \omega_{l\beta}$ | 0.5 rad/s | |
| $k_v$ | 0.05 | 0.11 |
| $k_\beta$ | 0.04 | 0.04 |
| $\gamma_v, \gamma_\beta$ | N/A | 0.5 rad/s |

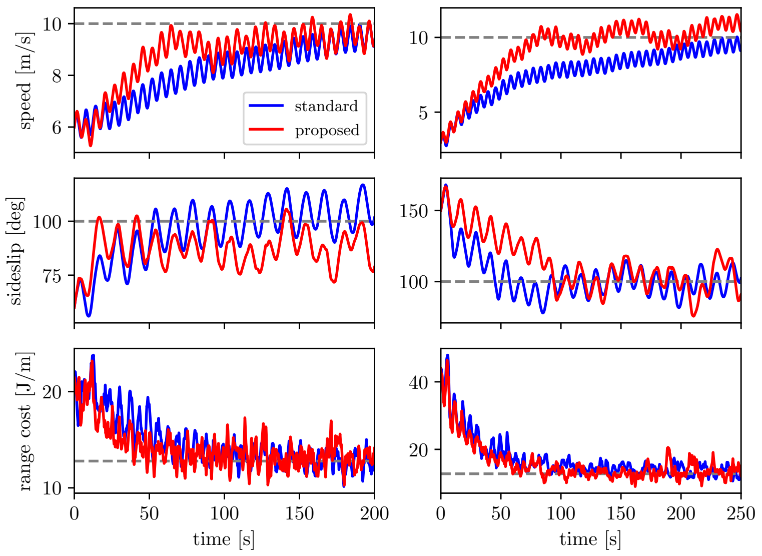
## Performance comparison under light wind

In the comparison experiments, the quadcopter was commanded to fly along a circular path with 30 meters in radius and a constant height of 5 meters. The experiments were conducted during good weather to minimize the effect of wind disturbances.

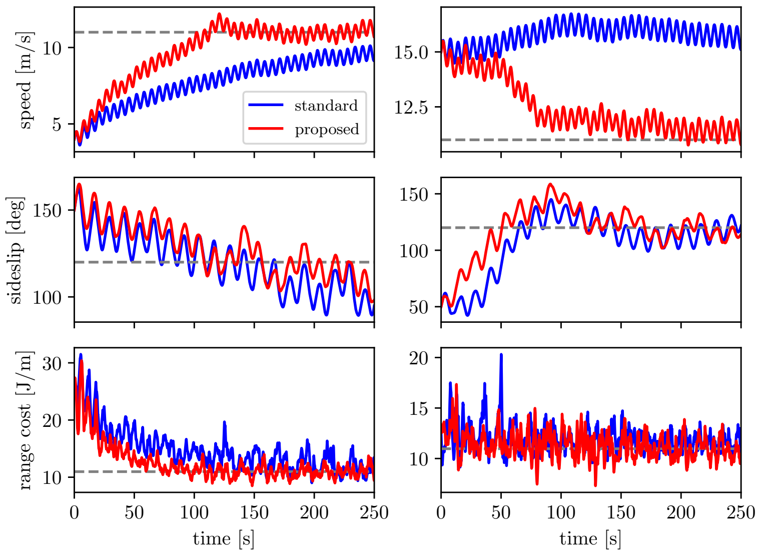
### Cost value ground truth

To verify that the proposed extremum seeking controller is able to converge close to the optimal speed and sideslip, we experimentally evaluated the optimal range and optimal endurance cost functions. When the vehicle is carrying the box payload, the values of the cost functions at various speed and sideslip are shown at Fig. 3.6(a) and Fig. 3.6(b), while Fig. 3.6(c) and Fig. 3.6(d) show the values without box.

The data shows the importance of flying at the energy efficient speed and sideslip: compared to flying at the maximum achievable speed in the experiments with a uniformly selected random sideslip, flying at the optimal range speed and sideslip on average increases the flight

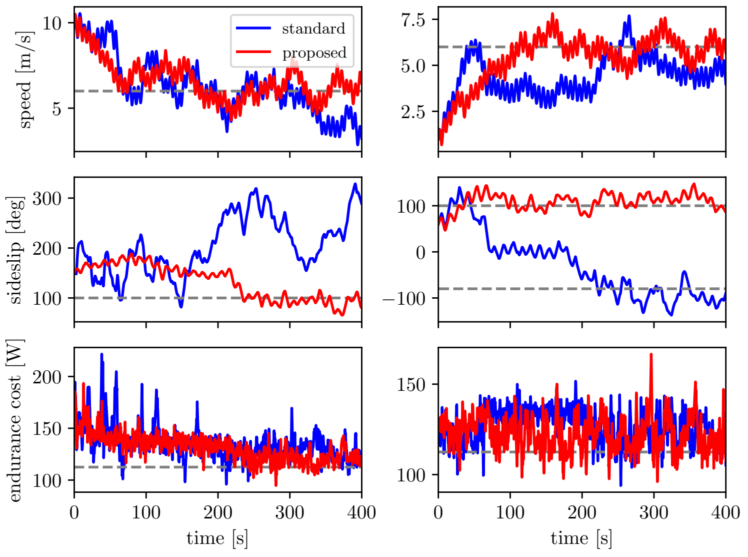


(a) Carrying an additional box payload.

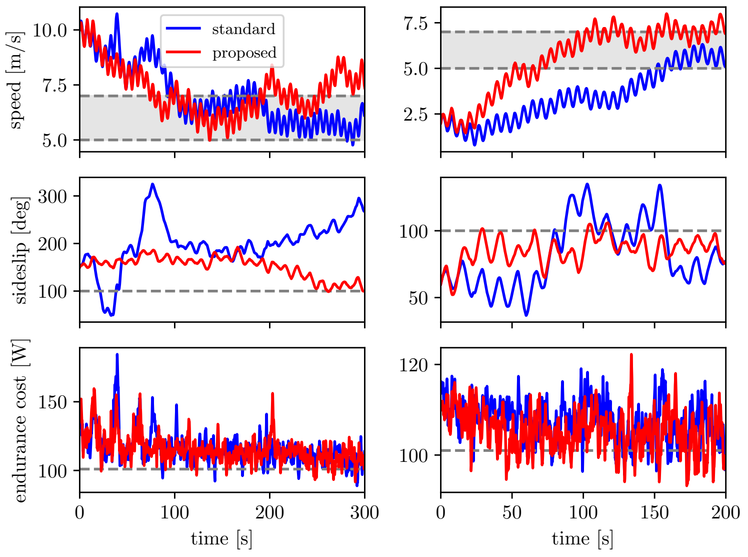


(b) Without carrying an additional box payload.

Figure 3.8: Optimal range speed and sideslip searching performance comparison between the proposed method (red lines) and the standard method (blue lines). The ground truth values for optimal speed and sideslip are marked as grey dashed lines (values from Fig. 3.6). The results when carrying an additional box payload are shown in (a) and the results with no additional box payload are shown in (b). Each column in the subfigures represent a test with a different initial speed and sideslip.



(a) Carrying an additional box payload.



(b) Without carrying an additional box payload.

Figure 3.9: Optimal endurance speed and sideslip searching performance comparison between the proposed method (red lines) and the standard method (blue lines). The ground truth values for optimal speed and sideslip are marked as grey dashed lines and grey shaded regions (values from Fig. 3.6). The results when carrying an additional box payload are shown in (a) and the results with no additional box payload are shown in (b). Each column in the subfigures represent a test with a different initial speed and sideslip.

range by 14.3% without payload and 19.4% with a box payload. Besides, compared to hovering, flying at the optimal endurance speed and sideslip increases the flight time by 7.5% without payload and 14.4% with a box payload.

### Convergence speed comparison and discussion

The convergence speed of the standard and the proposed methods are compared in experiments with/without a box payload and with different initial conditions. We consider the extremum seeking controller converges when both the speed and sideslip settle close to their optimal value. When the goal is to find the speed and sideslip which achieve the optimal flight range, the results are compared in Fig. 3.8(a) and Fig. 3.8(b). When the goal is to find the speed and sideslip which achieve the optimal flight endurance, the results are compared in Fig. 3.9(a) and Fig. 3.9(b). In the second test of Fig. 3.9(a), the optimal sideslip is marked at both 100 degrees and -80 degrees. This is because the vehicle and payload are rotational symmetric, such that a sideslip offset of 180 degrees has the same effect on the vehicle's power consumption. In Fig. 3.9(b), the optimal speed is marked as a range between 5 - 7 m/s, because the cost function values are very close in this range with less than 1% difference.

The convergence times are summarized in Table 3.3 for optimal range and in Table 3.4 for optimal endurance (N/A represents that the method failed to converge by the end of the experiment). We can see that the proposed method converged about twice as fast as the standard method in these tests.

Table 3.3: Optimal range speed and sideslip seeking

| Payload | Initial speed | Initial sideslip | Standard | Proposed |
|---|---|---|---|---|
| box | 6 m/s | 60 deg | 200 s | 50 s |
| | 3 m/s | 150 deg | 250 s | 100 s |
| none | 4 m/s | 150 deg | 250 s | 125 s |
| | 15 m/s | 50 deg | N/A | 100 s |

Table 3.4: Optimal endurance speed and sideslip seeking

| Payload | Initial speed | Initial sideslip | Standard | Proposed |
|---|---|---|---|---|
| box | 10 m/s | 150 deg | N/A | 250 s |
| | 1 m/s | 60 deg | 200 s | 100 s |
| none | 10 m/s | 150 deg | N/A | 250 s |
| | 2 m/s | 60 deg | 175 s | 75 s |

In summary, we can see that the proposed extremum seeking controller with step-size adapter converged about twice as fast as the standard extremum seeking controller. In addition, the parameters of the extremum seeking controller were tuned for optimal range speed and sideslip searching when carrying a box payload, as mentioned in Section 3.3. The same set of parameters still worked well for the other experiment setups (optimal endurance goal, with and without box payload) for the proposed method, showing that the method has good robustness to parameters. However, the standard extremum seeking method failed to converge in some cases, suggesting it is less robust.

Like other perturbation-based extremum seeking methods, the convergence speed of the proposed method is still limited by the time-scale separation, which requires the changing of the speed and sideslip setpoints to be slow compared to the perturbation frequencies. In our experimental tests, the proposed extremum seeking controller converged within 2 minutes in the majority of cases. We think this would be a practically useful convergence time considering the flight time of most multicopters are between 10 to 20 minutes [16].

## Cost of extremum seeking

Since the perturbations are applied by the extremum seeking controller, the power consumption of the vehicle will be higher than the flight at a constant reference without perturbations. In this subsection, we compare the optimal values of the cost function without perturbation (i.e., optimal cost values in Fig. 3.6) with the average cost values when flying at the same mean speed and sideslip but with perturbations applied. The increases in cost are summarized in Table 3.5.

Table 3.5: Optimal cost increase due to perturbation

| Optimization goal | Payload | Cost without perturbation | Cost with perturbation | Cost increase |
|---|---|---|---|---|
| range | box | 12.8 J/m | 13.2 J/m | 3.1 % |
| | none | 11.0 J/m | 11.4 J/m | 4.0 % |
| endurance | box | 112.5 W | 116.4 W | 3.5 % |
| | none | 101 W | 105.2 W | 4.2 % |

In summary, the increase in cost was 3.1 - 4.2 % because of the perturbations applied by the extremum seeking controller. This is less than the power consumption reduction when flying at the optimal endurance speed compared to hovering, which is 12.6% with the box payload and 7% without it, so the advantage of the proposed method outweighs its cost.

To reduce the impact of this increase, the extremum seeking controller can be enabled only when there is a model change (e.g., picking up a new payload), and disabled after convergence. In addition, decreasing the perturbation magnitude will be helpful for reducing the additional cost of perturbation, but this will also reduce the convergence speed. One should take these two factors into account when selecting the proper perturbation magnitude.

### Performance under strong wind disturbances

We further evaluated the performance of the proposed extremum seeking controller with adaptive step size under strong wind disturbances. Like the aforementioned experiments, the vehicle was commanded to follow a circular path with a radius of 30 meters at 5 meters in height. The wind was measured by a Young 81000 anemometer at 20 Hz with 0.01 m/s resolution, at a height of 2 meters. The extremum seeking controller's parameters are the same as the experiments under light wind in Section 3.5.

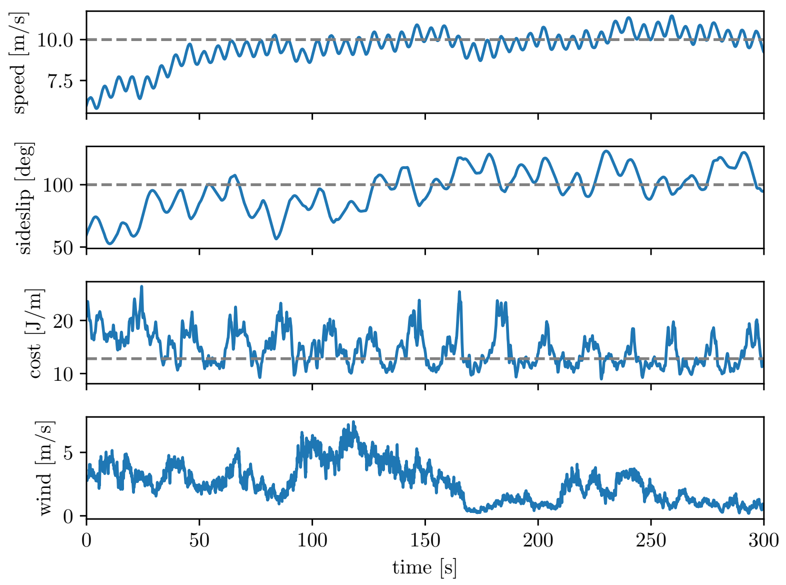


Figure 3.10: Optimal range speed and sideslip seeking under strong wind disturbances, with the box payload. The optimal values of the speed, sideslip and cost function are marked as grey dashed lines. The maximum magnitude of wind disturbances is 7.43 m/s.

The experiments demonstrated that the proposed method was still able to find the optimal range and endurance speed and sideslip, as shown in Fig. 3.10 and Fig. 3.11. The maximum wind magnitude was 7.43 m/s in the optimal range experiment, and was 4.83 m/s in the optimal endurance experiment. The proposed method is not very sensitive to wind disturbances: because of the time-scale separation in the extremum seeking controller, the change in the speed and sideslip setpoints by the extremum seeking controller is very slow compared with the closed-loop dynamics of the vehicle.

Compared with the tests with the same initial conditions but under light wind in Section 3.5, the wind disturbances caused larger oscillations in the reference sideslip (Fig. 3.10 compared with the first column of Fig. 3.8(a)) and longer convergence time (Fig. 3.11 compared with the second column of Fig. 3.9(b)).

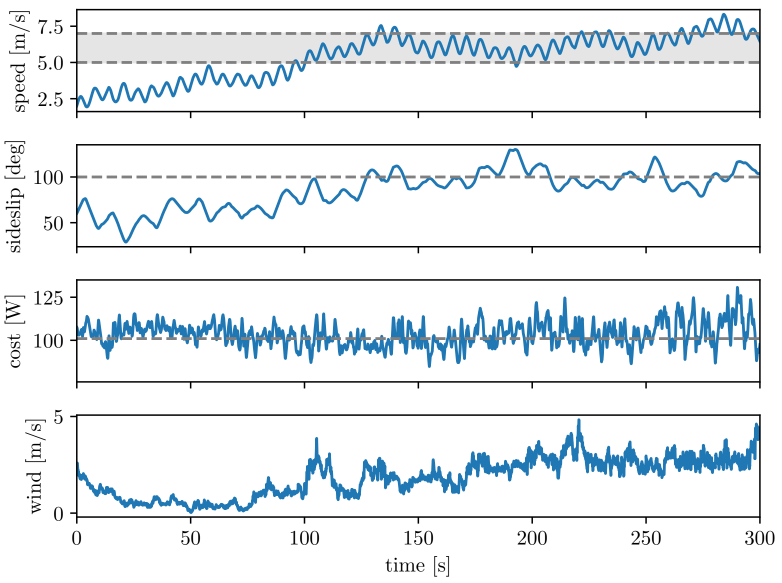


Figure 3.11: Optimal endurance speed and sideslip seeking under strong wind disturbances, without the box payload. The optimal values of the sideslip and cost function are marked as grey dashed lines. The optimal value of the speed is marked as a range between 5 - 7 m/s, because the cost function values are very close in this range, with less than 1% difference. The maximum magnitude of wind disturbances is 4.83 m/s

## 3.6 Conclusion

An online, adaptive, model-free method for finding the speed and sideslip that maximize the flight range or endurance of multicopters is proposed in this chapter. Not dependent on any power consumption model of the vehicle, it is able to adapt to different payloads and is easy to deploy. The proposed method can mitigate the common problem of limited flight range and endurance of multicopters. Based on a novel multivariable extremum seeking controller with adaptive step size, it is able to achieve faster convergence compared to the standard extremum seeking controller with fixed step size.

Through realistic outdoor experiments, we show that this method is able to find the optimal speed and sideslip correctly under different payloads and under strong wind disturbances. In addition to multicopters, this method can also be applied to fixed wing aerial robots to find the optimal flight speed (to achieve the longest flight time or distance) whose sideslip is usually not a free degree of freedom in path planning.

## Acknowledgements

The work in this chapter was partially supported by the J.K. Zee Fellowship, the UC Berkeley Graduate Division Block Grant Award, and the AFRI Competitive Grant no. 2020-67021-32855/project accession no. 1024262 from the USDA National Institute of Food and Agriculture. The AFRI Competitive Grant is being administered through AIFS: the AI Institute for Next Generation Food Systems. `https://aifs.ucdavis.edu`.

Research was also partially sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-20-2-0105. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

This work was also partially supported by the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge. The experimental testbed at the HiPeRLab is the result of the contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

# Chapter 4

# Autonomous flight through cluttered outdoor environments

In this chapter, we introduce a collision avoidance system for navigating a multicopter in cluttered outdoor environments based on the recent memory-less motion planner, rectangular pyramid partitioning using integrated depth sensors (RAPPIDS). The RAPPIDS motion planner generates collision-free flight trajectories at high speed with low computational cost using only the latest depth image. In this chapter, we extend it to improve the performance of the planner by taking the following issues into account. (a) Changes in the dynamic characteristics of the multicopter that occur during flight, such as changes in motor input/output characteristics due to the drop in battery voltage. (b) Noise from the inertial measurement unit (IMU), which can cause unwanted control input components. (c) Planner utility function which may not be suitable for the cluttered environment. Therefore, in this chapter we introduce solutions to each of the above problems and propose a system for the successful operation of the RAPPIDS planner in an outdoor cluttered flight environment. At the end of the chapter, we validate the proposed method's effectiveness by presenting the flight experiment results in a forest environment. Note that the material presented in this chapter is based on the following previously published work.

- Junseok Lee[1], Xiangyu Wu[1], Seung Jae Lee and Mark W. Mueller. "Autonomous flight through cluttered outdoor environments using a memoryless planner". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 1131–1138

## 4.1 Introduction

Motion planning algorithms for multicopter unmanned aerial vehicles to fly autonomously to their destination in cluttered environments are in general grouped into two categories. One

[1]Junseok Lee and Xiangyu Wu contributed equally to this article. Names are in alphabetical order.

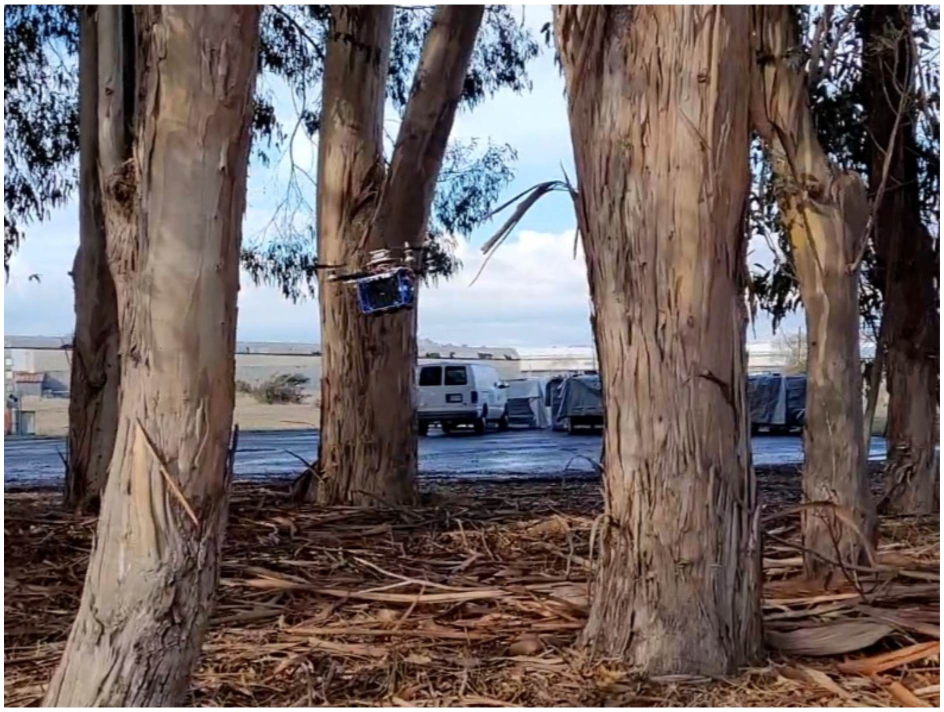


Figure 4.1: The vehicle flies autonomously using visual-inertial odometry and the collision-avoidance planner through a forest, avoiding trees.

is to separately run a path planning algorithm to generate collision-free path as a purely geometrical problem without considering dynamics [64], and use a path follower to follow the collision-free path. Since it does not consider dynamics constraints, collision avoidance can not be guaranteed at high speed since dynamics constraints, such as motor thrust limit, are not considered at the time of planning. The other approach considers dynamics constraints and directly generates control commands by including obstacle avoidance as a constraint inside an optimization problem rather than separating path planning and tracking, for example, based on the rapidly-exploring random tree star (RRT*) [65], the nonlinear optimization [66], and the mixed-integer programming (MIP) [67].

We focus on the latter category which considers dynamics in planning as well as collision avoidance. The collision-free trajectories are often further constrained by minimizing a cost function depending on applications, for example, the minimum time, the minimum energy, and the shortest distance. Trajectory generation algorithms may be divided into two major types: map-based algorithms and memory-less algorithms [68].

Map-based algorithms are global planning methods of creating collision-free optimal trajectories after a single large map is constructed or while creating a map by fusing all spatial sensor information obtained during flights. For instance, in [69], a local map of the environment is used and a nonconvex, nonlinear optimization problem is solved to get collision-free smooth trajectories. In [70, 71, 72] the free-space in the map is represented as multiple convex regions, and an optimization problem is then solved to find a series of trajectories through the free-space. In [73], the convex hull property of B-spline trajectories is used to solve for safe and fast trajectories, and the success rate and optimality is improved in the subsequent work of [74]. Map-based algorithms have the advantage of optimal trajectory generation because it presupposes that map information is already known when planning. However, they usually have high computational cost and due to fusing sensor data into the map.

On the other hand, memory-less algorithms use only the most recent sensor information to avoid obstacles, such as using the k-d-tree [75, 76], and a trajectory library precomputed offline to reduce a significant amount of online computation [77]. Therefore, memory-less algorithms are classified as local planning methods and are advantageous for obstacle avoidance in a dynamic obstacle environment due to low computational cost and high update frequency. However, there is a disadvantage that it is challenging to find a globally optimal trajectory, since global spatial information is not available at the time of planning.

For trajectory generation of a small-size multicopter that requires high-speed maneuver but has a limited payload capacity, memory-less algorithms have great utility for the following reasons. First, memory-less algorithms are easy to be implemented in real-time on miniature on-board computers with limited computational resources. Second, due to the fast trajectory update speed thanks to the low computational cost, memory-less algorithms can cope with rapidly changing surroundings during high-speed flight. Lastly, the algorithm is less prone to accumulated odometry errors during flights in a cluttered environment because it utilizes only the latest sensor information for the planning.

Recently, a memory-less planner is proposed using rectangular pyramid partitioning using integrated depth sensors (RAPPIDS) motion planner, which has high computational efficiency for high-speed collision avoidance flights [78]. Using the RAPPIDS motion planner, the authors were able to achieve high collision avoidance flight performance in cluttered indoor flight environment, using only stand-alone depth images and visual-inertial odometry information processed by an on-board computer mounted on the vehicle.

In this chapter, we extend the RAPPIDS planner to operate in a cluttered outdoor off-road environment. We describe the system's development process and flight results. In the experiment, the system was able to fly 30 meters in a forest environment, as shown in Fig. 4.1, with a maximum speed of 2.7 m/s.

## 4.2 RAPPIDS motion planning framework

In this section, we repeat some details from [78], and add a velocity-limiting check for stable visual-inertial odometry. Motion primitives are sampled and then go through a series of checks to find a trajectory that is minimum-cost, input-feasible, velocity-admissible, and collision-free, as shown in Algorithm 1. Since the planner has a low computational cost,

**Algorithm 1** Trajectory Constraint Checks

```
Require: A sampled candidate trajectory
 1: procedure ConstraintsCheck
 2:     if lower cost than known then
 3:         if input feasibility then
 4:             if velocity admissibility then
 5:                 if collision-free then
 6:                     trajectory_status ← collision_free
 7:                 else
 8:                     trajectory_status ← in_collision
 9:                 end if
10:             else
11:                 trajectory_status ← velocity_inadmissible
12:             end if
13:         else
14:             trajectory_status ← input_infeasible
15:         end if
16:     else
17:         trajectory_status ← higher_cost
18:     end if
19: end procedure
```

we plan in a receding horizon fashion every time a new depth image arrives. This keeps a collision-free trajectory being updated with the latest depth view, and allows avoiding obstacles that are not included in the previous camera view.

## Candidate trajectory sampling

We sample candidate trajectories first by sampling an endpoint and constructing a polynomial trajectory connecting the current position to the endpoint. Specifically, we first uniformly sample a 2D point in the pixel coordinates of a depth camera. We also draw a sampled depth from a uniform distribution, and then back-project the 2D point using the sampled depth to obtain a sampled endpoint $\mathbf{s}_T \in \mathbb{R}^3$.

Denote $\mathbf{s}(t)$, $\dot{\mathbf{s}}(t)$, and $\ddot{\mathbf{s}}(t) \in \mathbb{R}^3$ to be the position, velocity and acceleration of the vehicle in the inertial frame. The candidate motion primitives are described as below.

$$\mathbf{s}(t) = \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{\ddot{\mathbf{s}}(0)}{2}t^2 + \dot{\mathbf{s}}(0)t + \mathbf{s}(0),\ t \in [0\ T], \tag{4.1}$$

where $T$ is the trajectory duration, $\mathbf{s}(0)$, $\dot{\mathbf{s}}(0)$, and $\ddot{\mathbf{s}}(0)$ are the initial position, velocity, and acceleration of the vehicle at the time when the trajectory starts, and $\alpha$, $\beta$ and $\gamma$ are coefficients such that $\mathbf{s}(T) = \mathbf{s}_T$, and $\dot{\mathbf{s}}(T) = \ddot{\mathbf{s}}(T) = 0$. This terminal condition is selected to be at rest to guarantee safety in every depth frame. Specifically speaking, even if the planner fails to find a new collision-free trajectory in the following frames, the vehicle can keep tracking the current trajectory, which is collision-free in a static environment. This 5th order polynomial corresponds to the minimum-jerk trajectory, which minimizes the average Euclidean norm of jerk over the trajectory duration $T$. The trajectory is smooth and can be checked for collisions efficiently, as shown in [79].

## Velocity constraints for stable visual-inertial odometry

We impose velocity constraints to prevent the visual-inertial odometry from losing track in high-speed flights. The sampled trajectories are filtered out if their maximum velocity exceeds a predefined threshold, $v_{\max}$. For the sake of computational tractability, the constraints are checked for each per-axis velocity using analytical solution for third-order polynomials. It should be noted that checking the magnitude requires solving higher-order polynomials numerically, because analytical solutions do not exist. The following equation can be derived by taking the derivative of (4.1),

$$\dot{\mathbf{s}}(t) = \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + \ddot{\mathbf{s}}(0)t + \dot{\mathbf{s}}(0) \tag{4.2}$$

To find its extrema, we compute its derivative and find the zeros as below.

$$\ddot{\mathbf{s}}(t) = \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + \ddot{\mathbf{s}}(0) = 0 \tag{4.3}$$

The third-order polynomial can be efficiently solved, and the magnitude of (4.2) is evaluated at the roots as well as the boundary, 0 and $T$. The procedure is repeated for every axis, and we discard the motion primitive candidate if the speed on any axis exceeds the per-axis velocity limit $v_{\max}$.

## Collision check: Pyramid method

The RAPPIDS planner determines whether the candidate trajectory intrudes the obstacle by pyramid inflation. Fig. 4.2 shows the process in which the depth camera on the flying vehicle searches for area $\mathcal{P}$ that guarantees a non-collision path. First, we define free space $\mathcal{F}$ and occupied space $\mathcal{O}$ based on the depth camera image. We also treat all spaces outside the field of view that are $l$ distance from the vehicle as occupied spaces to avoid collisions with unrecognized obstacles outside the camera's field of view. Next, we select the final position $\mathbf{s}(T)$ through random sampling and then search for the nearest depth pixel $p$ from $\mathbf{s}(T)$. Then, starting at pixel $p$ and reading the surrounding depth pixels in a spiral sequence, we get the largest possible rectangular space $\mathcal{P}_{exp}$ that does not intrude the occupied space $\mathcal{O}$. Finally, pyramid $\mathcal{P}$ distanced with vehicle radius $r$ is created by shrinking the expanded pyramid $\mathcal{P}_{exp}$. By checking whether the $\mathbf{s}(t)$ candidate trajectory remaining inside $\mathcal{P}$, we can conclude that the trajectory is collision-free from the detected obstacles.

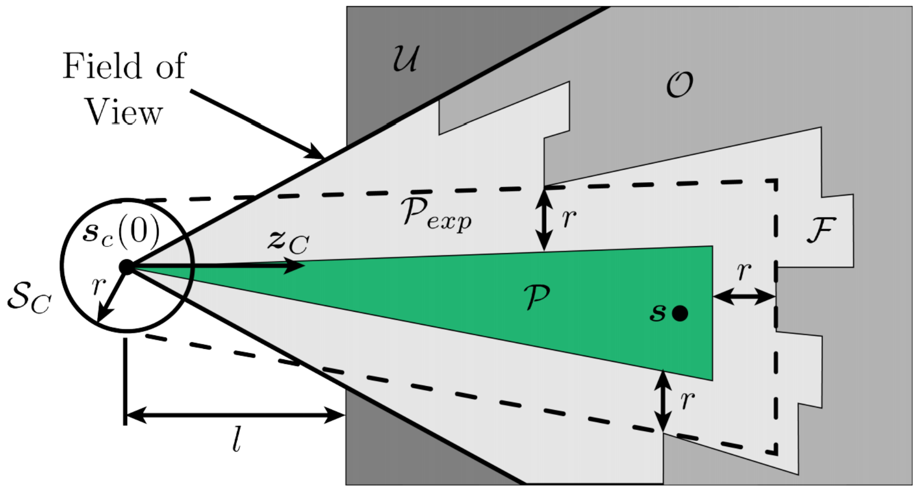


Figure 4.2: The planner uses a collection of pyramids to represent the free space, which allows simple and fast collision check of a sampled trajectory. *The figure is sourced from [78].*

The algorithm can guarantee zero collision trajectory, because the trajectory generated by the algorithm inherently avoids not only the obstacles recognized by the depth sensor but also the obstacles located in an unobserved ($\mathcal{U}$) or unknown area ($\mathcal{O}$) obscured by the detected obstacles. A detailed description of this algorithm can be found at [78].

## 4.3 Algorithm for fast outdoor flight

In this section, we describe modifications to the motion planner other than the velocity check described in section 4.2, for collision-free flight outdoors to autonomously reach a target waypoint. (a) We sample trajectories with final positions around the center of the view of the depth camera to improve the efficiency of sampling trajectories. (b)We proposed a new utility function that behaves similarly to the utility function of maximizing the average velocity, but also considers making the vehicle stay around the target. (c) The vehicle is always yawed towards the goal to dynamically change the view during the flight that can potentially increase the chance of finding a collision-free trajectory compare to the view with a fixed yaw. (d)The initial acceleration used for sampling trajectories is approximated by the total thrust command divided by the mass instead of using noisy IMU measurements, since the noise in acceleration hampers the planner from finding proper trajectories. (e)We compensate the thrust change because of battery voltage drop during the flight.

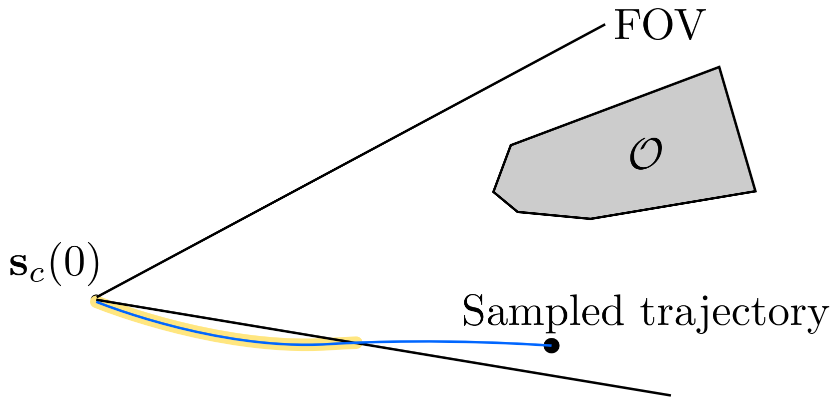


Figure 4.3: Efficient trajectory sampling in the field of view (FOV) of the depth camera. An occluded space $\mathcal{O}$ at the center makes the planner construct pyramids around the FOV. A sampled trajectory (blue) with an endpoint close to the FOV is likely to have a part that resides outside the field of view (highlighted by yellow), which is classified by the planner as in-collision as space outside of the FOV is considered as occupied. To increase the sampling efficiency, we sample points only between 10% and 90% of horizontal and vertical FOV.

## Sampling efficiency

As collision checking at the last step of the planner is computationally expensive, it is beneficial to increase the probability of finding a collision-free trajectory from candidate trajectories. We improve the sampling efficiency by excluding candidate trajectories with high chances of being classified as in collision. One of the most common cases is when a sampled candidate trajectory has an endpoint around the field of view (FOV), as shown in Fig. 4.3. If the endpoint is close to the edges of the FOV, it is likely that some parts of the sampled trajectory fall outside of the FOV, and it is classified by the planner as in-collision because regions outside of the FOV are considered occupied by the planner, as described in section 4.2. To improve the trajectory sampling efficiency by avoiding those cases, the final trajectory position is sampled between 10% and 90% of the field-of-view.

## Utility function

We propose a utility function, which is equivalent to a negative cost function, as below to generate trajectories that not only consider maximizing the average velocity to the goal, but also keep the trajectories' end points around the target.

$$U(P,\ t) = \frac{\|\mathbf{d}_{s_tG}\| - \|\mathbf{d}_{PG}\|}{t}, \tag{4.4}$$

where $\|\mathbf{d}_{s_tG}\|$, $\|\mathbf{d}_{PG}\|$, and $t$ are the distances between the current position and the goal, the distance between the endpoint of the motion primitives and the goal, and the primitive execution time, respectively. As shown in Fig. 4.4, when the vehicle is far from the waypoint, the term $\|\mathbf{d}_{PG}\|$ does not vary much, and hence the utility function is to maximize the average velocity to the waypoint. Around the waypoint, however, the term $\|\mathbf{d}_{PG}\|$ plays a role to encourage the planner to choose a trajectory whose final position falls around the goal.

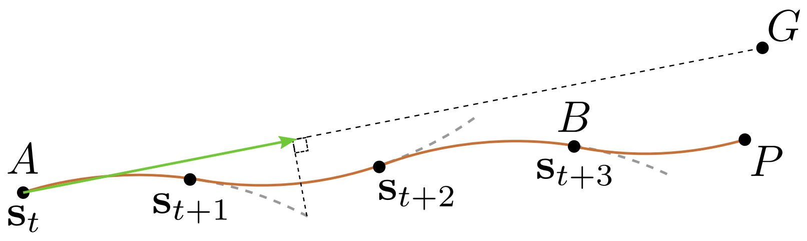


Figure 4.4: A new cost function to maximize average velocity and handle the behavior around the goal point $G$ is proposed. Every time a new depth image arrives, the planner attempts to find a new collision-free trajectory using the current estimates $s_t$, and if found, the tracking controller discards the previous trajectory (gray-dashed parts), and starts tracking the new trajectory (brown). When far from the goal point, for example point $A$, the utility function is roughly the average velocity measured in direction to the goal (depicted by the green arrow), which still allows lateral motion. However, around the goal, such as point $B$, the utility function encourages the planner to generate a new collision-free trajectory with the endpoint around the goal, not beyond the goal.

## Desired yaw angle

The yaw angle can be arbitrarily chosen while tracking a collision-free trajectory. We yaw the vehicle always towards the goal, because it is more likely to find a trajectory to the goal when facing towards it.

$$\begin{aligned}\psi_c &= \operatorname{arctan2}\left([0\ 1\ 0]\left(\mathbf{s}_G - \mathbf{s}\right),\ [1\ 0\ 0]\left(\mathbf{s}_G - \mathbf{s}\right)\right) \\ &\in \left[-\pi, \pi\right],\end{aligned} \tag{4.5}$$

where arctan2$(y, x)$ measures the signed angle between the point $(x, y)$ and the positive x-axis, and $\psi_c$, $\mathbf{s}_G$ and $\mathbf{s}$ are the commanded yaw angle, the positions of the goal and the vehicle in the inertial frame, respectively. It should be noted that the yaw control can be replaced with a more intelligent control from a high-level planner.

## Acceleration estimation

Our planner checks collision of a trajectory that is sampled given the current position, velocity, and acceleration (section 4.2). Using acceleration measurements from IMU for the current acceleration results in inaccurate sampled trajectories due to the noise in IMU acceleration measurements. We instead estimate acceleration using the last commanded thrust as below

$$\ddot{\mathbf{s}}\,(0) = \frac{c}{m}\mathbf{z}_B - \mathbf{g}, \tag{4.6}$$

where $c$, $\mathbf{z}_B$, $m$, and $\mathbf{g}$ are the last collective thrust command, body z-axis, mass, and gravitational acceleration.

## Thrust adaptation

Once the collision-free trajectory is selected, the cascaded flight controller generates motor speed commands to maneuver the vehicle, as shown in Fig. 4.9. The commands are then sent to the electric speed controllers (ESCs), which control each motor's rotation speed based on the predefined protocol in their firmware.

Commonly used ESCs, including those used in this project, run in 'open-loop' mode, meaning that the rotor angular velocity produced is not fixed for a fixed command, but varies with e.g. battery voltage. The relationship between the thrust command, battery voltage, and thrust force produced is shown in Fig. 4.5 for the experimental vehicle. This section describes the mitigation strategy used in this chapter to achieve reliable thrust commands with such open-loop ESCs, with specifically an offline calibration for the major variations, followed by an online adaptive loop to compensate for remaining model error.

### Offline calibration

At a constant battery voltage, a quadratic fit matches the data shown in Fig. 4.5 well:

$$\bar{f}_i(u_i) = k_v(V_{\text{batt}})(c_0(u_i + c_1)^2 + c_2), \tag{4.7}$$

with $\bar{f}_i$ the thrust produced by motor $i$ at command $u_i$, $k_v(\cdot)$ a term dependent on the battery voltage $V_{\text{batt}}$, and $c_{\{0,1,2\}}$ voltage-independent fixed parameters. The function $k_v(\cdot)$, and parameters $c_i$ are common to all motors, and are estimated using a least-squares approach.

The voltage-dependent function $k_v(\cdot)$ is estimated from a long-term hover, where the vehicle maintained a constant position as the battery voltage varied from fully charged to almost depleted. The result is shown in Fig 4.6, and a simple first-order fit for the voltage dependence is used:

$$k_v(V_{\text{batt}}) = k_{v_1} + k_{v_2}V_{\text{batt}} \tag{4.8}$$

with $V_{\text{batt}}$ the battery voltage, and $k_{v_{\{1,2\}}}$ parameters estimated via least squares.

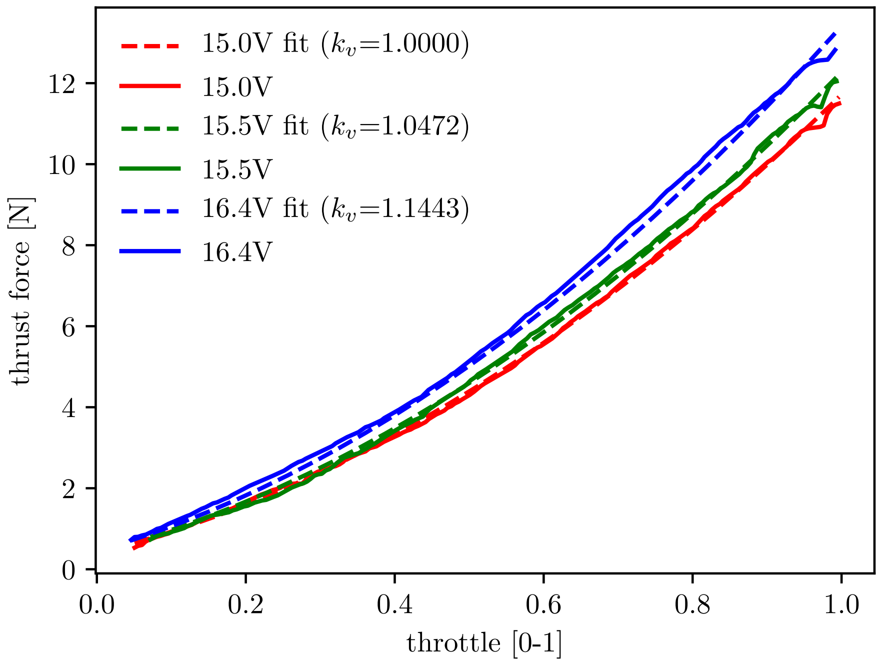


Figure 4.5: Thrust force at different thrust commands (solid lines) and second-order fitting results (dotted lines). The command-thrust relationship changes depending on the voltage applied to the propulsion system.

### Online adaptation

To compensate for remaining errors, we compare the thrust estimated from the accelerometer data to the desired thrust. We model the actual thrust produced by the propellers, $f_i$ in relation to the offline model $\bar{f}_i$ with the adaptation coefficient $k_a$, common to all propellers:

$$f_i = k_a \bar{f}_i \tag{4.9}$$

The estimate of the actual thrust is computed as follows, noting that the translational dynamics of a multicopter are given by

$$m\ddot{\mathbf{s}} = \mathbf{R}\mathbf{e_z}\Sigma f_i + m\mathbf{g}, \tag{4.10}$$

where $\mathbf{R}$ is the vehicle's attitude expressed as a rotation matrix, and $\mathbf{e_z} = [0\ 0\ 1]^T$. Note that this assumes that there are no other forces acting on the vehicle, such as aerodynamic disturbances, drag forces, etc. Noting that the vehicle's accelerometer measures the vehicle's proper acceleration, i.e. acceleration relative to free fall, we have that the output $\alpha$ of an

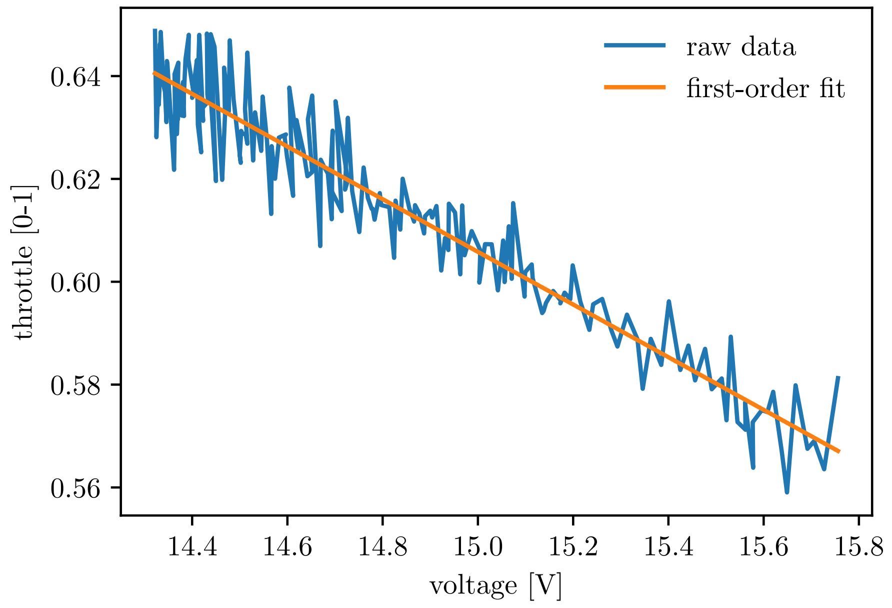


Figure 4.6: Change in throttle command required at hover as the vehicle's battery is depleted from fully charged (approx. 15.8V) to nearly depleted (approx. 14.4V).

ideal, noise-free accelerometer located at the vehicle centre of mass given by

$$\mathbf{R}\alpha + \mathbf{g} = \ddot{\mathbf{s}}, \tag{4.11}$$

Comparing (4.10) and (4.11), we can bring the following result

$$m\alpha_z = \Sigma f_i, \tag{4.12}$$

where $\alpha_z$ is the component of the accelerometer measurement parallel to the thrust vector. I.e. we can estimate the actual thrust force generated from the vehicle with z-directional IMU acceleration measurements. The coefficient $k_a$ is then computed as follows:

$$k_a = \mathrm{LPF}\left(\frac{m\alpha_z}{\Sigma\bar{f}_i}\right) \tag{4.13}$$

with LPF$(\cdot)$ a suitable low pass filter, necessary to average out the noisy data.

Given a desired thrust value (e.g. from the controller), an ESC command is generated by computing the corresponding calibrated thrust $\bar{f}_i$ from (4.9), and then solving the quadratic in (4.7) to compute the desired low-level ESC command $u_i$.

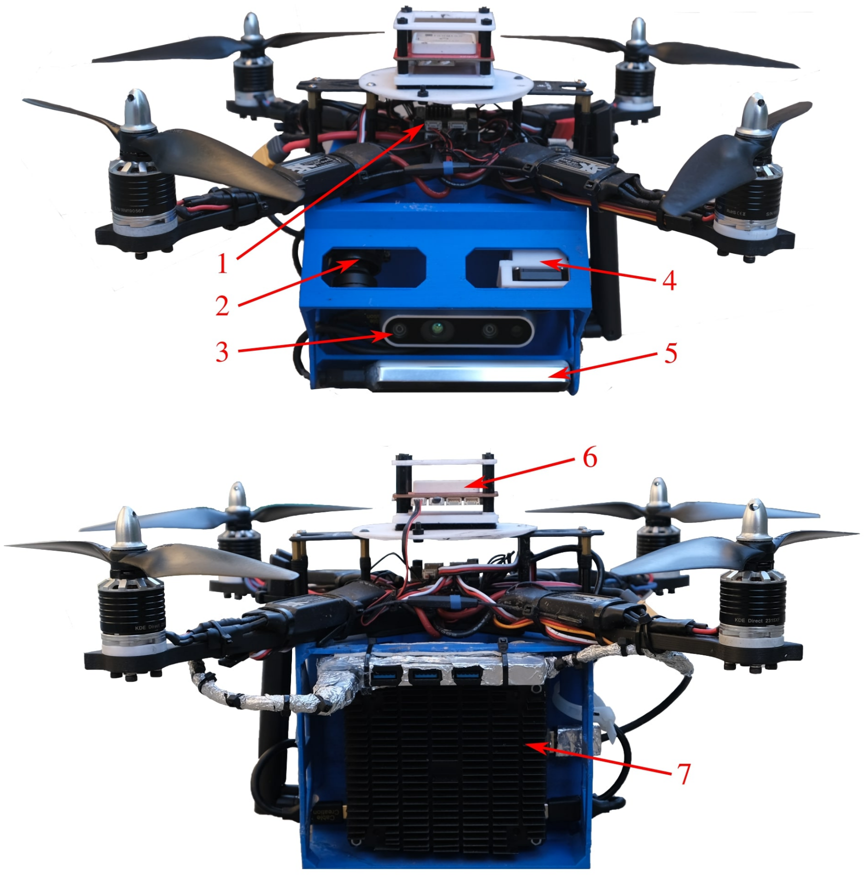


Figure 4.7: The custom built quadcopter. 1 - Pixracer flight controller; 2 - RGB camera (not used in feedback); 3 - D435i depth camera; 4 - infra-red camera (not used in feedback); 5 - T265 tracking camera; 6 - GPS (not used in feedback); 7 - Jetson AGX Xavier

## 4.4 Experimental results

We have shown in outdoor experiments that the system is able to navigate in a complex forest experiment with a maximum speed of 2.7 m/s. The experiment was repeated several times and we got similar performance. In this section we give detailed explanation of one of these experiments. A video can be found at `www.youtube.com/watch?v=3av5xEuKg2w&ab_channel=HiPeRLab`

### System setup

A custom-built quadcopter, as shown in Fig. 4.7, was used through the experiments. It weighs 2.4 kg and the distance between two diagonal motors is 382 mm. The diameter of

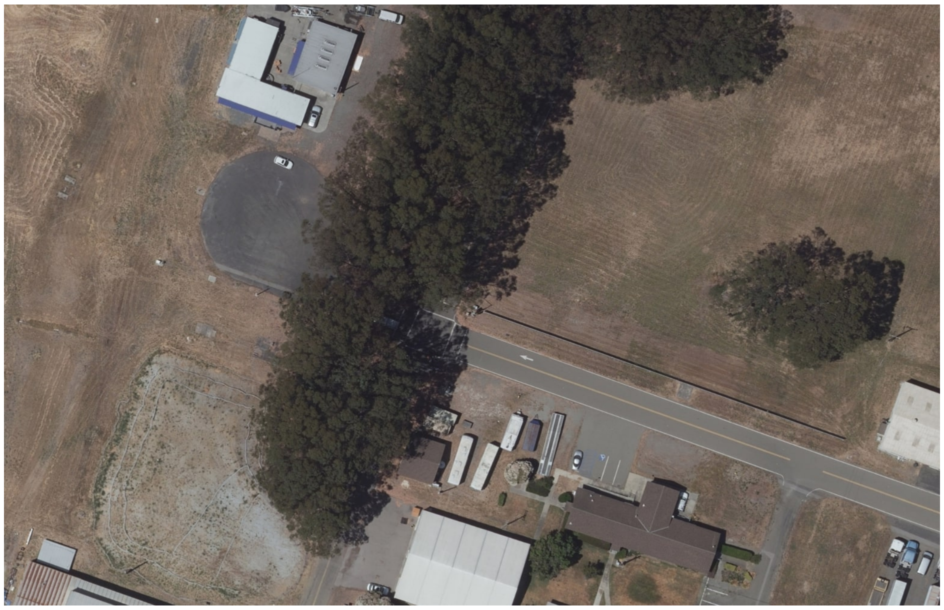


Figure 4.8: Satellite image of the small forest at the Richmond Field Station where the experiment was conducted. Image is from `www.usgs.gov`.

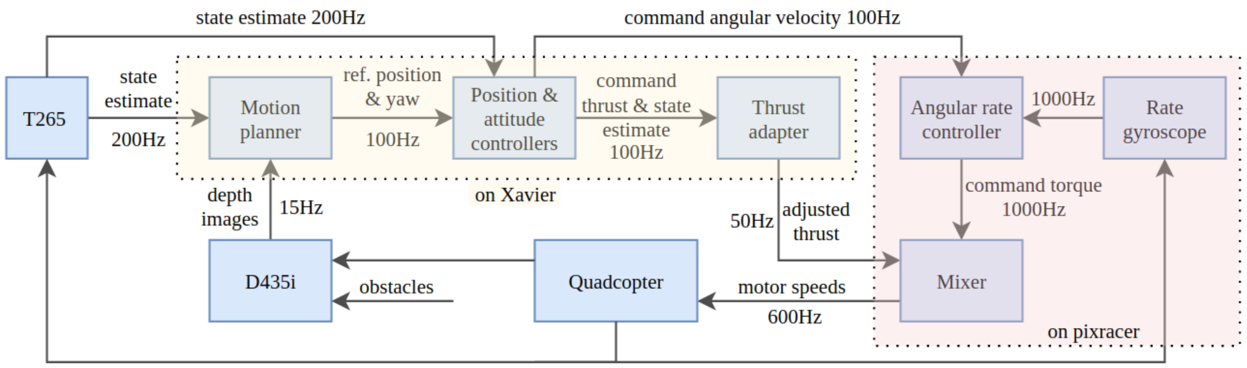


Figure 4.9: A block diagram of the system, showing the relationship between components. The yellow shaded area contains components running on the AGX Xavier on-board computer, while the red shaded area contains components running on the Pixracer flight controller.

each propeller is 229 mm. On the vehicle, an Intel D435i depth camera is installed for collision avoidance and an Intel T265 camera is installed for state estimation. The depth camera is forward-looking to detect obstacles in forward flights, and the T265 camera has a 37-degree angle with the horizontal ground, to track features on the ground for state estimation and to avoid view occlusion from other parts of the vehicle. The GPS and two other cameras (one is a RGB camera and the other one is an infra-red camera) on the vehicle are not used for the vehicle's motion planning. The relationship between the components of the system, is shown in Fig. 4.9. The RAPPIDS motion planner, the position and attitude controllers, and the thrust adapter runs on an on-board computer (Jetson AGX Xavier), at a frequency of 100Hz. The pixracer runs the standard PX4 firmware [63]. It takes in the desired angular velocity and thrust from the Xavier and sends the motor speed commands to the ESCs, which control the motors' rotation speed.

## Obstacle avoidance experiment

The experiments were conducted at a small forest at the Richmond Field Station (37.915535 N, -122.335059 E), as shown in Fig. 4.8. The vehicle's radius $r$ (shown in Fig. 4.2) was set to 0.6m for the planner during the experiment, leaving a minimum safety margin of about 0.3 m between the vehicle and the nearest obstacles. The velocity constraint $v_{max}$ in section 4.2 was set to 3 m/s because of the limit of the T265 tracking camera (a speed of above 3 m/s could make the state estimation of T265 unreliable), and trajectories exceeding this speed limit will be rejected.

The vehicle was first controlled to take off manually to about 1 m above the ground and then switched to autonomous hovering at its current position, which was used as the starting point. The target point was set to be 30 meters forward with respect to the starting point of the vehicle, to fly the vehicle to the other side of the forest. When the vehicle was close to the target point (less than 1 m in this case), the motion planner in Fig. 4.9 stopped generating new trajectories, and the vehicle tracked the last reference trajectory to reach the target point. After the vehicle reached the target point, it hovered there and waited for the pilot to send other commands, e.g. landing. In the experiment the vehicle was able to reach the target point while generating and tracking collision-free trajectories. The path of the vehicle is visualized in Fig. 4.10. The manual take-off and landing part are omitted and only the autonomous collision avoidance flight part is plotted for clarity. With the velocity check in section 4.2 on the sampled trajectories, the velocity on none of the three axis exceeds the velocity limit of 3.0 m/s, shown in Fig. 4.11.

The number of sampled trajectories and better-than-current trajectories (i.e. trajectories that pass all the checks in Algorithm 1 and have a lower cost than the current reference trajectory) throughout the experiment is shown in Fig. 4.12. Thanks to the computational efficiency of the algorithm, a large number of sampled trajectories could be processed on-board in real-time. The current reference trajectory was updated when a better-than-current trajectory was found, which happened most of the time during the flight. When no better trajectory was found (e.g. when the view of the depth camera was occluded by the obstacles),

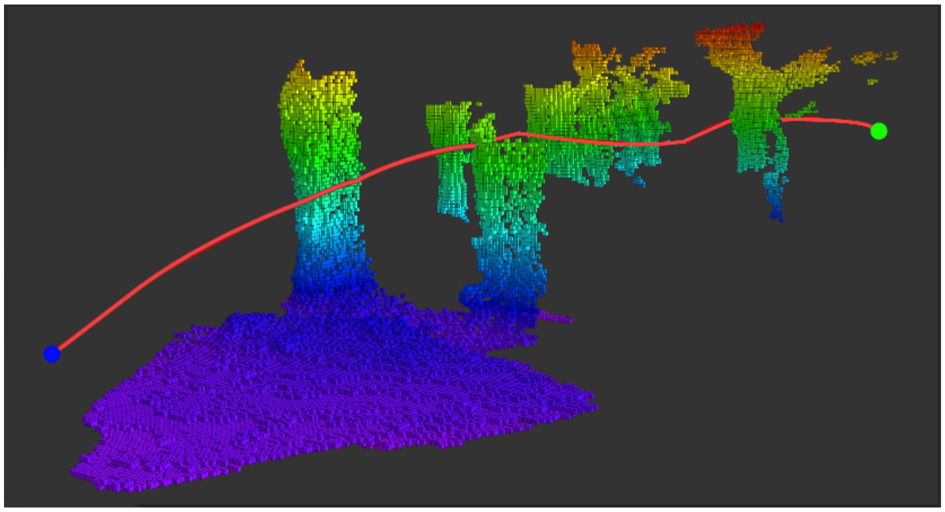


Figure 4.10: Path of the vehicle (red line) during the autonomous collision avoidance flight from the start point (marked with a blue dot) to the end point (marked with a green dot). The trees detected by the depth camera were visualized using Octomap [80]. The distance between the start point and the end point is 30 meters.

the vehicle would track the current trajectory. After 14.7 s, the vehicle was within 1 m to the target point, and the trajectory generator stopped generating new trajectories and followed the last reference trajectory to reach the target point.

## 4.5 Conclusion

In this chapter, we presented various considerations of the RAPPIDS motion planner for outdoor flights. We consider a planner under the assumption of static environment, in which recursive feasibility is guaranteed by the way how collision-free trajectories are constructed. Specifically speaking, the terminal conditions of sampled trajectories are selected to be at rest, and that guarantees safety in that executing a primitive should always lead to the vehicle advancing to the objective and coming to rest without collision. Continuous re-sampling of trajectories, in a receding horizon fashion, allows the vehicle to progress towards the objective. In the event that no samples result in feasible trajectories, the system simply continues the previous trajectory, thereby guaranteeing recursive feasibility.

We also introduced the velocity constraint of the planner to satisfy the speed limit of the visual-inertial odometry camera, increased the trajectory sampling efficiency based on the prior that sampling close to the edge of field of view of the depth camera is prone to result in in-collision trajectories, and used estimated acceleration instead of noisy IMU acceleration measurements. In addition, a new utility function was proposed to consider not

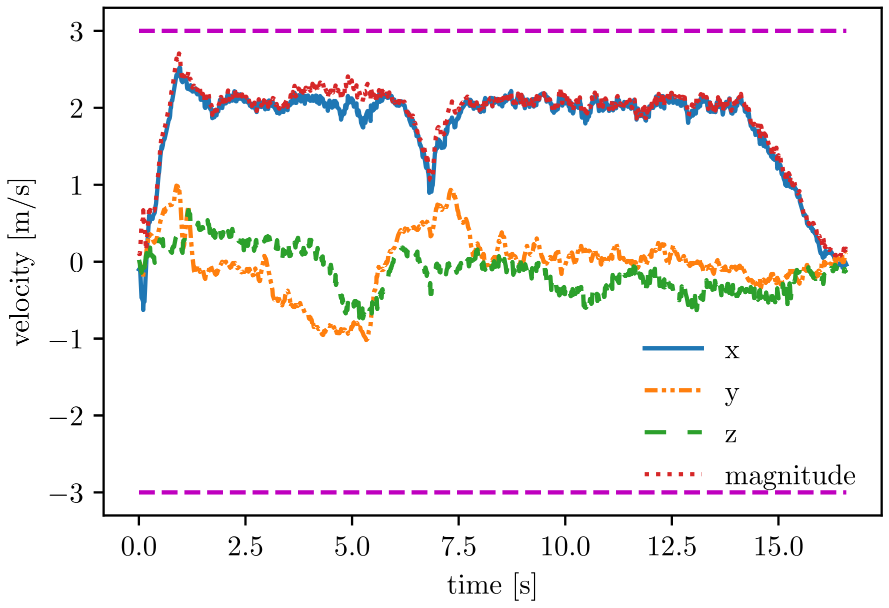


Figure 4.11: The estimated velocity for each axis of the vehicle, as well as the Euclidean norm (magnitude) of the velocity during the autonomous collision avoidance flight. With the velocity check in section 4.2, the velocity on none of the three axis exceeds the velocity limit of 3.0 m/s (marked as magenta dashed lines).

only maximizing the average velocity toward the goal but also keeping the vehicle around the goal. A thrust adaptation method is introduced to compensate decrease in motor thrusts due to voltage drop during flights. Lastly, the experimental results in a challenging outdoor environment were presented, which validated the ability of this system to autonomously navigate through complex obstacles outdoors.

## Acknowledgement

Research in this chapter was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-20-2-0105. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The experimental testbed at the HiPeRLab is the result of contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

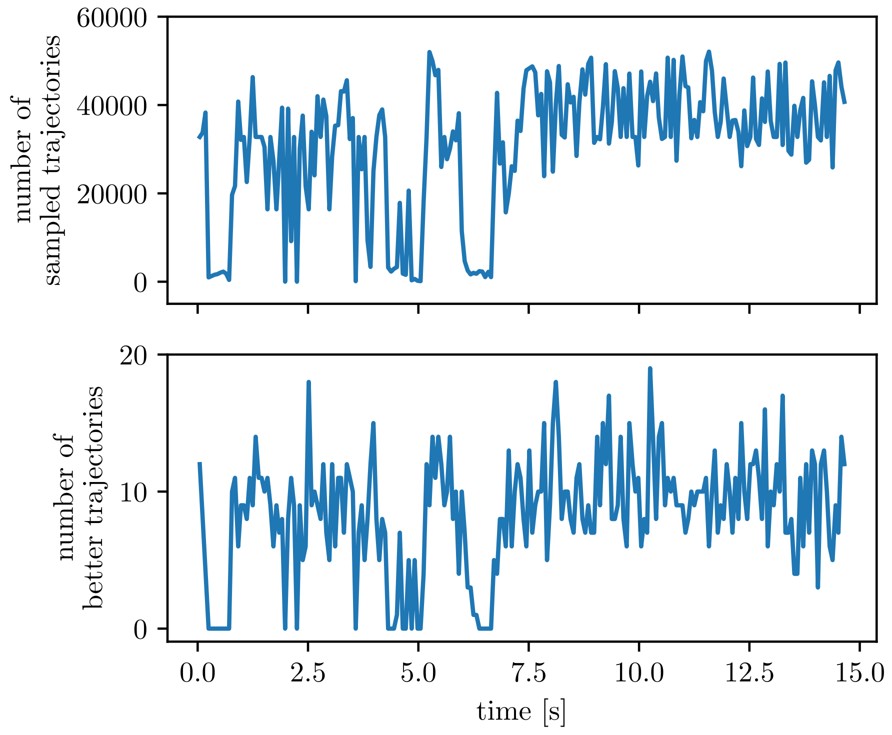


Figure 4.12: A large number of trajectories were sampled during the autonomous flight of the vehicle, as shown in the first row. The sampled trajectories then went through the checks in Algorithm 1, and the number of trajectories that are better than the current trajectory is shown in the second row.

# Chapter 5

# Perception-aware trajectory planning with VIO

Visual inertial odometry (VIO) is widely used for the state estimation of multicopters, but it may function poorly in environments with few visual features or in overly aggressive flights. In this chapter, we propose a perception-aware collision avoidance local planner for multicopters. Our approach is able to fly the vehicle to a goal position at high speed, avoiding obstacles in the environment while achieving good VIO state estimation accuracy. The proposed planner samples a group of minimum jerk trajectories and finds collision-free trajectories among them, which are then evaluated based on their speed to the goal and perception quality. Both the features' motion blur and their locations are considered for the perception quality. The best trajectory from the evaluation is tracked by the vehicle and is updated in a receding horizon manner when new images are received from the camera. All the sampled trajectories have zero speed and acceleration at the end, and the planner assumes no other visual features except those already found by the VIO. As a result, the vehicle will follow the current trajectory to the end and stop safely if no new trajectories are found, avoiding collision or flying into areas without features. Our proposed method can run in real time on a small embedded computer on board. We validated the effectiveness of our proposed approach through experiments in indoor and outdoor environments. Compared to a perception-agnostic planner, the proposed planner kept more features in the camera's view and made the flight less aggressive, making the VIO more accurate. It also reduced VIO failures, which occurred for the perception-agnostic planner but not for the proposed planner. The experiment video can be found at `https://youtu.be/LjZju4KEH9Q`.

- Xiangyu Wu et al. "Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry". In: *arXiv preprint arXiv:2204.03134* (2022)

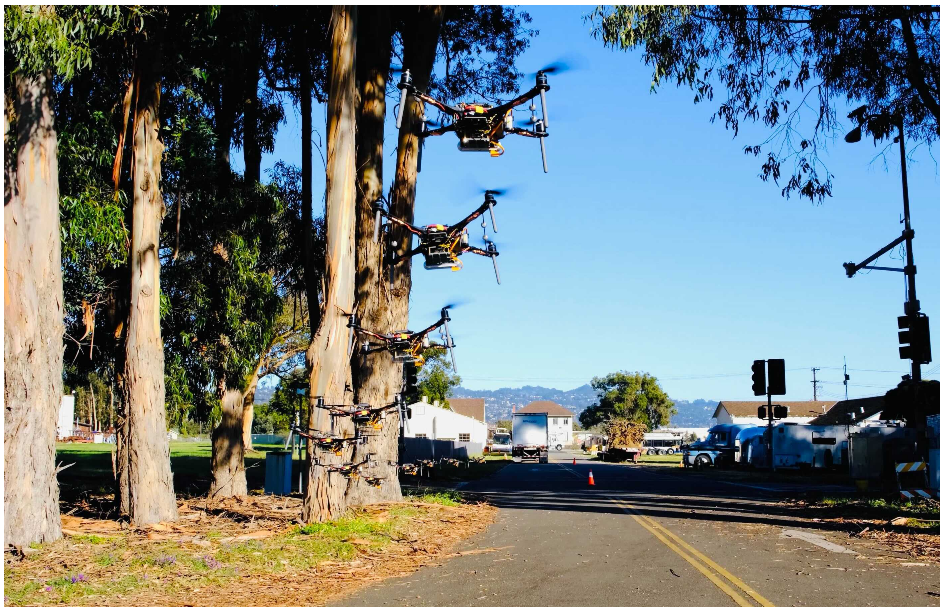


Figure 5.1: The perception-aware planner guides the quadcopter to fly close to areas with more visual features for better VIO accuracy – the paved road has much fewer visual features than the trees. The goal is 20 meters forward from the starting position, in the direction away from the camera.

## 5.1 Introduction

Multicopters are useful for a wide range of applications such as aerial photography [2] inspection [3], and transportation [4] thanks to their simple design and high maneuverability. Accurate state estimation is necessary for these operations, where visual inertial odometry (VIO) is a popular solution: it only requires light-weight, low-power, and low-cost onboard sensors – cameras and inertial measurement units (IMUs), suitable even for small aerial robots [81]. Additionally, VIO does not require other infrastructure in the operating environment. These advantages make it especially useful in applications where the GPS signal is unreliable, such as indoors, in the forest, or near tall buildings.

However, VIO may struggle when the vehicle flies in areas with few visual features or when the motion of the vehicle becomes too aggressive. As a result, the trajectory planning of a vehicle should include perception-awareness: it should consider not only the goal of the mission, but also the trajectory's impact on the VIO. Taking into account state estimation in path planning has drawn increased research interest over the past few years [82, 83, 84]. Most related works in the literature plan multicopter trajectories by solving an optimization problem, encoding the perception-awareness as a cost term or constraint. In [85], the authors add the visibility of features as a constraint in the optimization of B-spline trajectories. The differential flatness property of multicopters is used to speed up the optimization. In [86, 87], the vehicle's trajectories are planned while maintaining a given set of landmarks within the field of view of its on-board camera. The first step is geometric path planning, followed by a time parameterization of the planned path to satisfy the kinodynamic constraints of the quadcopter. In addition to the visibility of the features, maximizing the covisibility of features is also helpful in reducing state estimation error [88]. The goal is to keep features visible in the camera field of view from one keyframe to the next, instead of to maximize the visibility of features in each image. The authors first plan a minimum snap trajectory with only position, and then the yaw angle is planned to maximize the features' covisibility. Authors of [89] additionally take the feature's movement speed into account by adding a perception cost term to reduce the movement speed of the feature points' centroid in the image and keep it close to the image center. Model predictive control (MPC) is used for trajectory planning and the optimization is accelerated by a sequential quadratic program (SQP) approximation. Semantic information is used to plan trajectories in areas with more texture and to avoid places with unreliable visual features, such as lakes [90].

In addition, some works in the literature use sampling-based planners with perception-awareness, where the sampled trajectories are evaluated based on both the original mission goal and their impact on state estimation. Perception-awareness is considered in [91, 92] to improve the mapping and state estimation accuracy during quadcopter exploration. The outer layer planner generates paths that explore the space using the rapidly-exploring random tree (RRT) [93], and the inner layer planner aims to improve the mapping and state estimation accuracy. The authors propagate the state estimation for different paths found by the inner layer planner, and choose the one which minimizes the state estimation uncertainty. In [94], the task of reaching a given goal with the highest accuracy while avoiding obstacles in the environment is investigated. The planner generates candidate trajectories and evaluates them in terms of perception quality, collision probability, and distance to the goal. Given each sampled trajectory, the authors simulate the observed features if the trajectory is followed and construct a least squares problem to estimate the vehicle's pose estimation error.

In this work, we focus on the problem of flying a multicopter to a desired position at a high speed, avoiding obstacles in the environment, while achieving good state estimation accuracy from the VIO. A stereo depth camera with an IMU is used for collision avoidance and VIO. To quickly check if a trajectory is collision-free, we use a sampling-based planner named RAPPIDS [78, 95]. In the next step, given a sampled trajectory that is collision-free, we predict the pose of the vehicle (assuming perfect trajectory tracking) and then the position and velocity of the VIO features in the camera frame. The vehicle pose estimation is constructed as a least-squares problem, and each feature's variance is estimated from its velocity in the image. We then evaluate the perception cost of the trajectory based on the vehicle's predicted position estimation uncertainty if that trajectory is followed. In addition, the speed cost of the trajectory is the negative value of its average speed towards the goal. The trajectory that minimizes the perception cost plus the speed cost is selected as the trajectory to follow. Our planner runs in a receding horizon manner, and it replans every time a new image arrives.

Our proposed planner can reduce the state estimation error of the VIO by planning trajectories that guide the vehicle towards feature-rich areas and by preventing the vehicle from executing overly aggressive trajectories (causing severe motion blur). Meanwhile, the trajectories it plans are collision-free and dynamically feasible. The planner is also computationally efficient enough to run on an onboard embedded computer in real-time. Compared with the existing works in the literature, the contributions of the method we propose in this chapter are:

1. We propose a local planner generating collision-free trajectories that guides the vehicle towards the target, while avoiding regions with few visual features and overly aggressive flights.
2. We propose a perception cost function considering both the motion blur of the features and their locations.
3. Natural adaptation of the trajectory's aggressiveness under environments with different light levels.
4. Experimental validation in both indoor and outdoor environments, showing the effectiveness of the proposed method.

## 5.2 System overview

In this section, we give a brief overview of the proposed perception-aware planning system, a block diagram of which is shown in Fig. 5.2. The goal is to fly the drone to a desired position at a fast speed, avoid obstacles in the way, and achieve a good state estimation accuracy from the VIO. The environment is assumed to be stationary.

The perception-aware planner uses the depth images from the stereo depth camera to detect obstacles in the environment. We use the RAPPIDS planner [78, 95], a memoryless sampling-based planner, to generate a group of collision-free candidate trajectories at low computational cost. We use OpenVINS [96] for VIO, which uses monocular images from the depth camera and IMU measurements to estimate the state of the vehicle, while other feature-based VIO algorithms can also be used with the planner. The VIO also sends tracked features to the perception-aware planner to evaluate the perception cost $c_{\text{perc}}$ of each candidate trajectory. The derivation of the perception cost is detailed in Section 5.3 In addition, we add a flight speed related cost $c_{\text{speed}}$ (defined in Section 5.4) to encourage fast flight towards the goal. A parameter $k_{\text{perc}}$ is introduced to determine the weight of the perception quality, and the collision-free trajectory with the minimum total cost $k_{\text{perc}}c_{\text{perc}} + c_{\text{speed}}$ is chosen as the best trajectory.

The perception-aware planner runs in a receding horizon manner, and it replans every time a new depth image is received. The tracked trajectory is updated if a lower-cost trajectory is found.

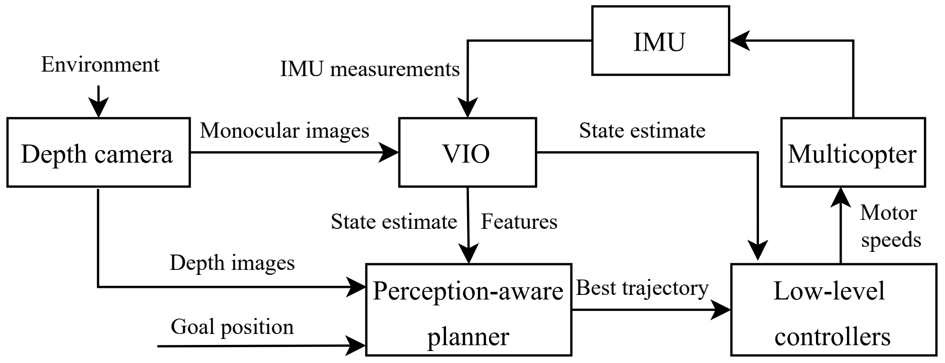


Figure 5.2: Block diagram of the proposed percerption-aware planning system for multicopters.

## 5.3 Perception-aware cost derivation

In this section, we introduce the derivation of the perception cost. We define the world frame W, body frame B, and camera frame C, as shown in Fig. 5.3. Vector and matrix variables are written in boldface. The notations used in this section are summarized in Table 5.1.

The pose of the vehicle's body frame B with respect to the world frame W is given by $\boldsymbol{T}^{WB} = \begin{bmatrix}\boldsymbol{R}^{WB} & \boldsymbol{t}^{WB}\end{bmatrix}$, where $\boldsymbol{R}^{WB}$ and $\boldsymbol{t}^{WB}$ are the orientation and position of B with respect to W. We assume that the vehicle is equipped with a depth camera, whose pose in the vehicle's body frame is given by a rigid body transformation $\boldsymbol{T}^{BC} = \begin{bmatrix}\boldsymbol{R}^{BC} & \boldsymbol{t}^{BC}\end{bmatrix}$. The extrinsics of the camera is given by $\boldsymbol{T}^{CW} = \begin{bmatrix}\boldsymbol{R}^{CW} & \boldsymbol{t}^{CW}\end{bmatrix}$.

The perception cost of a trajectory $c_{\text{perc}}$ represents the uncertainty of vehicle position estimation if the trajectory is followed. Similarly to [94], we estimate the variance of vehicle pose estimation by formulating a least-squares problem. However, we also estimate the variance of the feature points in the image according to their movement speed in the image, instead of assuming a constant variance for the features as in [94]. This helps to take motion blur into account and discourages the multicopter from executing an overly aggressive trajectory, which would adversely affect the VIO accuracy. The function mapping the pose estimation variance to the perception cost $c_{\text{perc}}$ is also different, for faster computation and more intuitive tuning of the perception cost's weight coefficient $k_{\text{perc}}$.

### Cost of a trajectory

Given a trajectory $\boldsymbol{\Gamma}(t) = \begin{bmatrix}x(t) & y(t) & z(t) & \psi(t)\end{bmatrix}^T$ and the 3D positions of VIO features $F_{\text{feature}} := \{\boldsymbol{P}_k\}_{k=1}^{M}$ in the world frame, we need to predict the pose estimation error if the given trajectory is followed. In $\boldsymbol{\Gamma}(t)$, the 3D positions of the vehicle are represented as $\begin{bmatrix}x(t) & y(t) & z(t)\end{bmatrix}^T$, and the yaw angle of the vehicle is represented as $\psi(t)$. We first sample

Table 5.1: Notations used in Section 5.3.

| Symbol | Meaning |
|---|---|
| $c_{\text{perc}}, c_{\text{speed}}, c_{\text{tot}}$ | perception, speed, and total cost |
| $k_{\text{perc}}$ | weight coefficient of perception cost |
| $W, B, C$ | world frame, vehicle body frame, camera frame |
| $\boldsymbol{T}^{WB} = [\boldsymbol{R}^{WB}\ \boldsymbol{t}^{WB}]$ | pose of the vehicle body frame B in the world frame W |
| $\boldsymbol{T}^{BC} = [\boldsymbol{R}^{BC}\ \boldsymbol{t}^{BC}]$ | pose of the camera frame C in vehicle body frame B (rigid transformation) |
| $\boldsymbol{T}^{CW} = [\boldsymbol{R}^{CW}\ \boldsymbol{t}^{CW}]$ | camera extrinsics: pose of the world frame W in the camera frame C |
| subscript $j$ | the $j$th sampled pose of the trajectory |
| subscript $k$ | the $k$th VIO feature |
| $\boldsymbol{I}_j$ | set of visible features at sampled pose $j$ |
| $\boldsymbol{\Gamma}(t)$ $= [x(t)\ y(t)\ z(t)\ \psi(t)]^T$ | a trajectory consisted of 3D positions $x(t), y(t), z(t)$, and yaw angle $\psi(t)$ |
| $\boldsymbol{P}_k = [X_k\ Y_k\ Z_k]^T$, $\boldsymbol{P}'_{kj} = [X'_{kj}\ Y'_{kj}\ Z'_{kj}]^T$ | 3D coordinates of the $k$th feature in W, 3D coordinates of the $k$th feature in C when vehicle is at sampled pose $j$ |
| $\hat{\boldsymbol{b}}_{kj} = [\hat{u}_{kj}\ \hat{v}_{kj}]^T$, $\boldsymbol{b}_{kj} = [u_{kj}\ v_{kj}]^T$ | $k$th feature's observation in image, estimated projection in the image (when vehicle is at sampled pose $j$) |
| $S(\cdot)$ | function converting a $\mathbb{R}^3$ vector to its corresponding skew symmetric matrix |
| $\boldsymbol{\xi}$ | error of estimated camera extrinsics |
| $\boldsymbol{J}_{\boldsymbol{\xi},kj}$ | $\partial \boldsymbol{b}_{kj}/\partial \boldsymbol{\xi}$, Jacobian of the $k$th feature with respect to $\boldsymbol{\xi}$ at sampled pose $j$ |
| $\boldsymbol{\Sigma}_{\boldsymbol{\xi}j}$ | covariance of estimated $\boldsymbol{\xi}$ at sampled pose $j$ |
| $\boldsymbol{v}_j, \boldsymbol{\omega}_j$ | vehicle's velocity, angular velocity at sampled pose $j$ |
| $\sigma^2_{kj,\parallel}, \sigma^2_n$ | feature's variance because of speed, feature's variance because of vibration |
| $\bar{\dot{\boldsymbol{b}}}_{kj} = [\bar{\dot{u}}_{kj}\ \bar{\dot{v}}_{kj}]^T$ | the normalized speed of the $k$th feature in image at sampled pose $j$ |
| $t_{\text{exp}}$ | camera's exposure time |
| $\boldsymbol{\Sigma}_{\boldsymbol{b},kj}$ | covariance of the $k$th feature in the image at sampled pose $j$ |
| $\boldsymbol{\Sigma}_{\boldsymbol{b}j}$ | covariance of visible features in the image at sampled pose $j$ |

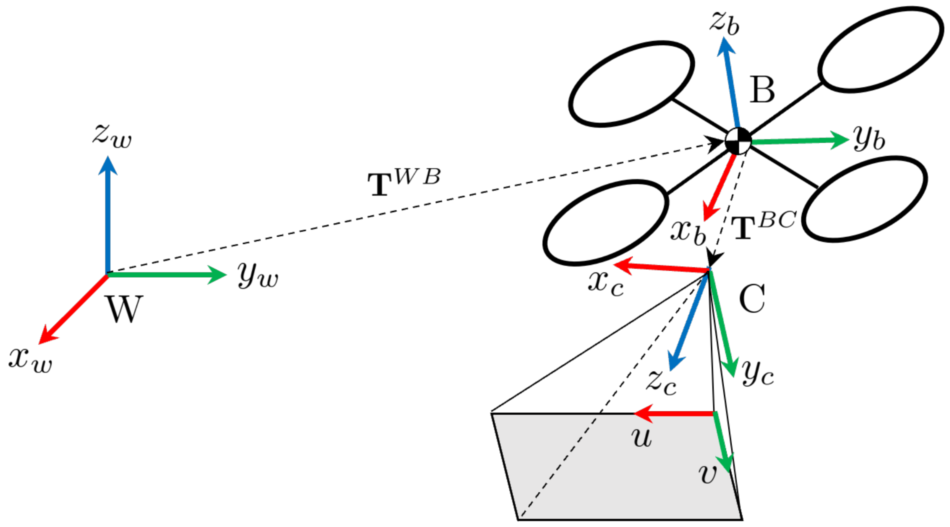


Figure 5.3: An illustration of the world frame W, body frame B and camera frame C. The position and attitude of the vehicle's body frame with respect to the world frame is given by $\boldsymbol{T}^{WB}$. The position and attitude of the camera frame in the body frame is given by $\boldsymbol{T}^{BC}$, which is a rigid body transformation.

the poses of the vehicle $\{\boldsymbol{T}_j^{WB}\}_{j=1}^{N}$ along the trajectory at a fixed time interval. For a sampled pose $\boldsymbol{T}_j^{WB}$, we can find the extrinsics of the camera $\boldsymbol{T}_j^{CW}$ and thus the visible features in $F_{\text{feature}}$ , as shown in Fig. 5.4. We denote the indices of visible features at the camera pose $\boldsymbol{T}_j^{CW}$ as $\boldsymbol{I}_j$.

Then if the vehicle moves to $\boldsymbol{T}_j^{WB}$, the extrinsics of the camera $\boldsymbol{T}_j^{CW}$ can be estimated by the following least squares problem:

$$\boldsymbol{T}_j^{*} = \arg\min_{\boldsymbol{T}} \sum_{k\in \boldsymbol{I}_j} \left\| \hat{\boldsymbol{b}}_{kj} - \operatorname{proj}\left(\boldsymbol{T}\boldsymbol{P}_k\right)\right\|_2^2, \tag{5.1}$$

where proj$(\cdot)$ represents the projection function of the camera, and $\hat{\boldsymbol{b}}_{kj} = \begin{bmatrix}\hat{u}_{kj} & \hat{v}_{kj}\end{bmatrix}^T$ is the $k$th feature point's observation in the image, which has noise due to motion blur and camera lens imperfections.

The optimization problem (5.1) can usually be solved in an iterative way. It can be converted into the following form:

$$\boldsymbol{\xi}^{*} = \arg\min_{\boldsymbol{\xi}} \sum_{k\in \boldsymbol{I}_j} \left\| \hat{\boldsymbol{b}}_{kj} - \operatorname{proj}\left(\exp\left(\boldsymbol{\xi}\right)\boldsymbol{T}_j^{CW}\boldsymbol{P}_k\right)\right\|_2^2, \tag{5.2}$$

where $\boldsymbol{\xi}$ is in $\mathfrak{se}(3)$ and represents the error of the estimated camera extrinsics. The function $\exp(\cdot)$ maps $\mathfrak{se}(3)$ to $SE(3)$. The estimated camera extrinsics $\boldsymbol{T}_j^{CW}$ is updated iteratively by $\boldsymbol{T}_j^{CW} = \exp\left(\boldsymbol{\xi}\right)\boldsymbol{T}_j^{CW}$.

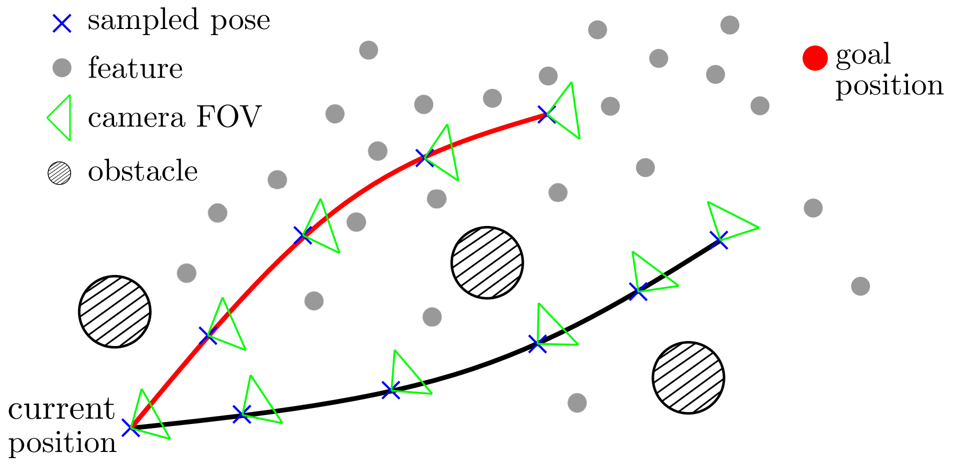


Figure 5.4: We discretize sampled trajectories at a fixed time interval to evaluate their perception cost. When the camera's exposure time is short and motion blur is not significant, the trajectory in red has lower perception cost than the trajectory in black. Because it keeps more features in the camera's view and is also closer to the features.

Define $\boldsymbol{b}_{kj} = \mathrm{proj}\left(\boldsymbol{T}_j^{CW}\boldsymbol{P}_k\right)$, which is the projected coordinates of the $k$th feature in the image frame. The equation (5.2) can be solved through linearization at the current estimation of $\boldsymbol{T}_j^{CW}$:

$$\begin{aligned}\boldsymbol{\xi}^* &= \arg\min_{\boldsymbol{\xi}} \sum_{k\in\boldsymbol{I}_j} \left\| \hat{\boldsymbol{b}}_{kj} - \mathrm{proj}\left(\boldsymbol{T}_j^{CW}\boldsymbol{P}_k\right) - \boldsymbol{J}_{\boldsymbol{\xi},kj}\boldsymbol{\xi} \right\|_2^2 \\ &= \arg\min_{\boldsymbol{\xi}} \sum_{k\in\boldsymbol{I}_j} \left\| \hat{\boldsymbol{b}}_{kj} - \boldsymbol{b}_{kj} - \boldsymbol{J}_{\boldsymbol{\xi},kj}\boldsymbol{\xi} \right\|_2^2,\end{aligned} \tag{5.3}$$

where $\boldsymbol{J}_{\boldsymbol{\xi},kj} = \frac{\partial \boldsymbol{b}_{kj}}{\partial \boldsymbol{\xi}},\ k \in \boldsymbol{I}_j$.

Denote $\boldsymbol{P}'_{kj} = \begin{bmatrix} X'_{kj} & Y'_{kj} & Z'_{kj} \end{bmatrix}^T = \boldsymbol{T}_j^{CW}\boldsymbol{P}_k$ as the coordinates of features in the camera frame, then we have:

$$\boldsymbol{J}_{\boldsymbol{\xi},kj} = \frac{\partial \boldsymbol{b}_{kj}}{\partial \boldsymbol{\xi}} = \frac{\partial \boldsymbol{b}_{kj}}{\partial \boldsymbol{P}'_{kj}} \frac{\partial \boldsymbol{P}'_{kj}}{\partial \boldsymbol{\xi}}. \tag{5.4}$$

Denote the camera's focal length as $f_x, f_y$ and its focal point's coordinates in the image as $\begin{bmatrix} c_x & c_y \end{bmatrix}^T$, from the pinhole camera model, we have:

$$\boldsymbol{b}_{kj} = \begin{bmatrix} u_{kj} \\ v_{kj} \end{bmatrix} = \mathrm{proj}\left(\boldsymbol{P}'_{kj}\right) = \begin{bmatrix} f_x \frac{X'_{kj}}{Z'_{kj}} + c_x \\ f_y \frac{Y'_{kj}}{Z'_{kj}} + c_y \end{bmatrix}. \tag{5.5}$$

Besides, with some Lie algebra derivations, we can get

$$\frac{\partial \boldsymbol{P}_{kj}^{'}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & -S\left(\boldsymbol{P}_{kj}^{'}\right) \end{bmatrix}, \tag{5.6}$$

where the function $S(\cdot)$ converts a vector in $\mathbb{R}^3$ to its corresponding 3-by-3 skew-symmetric matrix. By combining (5.4), (5.5) and (5.6), we can get the expression of the Jacobian:

$$\boldsymbol{J}_{\boldsymbol{\xi},kj} = \begin{bmatrix} \frac{f_x}{Z_{kj}^{'}} & 0 & -\frac{f_x X_{kj}^{'}}{Z_{kj}^{'\,2}} & -\frac{f_x X_{kj}^{'} Y_{kj}^{'}}{Z_{kj}^{'\,2}} & f_x + \frac{f_x X_{kj}^{'\,2}}{Z_{kj}^{'\,2}} & -\frac{f_x Y_{kj}^{'}}{Z_{kj}^{'}} \\ 0 & \frac{f_y}{Z_{kj}^{'}} & -\frac{f_y Y_{kj}^{'}}{Z_{kj}^{'\,2}} & -f_y - \frac{f_y Y_{kj}^{'\,2}}{Z_{kj}^{'\,2}} & \frac{f_y X_{kj}^{'} Y_{kj}^{'}}{Z_{kj}^{'\,2}} & \frac{f_y X_{kj}^{'}}{Z_{kj}^{'}} \end{bmatrix} \tag{5.7}$$

As a result, the covariance of the estimated parameter $\boldsymbol{\xi}$ in (5.3) is given by

$$\boldsymbol{\Sigma}_{\boldsymbol{\xi}j} = (\boldsymbol{J}_j^T \boldsymbol{J}_j)^{-1} \boldsymbol{J}_j^T \boldsymbol{\Sigma}_{\boldsymbol{b}j} \boldsymbol{J}_j (\boldsymbol{J}_j^T \boldsymbol{J}_j)^{-1} = (\boldsymbol{J}_j^T \boldsymbol{\Sigma}_{\boldsymbol{b}j}^{-1} \boldsymbol{J}_j)^{-1}, \tag{5.8}$$

where the matrix $\boldsymbol{J}_j$ is stacked up of $\boldsymbol{J}_{\boldsymbol{\xi},kj}$, and $\boldsymbol{\Sigma}_{\boldsymbol{b}j}$ is the covariance of visible features, which is related to the feature's speed in the image plane and is derived in the following Section 5.3.

For computational efficiency and more intuitive tuning of $k_{\text{perc}}$, we only consider the estimation variance of the first three elements of $\boldsymbol{\xi}$, which corresponds to the estimated position of the vehicle. The perception cost of a trajectory is defined as:

$$c_{\text{perc}} = \sum_{j=1}^{N} \frac{\sqrt{\boldsymbol{\Sigma}_j^{(1,1)}} + \sqrt{\boldsymbol{\Sigma}_j^{(2,2)}} + \sqrt{\boldsymbol{\Sigma}_j^{(3,3)}}}{3N}, \tag{5.9}$$

which corresponds to the mean sum of per-axis standard deviation of position estimate over the sampled times.

## Feature variance estimation

When the vehicle moves to $\boldsymbol{T}_j^{WB}$, the 3D position of a visible VIO feature in the camera's frame $\boldsymbol{P}_{kj}^{'} = \begin{bmatrix} X_{kj}^{'} & Y_{kj}^{'} & Z_{kj}^{'} \end{bmatrix}^T$ is given by:

$$\boldsymbol{P}_{kj}^{'} = \boldsymbol{T}^{CW} \boldsymbol{P}_k = \boldsymbol{R}^{CW} \left( \boldsymbol{P}_k - \boldsymbol{R}^{WB} \boldsymbol{t}^{BC} - \boldsymbol{t}^{WB} \right). \tag{5.10}$$

To obtain the feature's velocity in the camera frame, we differentiate (5.10) with respect to time:

$$\dot{\boldsymbol{P}}_{kj}^{'} = \begin{bmatrix} \dot{X}_{kj}^{'} & \dot{Y}_{kj}^{'} & \dot{Z}_{kj}^{'} \end{bmatrix}^T = -\boldsymbol{R}^{CB} S(\boldsymbol{\omega}_j) \boldsymbol{R}^{CB} \boldsymbol{P}_k^{'} - \boldsymbol{R}^{CB} S(\boldsymbol{\omega}_j) \boldsymbol{t}^{BC} - \boldsymbol{R}_j^{CW} \boldsymbol{v}_j^{WB} \tag{5.11}$$

where $\boldsymbol{v}_j^{WB}$ is the vehicle's velocity in the world frame and $\boldsymbol{w}_j$ is the vehicle's angular velocity in the body frame. They can be predicted given a trajectory $\boldsymbol{\Gamma}(t)$ [97]. Also, $\boldsymbol{R}^{CB} = \left(\boldsymbol{R}^{BC}\right)^{-1}$.

We then differentiate (5.5) with respect to time, to get the feature's speed in the image plane:

$$\dot{u}_{kj} = f_x \frac{\dot{X}'_{kj} Z'_{kj} - X'_{kj} \dot{Z}'_{kj}}{{Z'_{kj}}^2}, \quad \dot{v}_{kj} = f_y \frac{\dot{Y}'_{kj} Z'_{kj} - Y'_{kj} \dot{Z}'_{kj}}{{Z'_{kj}}^2}. \tag{5.12}$$

Due to the movement of the camera, a feature point will have some motion blur in the image. Denote the camera's exposure time as $t_{\text{exp}}$, and the feature could be approximated by a straight line of length $t_{\text{exp}} \sqrt{\dot{u}_{kj}^2 + \dot{v}_{kj}^2}$ when the exposure time is relatively short. We assume that the feature point is uniformly distributed on this straight line, whose direction is the feature's speed ($\dot{\boldsymbol{b}}_{kj}$) direction in the image plane. The feature's variance in this direction is approximated by:

$$\sigma_{kj,\parallel}^2 = \frac{t_{\text{exp}}^2 \left(\dot{u}_{kj}^2 + \dot{v}_{kj}^2\right)}{12}. \tag{5.13}$$

In addition, due to vehicle vibration and the imperfect lens, the feature has an additional variance $\sigma_n^2$, which we assume to be omnidirectional and can be measured experimentally.

Define the normalized feature speed in the image to be $\bar{\dot{\boldsymbol{b}}}_{kj} = \begin{bmatrix} \bar{\dot{u}}_{kj} & \bar{\dot{v}}_{kj} \end{bmatrix}^T$. Then, the covariance of the feature in the image plane is given by:

$$\begin{gathered} \boldsymbol{\Sigma}_{\boldsymbol{b},kj} = \begin{bmatrix} \bar{\dot{u}}_{kj} & -\bar{\dot{v}}_{kj} \\ \bar{\dot{v}}_{kj} & \bar{\dot{u}}_{kj} \end{bmatrix} \begin{bmatrix} \sigma_{kj,\parallel}^2 + \sigma_n^2 & 0 \\ 0 & \sigma_n^2 \end{bmatrix} \begin{bmatrix} \bar{\dot{u}}_{kj} & -\bar{\dot{v}}_{kj} \\ \bar{\dot{v}}_{kj} & \bar{\dot{u}}_{kj} \end{bmatrix}^T = \\ \begin{bmatrix} \bar{\dot{u}}_{kj}^2 \sigma_{kj,\parallel}^2 + \left(\bar{\dot{u}}_{kj}^2 + \bar{\dot{v}}_{kj}^2\right) \sigma_n^2 & \bar{\dot{u}}_{kj} \bar{\dot{v}}_{kj} \sigma_{kj,\parallel}^2 \\ \bar{\dot{u}}_{kj} \bar{\dot{v}}_{kj} \sigma_{kj,\parallel}^2 & \bar{\dot{v}}_{kj}^2 \sigma_{kj,\parallel}^2 + \left(\bar{\dot{u}}_{kj}^2 + \bar{\dot{v}}_{kj}^2\right) \sigma_n^2 \end{bmatrix} \end{gathered} \tag{5.14}$$

We can then get the covariance matrix of the visible features in (5.8):

$$\boldsymbol{\Sigma}_{\boldsymbol{b}j} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{b},k_1 j} & 0 & \dots & 0 \\ 0 & \boldsymbol{\Sigma}_{\boldsymbol{b},k_2 j} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{\Sigma}_{\boldsymbol{b},k_m j} \end{bmatrix}, \tag{5.15}$$

where $k_1, k_2, \dots, k_m \in \boldsymbol{I}_j$. By substituting (5.15) into (5.8) and (5.9), we can get the perception cost $c_{\text{perc}}$ of a sampled trajectory.

## 5.4 Perception-aware trajectory planning

In this section, we briefly introduce the RAPPIDS planner that we use to generate collision-free trajectories, which is first introduced in [78] and is improved in [95]. In addition, this section introduces how the perception-aware cost is integrated with the RAPPIDS planner to reduce the multicopter's state estimation uncertainty of VIO.

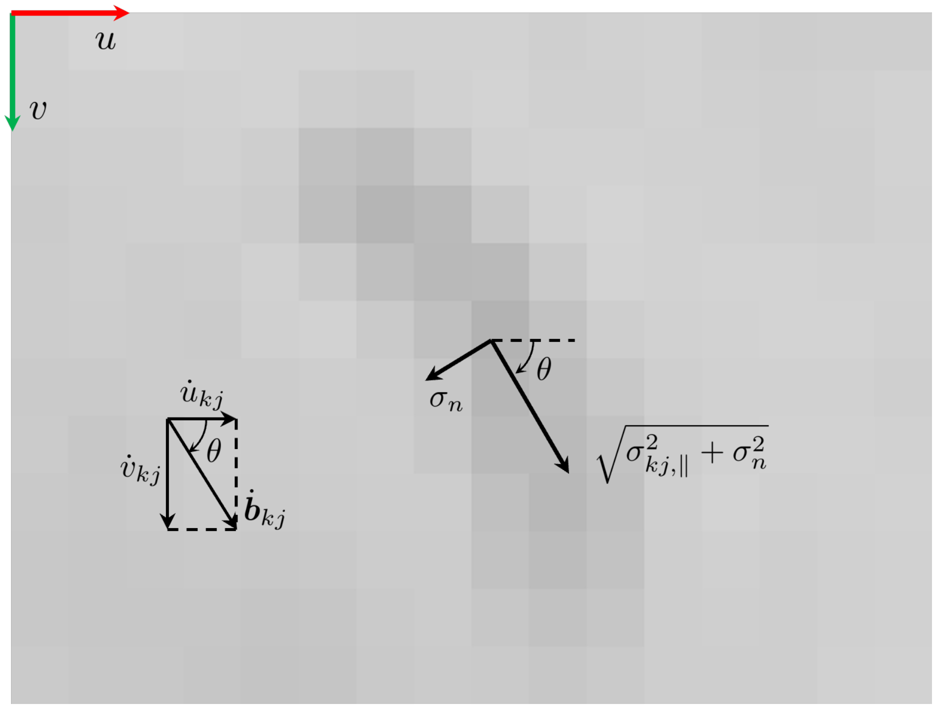

$u$
$v$
$\dot{u}_{kj}$
$\dot{v}_{kj}$
$\theta$
$\dot{\boldsymbol{b}}_{kj}$
$\sigma_n$
$\theta$
$\sqrt{\sigma_{kj,\parallel}^2 + \sigma_n^2}$

Figure 5.5: A feature point becomes a blurry line in the image because of motion blur. The angle of the line is $\theta$, which is the same as the feature's speed in the image plane $\dot{\boldsymbol{b}}_{kj}$. The standard deviation of the feature's position on the speed direction is $\sqrt{\sigma_{kj,\parallel}^2 + \sigma_n^2}$, and is $\sigma_n$ on the perpendicular direction.

## RAPPIDS planner overview

### Trajectory sampling

The RAPPIDS planner samples fifth-order minimum jerk polynomial trajectories $\boldsymbol{s}(t)$ with different duration $T$ and end position $\boldsymbol{s}_T$ using a computationally efficient planner proposed in [98]. The sampled trajectories then go through checks to find if they are input feasible, below the flight speed limit (for safety), and collision-free. The planner replans when a new depth image arrives, to take into account the latest obstacle information.

The sampled trajectories are described as:

$$\boldsymbol{s}(t) = \frac{\boldsymbol{\alpha}}{120}t^5 + \frac{\boldsymbol{\beta}}{24}t^4 + \frac{\boldsymbol{\gamma}}{6}t^3 + \frac{\ddot{\boldsymbol{s}}(0)}{2}t^2 + \dot{\boldsymbol{s}}(0)t + \boldsymbol{s}(0),\ t \in [0,\ T] \tag{5.16}$$

where $\boldsymbol{s}(0)$, $\dot{\boldsymbol{s}}(0)$, and $\ddot{\boldsymbol{s}}(0)$ are the position, velocity, and acceleration of the vehicle at the start time of the trajectory. The terminal condition is selected to be at rest to ensure safety, which requires $\boldsymbol{s}(T) = \boldsymbol{s}_T$ and $\dot{\boldsymbol{s}}(T) = \ddot{\boldsymbol{s}}(T) = 0$. Even if the planner cannot find a new collision-free trajectory (when new depth images arrive), the vehicle can continue following the current trajectory and stop safely. The coefficients $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ can be solved in closed form [98].

The yaw angle $\psi(t)$ of the trajectory is selected such that the multicopter always faces toward the target point, to better see the obstacles on the way:

$$\psi(t) = \text{arctan2}\left(\begin{bmatrix}0\ 1\ 0\end{bmatrix}(\boldsymbol{s}_G - \boldsymbol{s}(t)),\ \begin{bmatrix}1\ 0\ 0\end{bmatrix}(\boldsymbol{s}_G - \boldsymbol{s}(t))\right), \tag{5.17}$$

where $\boldsymbol{s}_G$ and $\boldsymbol{s}(t)$ are the positions of the goal and the vehicle's position in the world frame, respectively.

### Collision check

The RAPPIDS planner checks whether a sampled trajectory collides with obstacles by partitioning the free space into rectangular pyramids and checks if the trajectory remains in the union of these pyramids. In collision check, the vehicle is simplified as the smallest sphere $S_c$ containing the vehicle. The space partitioning process is illustrated in Fig. 5.6. Firstly, we can get the free space $\mathcal{F}$ and the occupied space $\mathcal{O}$ based on the depth image. To avoid collisions with potential unseen obstacles, we treat all spaces $\mathcal{U}$ outside the depth camera's field of view that are $l$ distance away from the vehicle as occupied. The next step is to search for the depth pixel closest to the end position of the trajectory $\boldsymbol{s}(T)$, marked as $\bar{\boldsymbol{s}}_1$ in the figure. Then, starting with the nearest depth pixel and reading the surrounding depth pixels in a spiral sequence, we find the largest possible rectangular space $\mathcal{P}_{exp1}$ that does not intrude into the occupied space $\mathcal{O}$. Finally, a pyramid $\mathcal{P}_1$ is created by shrinking the expanded pyramid $\mathcal{P}_{exp1}$ with the vehicle's radius $r$.

Whether the sampled trajectory is within the union of generated collision-free pyramids can be determined efficiently using the method proposed in [79]. If a sampled trajectory intersects with the union of existing pyramids, the algorithm tries to generate a new pyramid $\mathcal{P}_2$, starting the search from the intersection point, marked as $\bar{\boldsymbol{s}}_2$ in Fig. 5.6. The pyramid generation process continues until the trajectory is within the union of the pyramids – the trajectory is collision-free, or when no new pyramid could be generated – the trajectory can collide with obstacles. A detailed description of this algorithm can be found at [78].

## Selection of the best trajectory

To encourage fast flight towards the target point (following [95]), we define the speed cost $c_{\text{speed}}$ as:

$$c_{\text{speed}} = -\frac{\|\boldsymbol{s}_G - \boldsymbol{s}(0)\|_2 - \|\boldsymbol{s}_G - \boldsymbol{s}(T)\|_2}{T}, \tag{5.18}$$

which is the negative of the average flight speed towards the goal for a sampled trajectory.

There is a trade-off between fast flight and state estimation quality, especially in indoor environments where the camera's exposure time is long. Flying at a fast speed will increase

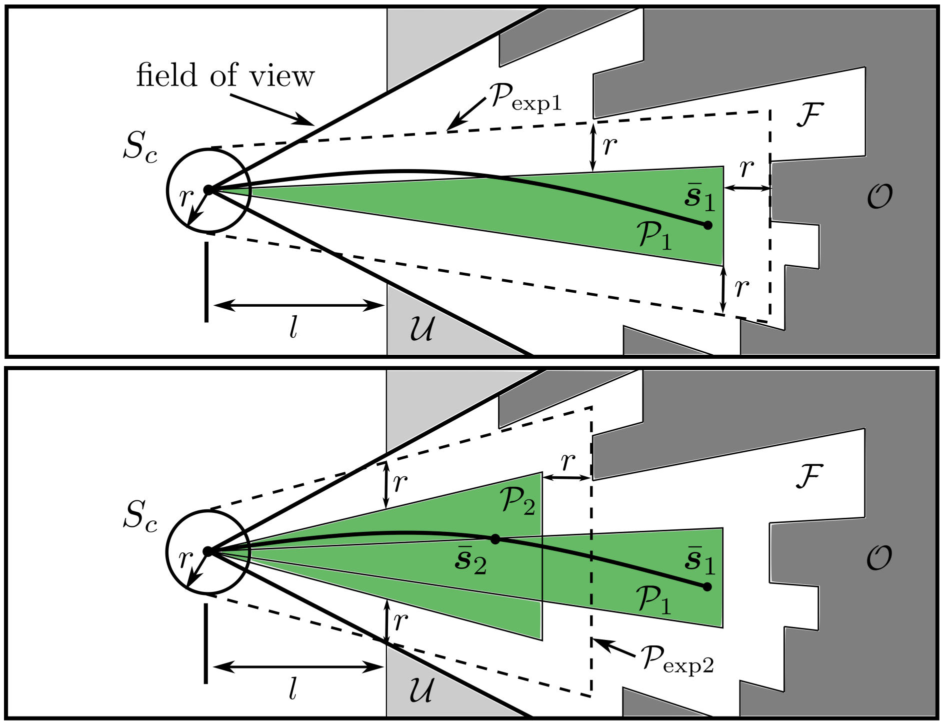


Figure 5.6: The RAPPIDS planner partitions the free space with rectangular pyramids (shown in green) for fast collision check of a sampled trajectory.

the motion blur, which makes the VIO less accurate and causes the perception cost $c_{\text{perc}}$ to increase. As a result, we introduce a coefficient $k_{\text{perc}}$ to determine the importance of the VIO quality. The larger we set $k_{\text{perc}}$, the more weight we put on the state estimation quality over fast flight and vice versa.

Among the sampled trajectories that are collision-free from RAPPIDS, the trajectory with the minimum total cost $c_{\text{tot}} = k_{\text{perc}}c_{\text{perc}} + c_{\text{speed}}$ is chosen as the trajectory to follow. The proposed perception-aware planner is run in a receding horizon manner, to take into account the latest information of the obstacles in the environment and the new feature points found by the VIO. It tries to find a trajectory with lower total cost $c_{\text{tot}}$ every time a new depth image arrives and the current trajectory will continue to be followed if no better trajectory is found. Since each generated trajectory has zero speed and acceleration at the end, and the planner assumes the only features in the environment are those found by the VIO, the planned trajectory is always safe. The vehicle would stop safely if no new collision-free trajectory that allows the camera to see tracked VIO features could be found, avoiding the vehicle flying to areas with no VIO features.

## 5.5 Experimental results

We validate the effectiveness of the proposed perception-aware planning method in both indoor and outdoor environments, by comparing it with the original perception-agnostic RAPPIDS planner. The perception-aware planner is shown to improve the VIO's state estimation accuracy and the number of tracked features in the camera's field of view. The experiment video can be found at `https://youtu.be/LjZju4KEH9Q`.

### Hardware setup

A custom-built quadcopter was used during the experiments, as shown in Fig. 5.7. It weighs 1.5 kg, and the distance between two diagonal motors is 500 mm. The diameter of each propeller is 254 mm (10 inches). The vehicle is equipped with a forward-looking Intel Realsense D455 depth camera for collision avoidance and VIO (with structure light turned off). The VIO uses images from the left and right cameras of D455 at 15Hz and IMU data at 400Hz to give the estimated states of the vehicle. The RAPPIDS planner uses the depth images (15Hz) to generate collision-free trajectories.

The proposed perception-aware planner and the VIO run on a small onboard computer (Qualcomm RB5). The trajectories generated by the perception-aware planner are sent to the Pixracer flight controller running the standard PX4 firmware [63], and are tracked by the low-level position and attitude controllers run on the flight controller.

### Indoor experiments

In indoor experiments, we used a motion capture system to provide the ground truth for the vehicle's position. As shown in Fig. 5.8, the vehicle first took off to 1.2 meters in height, flew to the target point 4 meters forward avoiding the obstacles, and then landed. The experiments were repeated for 12 times each for the proposed perception-aware planner and the original RAPPIDS planner, respectively. The weight of the perception cost $k_{perc}$ was set to 100. The camera's exposure time $t_{\exp}$ was set to 8 milliseconds. Since the planner was only used in the collision avoidance flight, not in the taking-off and landing stages, we exclude the landing stage when comparing the two planners to minimize the uncertainty they introduce.

In the 12 tests, the VIO diverged twice for the original perception-agnostic RAPPIDS planner, while the VIO divergence did not occur for the perception-aware planner. Excluding the VIO diverged cases, the performance of the perception-agnostic and perception-aware planners is compared in Table 5.2. We can see that, on average, the perception-aware planner reduced the final position estimation error (root mean square error) by 19%. Furthermore, the perception-aware planner significantly reduced the aggressiveness of the flights, which can be seen in the reduction of the angular velocity by 24%. This helped reduce the motion blur of the VIO feature points and improved the accuracy of the state estimation. The flight speed was only slightly slower than the perception-agnostic planner by approximately

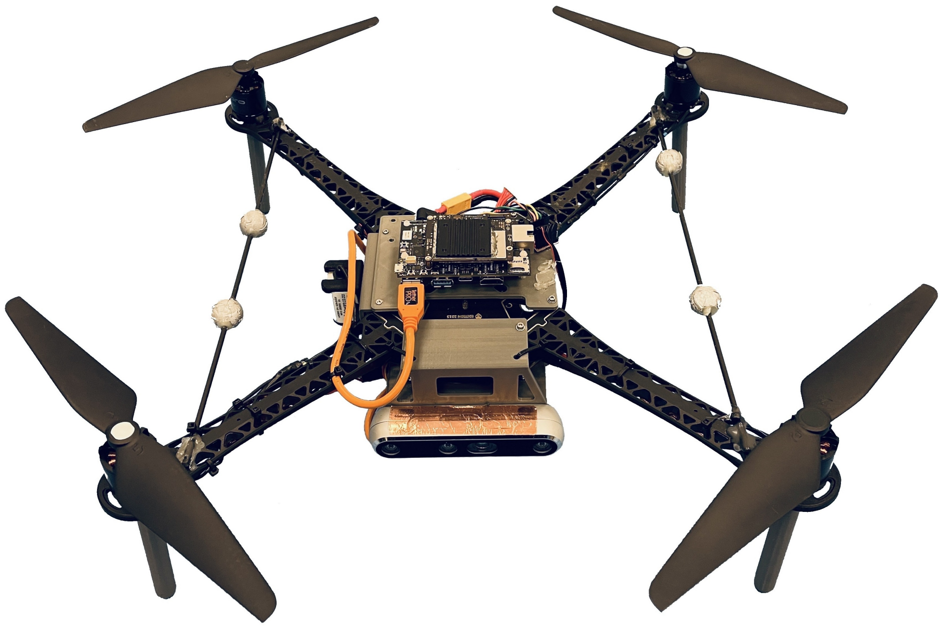


Figure 5.7: The quadcopter used in the experiments. The distance between two diagonal motors is 500 mm.

6%. In addition, the perception-aware planner kept slightly more feature points within the camera's field of view (6%). The difference in the average feature number was small due to the space-constrained experiment setup, making the geometric paths the vehicle could take similar.

Table 5.2: Comparison of indoor experimental results between the proposed perception-aware planner and the original perception-agnostic planner.

| | original | proposed | difference |
|---|---|---|---|
| mean final pos. est. error [m] | 0.141 | 0.114 | -19.1% |
| mean angular velocity [rad/s] | 1.390 | 1.052 | -24.3% |
| mean feature number in FOV | 97.892 | 103.806 | +6.0% |
| mean speed [m/s] | 1.376 | 1.298 | -5.7% |

## Outdoor experiments

In the outdoor experiments, the vehicle first took off to 2 meters in height, flew to the specified target point 20 meters forward, and then landed. The tests were conducted along a

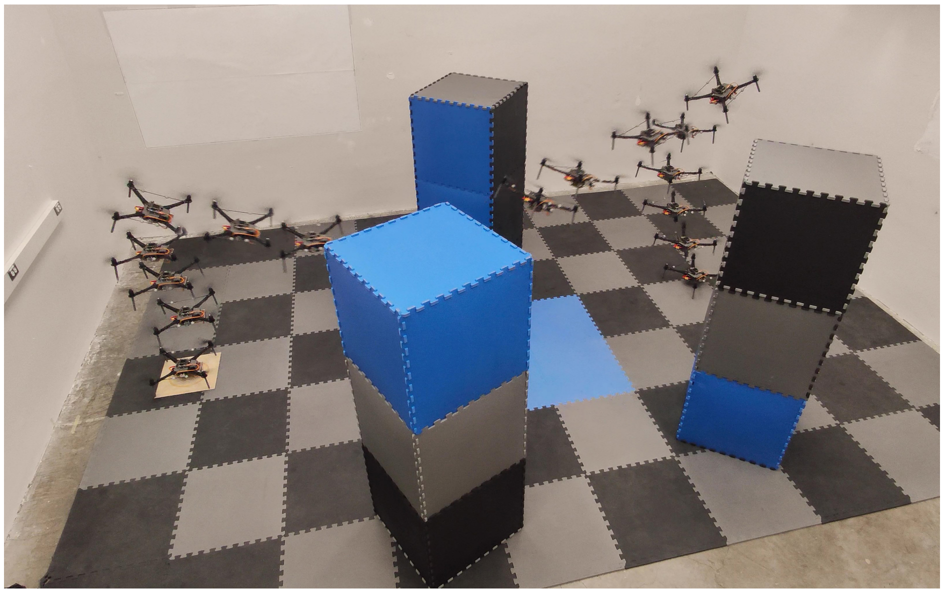


Figure 5.8: The indoor experiment setup. The vehicle's goal is 4 meters ahead of the starting point and the vehicle flies from left to right in the image. The proposed perception aware planner guides the vehicle to reach the goal avoiding the obstacles on its way, while ensuring good state estimation quality from the VIO.

road near a small forest, where there were much more visual features on the trees than on the road. The perception-aware planner guided the vehicle to fly closer to the forest to see more features and at a closer distance to increase the VIO's accuracy, as shown in Fig. 5.1. On the contrary, the original perception-agnostic planner flew the vehicle directly to the goal from the starting position. The experiment was repeated three times for the perception-aware and perception-agnostic planner, respectively.

The experimental results are summarized in Table 5.3, since both planners were only used in the forward flight, we exclude the taking-off and landing stages. We can see that the proposed perception-aware planner reduced the standard deviation of position estimation by 17.8% and increased the number of features in the camera's field of view (FOV) by 12.4% compared to the original perception-agnostic planner. The detailed result for each test is shown in Fig. 5.9. In addition, the perception-aware planner prevented the vehicle from seeing very few features in the camera's FOV, which happened at around 1 second in test 3 of the perception-agnostic planner (marked in red in Fig. 5.9) and would cause the triangulation to fail. In test 2 of the perception-agnostic planner, the camera saw a small number of features at around 6 second (marked by ①), causing an increase in position estimation uncertainty. The standard deviation of the position estimation (i.e. (5.9) at the current pose of the vehicle, N = 1) is used to evaluate the performance of the VIO instead of the error of position estimation in indoor tests, due to the lack of ground truth from the motion capture system. Because there was some randomness in the VIO's feature selection, the distribution of features was different at the beginning of the flights, causing the position estimation's standard deviation to be different (marked by ②).

Since there was abundant light outdoors, the exposure time $t_{\text{exp}}$ for the camera was set to 0.05 millisecond, much faster than the indoor experiments. This short exposure time meant that the motion blur was much smaller compared to indoors, and the features' motion speed played a very small role in a feature's covariance (5.14). As a result, the angular velocity of the perception-aware planner was only 3.9% lower than that of the perception-agnostic planner, showing similar aggressiveness in flight. The aggressiveness of trajectories generated by the perception-aware planner naturally changes under different light levels, rather than requiring manual parameter tuning, making it easy to deploy. Its average flight speed was slightly slower than the perception-agnostic planner, similar to the indoor experiments.

Table 5.3: Comparison of outdoor experimental results between the proposed perception-aware planner and the original perception-agnostic planner.

| | original | proposed | difference |
|---|---|---|---|
| mean pos. est. std [m] | 0.0259 | 0.0213 | -17.8% |
| mean angular velocity [rad/s] | 0.821 | 0.789 | -3.9% |
| mean feature number in FOV | 45.379 | 50.997 | +12.4% |
| mean speed [m/s] | 2.419 | 2.225 | -8.0% |

## 5.6 Conclusion

In this chapter, we proposed a receding horizon perception-aware local planner for multi-copters, which is able to guide the vehicle to areas with rich visual features and reduce the features' motion blur by reducing the planned trajectory's aggressiveness. We conducted both indoor and outdoor experiments to show the effectiveness of the proposed method in improving the VIO's position estimation accuracy and reducing the VIO's failure rate. The proposed method is capable of running in real time on a small embedded computer onboard the vehicle.

## Acknowledgements

The research in this chapter was partially supported by the AFRI Competitive Grant no. 2020-67021-32855/project accession no. 1024262 from the USDA National Institute of Food and Agriculture. The AFRI Competitive Grant is being administered through AIFS: the AI Institute for Next Generation Food Systems, `https://aifs.ucdavis.edu`. Research was

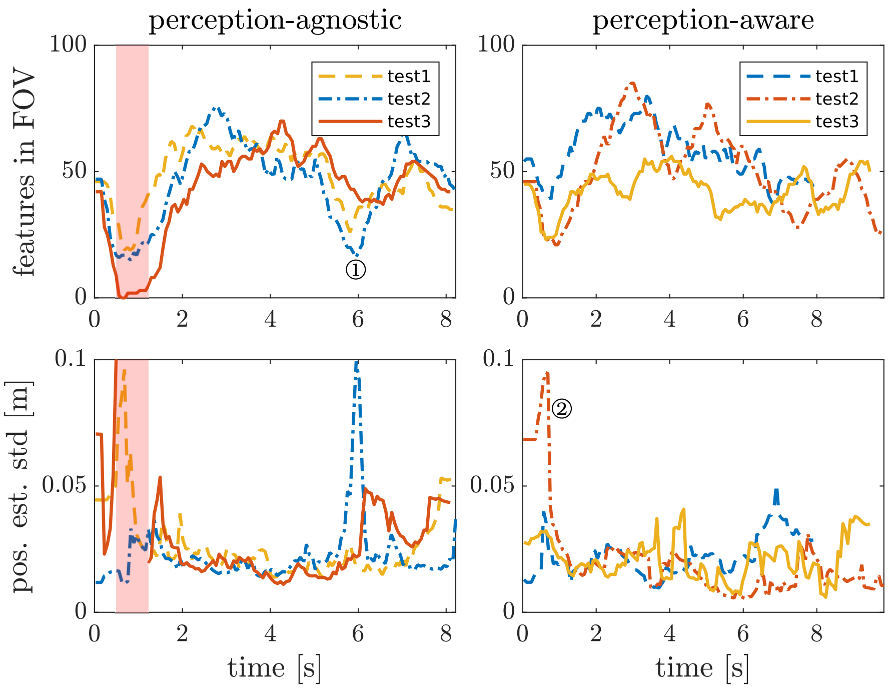


Figure 5.9: Comparison between the perception-agnostic planner (column 1) with the perception-aware planner (column 2) in the outdoor experiments. For each planner, the tests are repeated three times, as shown in lines with different color and line type. The red shaded area marks the time when triangulation fails in test 3 of the perception-agnostic planner.

also partially sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-20-2-0105. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

The authors thank Patrick Geneva of the Robot Perception and Navigation Group, University of Delaware, for his helpful support and suggestions over the usage of the OpenVINS. The experimental testbed at the HiPeRLab is the result of contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

# Chapter 6

# Inertial navigation motion planning strategy

In certain challenging environments, such as inside buildings on fire, the main sensors (e.g. cameras, LiDARs and GPS systems) used for multicopter localization can become unavailable. Direct integration of the inertial navigation sensors (the accelerometer and rate gyroscope), is however unaffected by external disturbances, but the rapid error accumulation quickly makes a naive application of such a strategy feasible only for very short durations. In this chapter, we propose a motion strategy for reducing the inertial navigation state estimation error of multicopters. The proposed strategy breaks a long duration flight into multiple short duration hops between which the vehicle remains stationary on the ground. When the vehicle is stationary, zero-velocity pseudo-measurements are introduced to an extended Kalman Filter to reduce the state estimation error. We perform experiments for closed-loop control of two multicopters for evaluation: one is a standard quadcopter, and the other is a novel tensegrity quadcopter. The mean absolute position estimation error was 3.4% over a total flight distance of 5m in the experiments. The results showed a 80% reduction compared to the standard inertial navigation method without using this strategy. In addition, an outdoor experiment demonstrated that the proposed strategy is able to navigate a quadcopter in real-world environments.

Note that the materials in this chapter is mainly based on the following published paper:

- Xiangyu Wu and Mark W. Mueller. "Using multiple short hops for multicopter navigation with only inertial sensors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 8559–8565

In addition, inertial navigation experimental results with a novel tensegrity vehicle are also included in this chapter, which is from the following paper to be published:

- Jiaming Zha, Xiangyu Wu, Ryan Dimick, and Mark W. Mueller. "A collision-resilient aerial robot design with an icosahedron tensegrity shell".

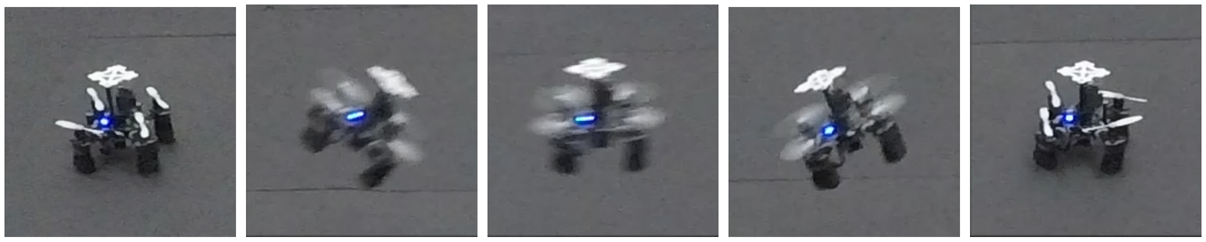


Figure 6.1: Video sequences of a quadcopter doing a short hop flight. The state estimation error of inertial navigation can be reduced significantly by breaking a long time flight into multiple short time hop flights.

## 6.1 Introduction

Reliable and cost effective state estimation methods are critical for the operation of unmanned aerial vehicles (UAV). In laboratories, motion capture systems are often used to obtain very accurate state estimation of multicopters at high frequency [99]. While these systems are ideal for indoor research use, they are usually very expensive, inconvenient to set up and can only cover a small area in the order of several square meters [100]. For operations in open areas, Global Navigation Satellite System (GNSS) is widely used and can achieve localization accuracy of a few meters. In places where GNSS signal is weak or unavailable (e.g. indoor, underground or near tall buildings), radio beacons can be used to setup a localization networks [101] [102]. Such radio based localization systems can be created at a relatively low cost but the reliance on infrastructure makes them less flexible compared to only relying on on-board sensors.

Another category of UAV state estimation methods only relies on sensors on the vehicle itself such as cameras, LiDAR and Inertial Measurement Units (IMUs). One popular method in this category is simultaneous localization and mapping (SLAM) where the measurements from on-board sensors are often fused to build a map of the vehicle's surrounding environment and find the vehicle's location in the map [103] [104] [105]. Not dependent on any particular infrastructure, these methods are easy to deploy. In the recent past there have been significant progress in this area which enables them to transit into real-world applications [105]. On the other hand, these methods often require expensive sensors such as LiDAR and powerful computers and it is challenging to make them work robustly in challenging (e.g. featureless or dusty) environments.

Inertial navigation is a potential solution under these challenging environments since the only sensor it requires is the IMU, which is usually unaffected by environments. Inertial state estimation has many applications in robotics state estimation. For example, [106] used inertial navigation for wheeled robots. By utilizing the information that the wheeled vehicle's lateral and vertical velocities are roughly zero in body frame and using a Convolutional Neural Network (CNN) for the IMU noise estimation, the method was able to achieve a position estimation error comparable to methods of using LiDAR or stereo vision. In addition, [107] [108] used inertial measurements with the aerodynamic modeling of the vehicle for the velocity and attitude estimation of multicopters.

A major challenge of inertial navigation is that consumer-level IMUs have large measurement noise which will result in fast error accumulation [109]. One technique to reduce inertial navigation error is detecting when the tracked object is stationary and add "zero-velocity pseudo measurements" to the state estimator during the stationary period. For example, [110] and [111] discussed the use of this technique to reduce pedestrain tracking error.

In this chapter, we propose a strategy for multicopter inertial navigation with only the accelerometer and rate gyroscope as sensors. In our proposed strategy, the multicopter moves by taking a series of short duration flights instead of a single long duration flight. Between the short flights, the vehicle remains stationary on the ground and we introduce zero-velocity pseudo measurements to an extended Kalman Filter (EKF) to reduce the state estimation error. Analytical analysis of state estimation error of this method is given and experiments were done to evaluate its effectiveness.

The proposed method is especially helpful in challenging environments, because 1) it does not require any infrastructure and can be easily deployed; 2) rapid technological advancement of Micro-electromechanical system (MEMS) based IMUs have made them cheap, small and lightweight to be widely used on small multicopters; 3) it is unaffected by GPS-denied, limited visibility (e.g. smoky) or featureless environments. For example, when a multicopter is used to help firefighters to get information about a building on fire, this method can be used to navigate the vehicle to go through sections with dense smoke when other sensors (e.g. cameras and LiDARs) used for navigation are temporarily unavailable.

## 6.2 Multicopter modelling

In this section, we define the reference frames, briefly introduce the dynamic model of a multicopter, and analyze the accelerometer and rate gyroscope error characteristics.

### Multicopter dynamics

As shown in Fig. 6.2 an inertial frame $I$ attached to the ground and a body frame $B$ attached to the Center of Mass (COM) of the multicopter, are defined. The multicopter is defined as a rigid body with six degrees of freedom: three degrees of freedom from the linear translation $\boldsymbol{p}$ along the three axes of the inertial frame and three degrees of freedom from the three-axis rotation from the body frame to the inertial frame, described by an orthogonal rotation matrix $\boldsymbol{R}$. Denote the thrust produced by each propeller as $\boldsymbol{f}_i$, expressed in the vehicle's body frame. With linear velocity $\boldsymbol{v}$, linear acceleration $\boldsymbol{a}$ and the gravity acceleration $\boldsymbol{g}$, all expressed in the inertial frame, the translational dynamics of the vehicle is expressed as

$$\frac{d}{dt}\boldsymbol{p} = \boldsymbol{v} \tag{6.1}$$

$$\frac{d}{dt}\boldsymbol{v} = \boldsymbol{a} \tag{6.2}$$

$$m\boldsymbol{a} = m\boldsymbol{g} + \boldsymbol{R}\sum \boldsymbol{f}_i \tag{6.3}$$

Denote the angular velocity of the vehicle as $\boldsymbol{\omega} = (w_1, w_2, w_3)$ and torque produced by each propeller as $\boldsymbol{\tau}_i$, both expressed in the vehicle's body frame. The rotational dynamics of the vehicle is expressed as

$$\frac{d}{dt}\boldsymbol{R} = \boldsymbol{R}\, S(\boldsymbol{\omega}) \tag{6.4}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\,\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \sum \boldsymbol{\tau}_i \tag{6.5}$$

where $S(\boldsymbol{\omega})$ is the skew-symmetric matrix form of the vector cross product such that

$$S(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{6.6}$$

A detailed description of multicopter dynamics can be found in e.g. [112] [1].

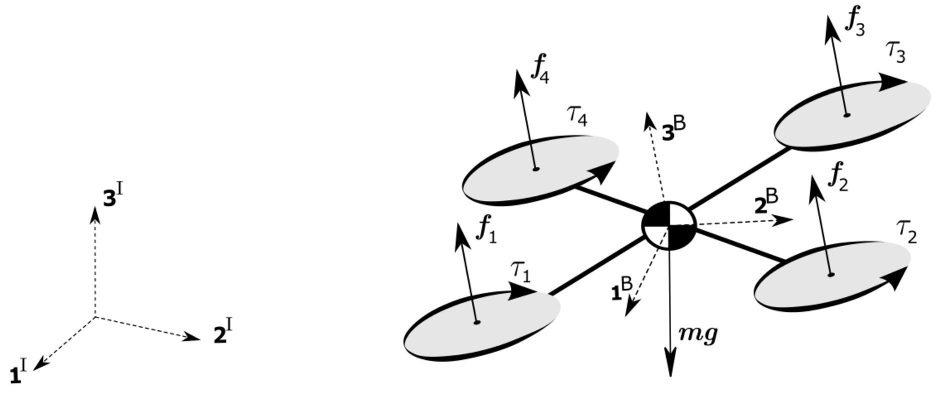


Figure 6.2: Definition of reference frames. $I$ represents the inertial reference frame and $B$ represents the vehicle's body frame.

### Sensor error characteristics

In this chapter we focus on MEMS IMUs consisting of a gyroscopes and an accelerometer, which are widely used on micro multicopters because of their small size and weight, low power consumption and low cost. The rate gyroscope measures angular velocity and the accelerometer measures proper acceleration (acceleration relative to a free-fall) of the vehicle, both measurements are expressed in the body frame $B$.

Measurement errors of MEMS IMUs usually include the following types [109]:

1. Bias: the offset of IMU measurements from the true values, which can be compensated by simply subtracting out from IMU outputs.
2. Thermo-mechanical white noise
3. Bias instability: the change in IMU bias.
4. Scale factor error and nonlinearity.

Among these four types of error, the thermo-mechanical white noise and uncorrected bias and scale factor error are usually the most significant errors [109]. In the proposed inertial navigation method, the bias and scale error are corrected by IMU calibration. The sensors are modeled as

$$\boldsymbol{\alpha} = \boldsymbol{R}^{-1}(\boldsymbol{a} - \boldsymbol{g}) + \boldsymbol{n}_\alpha \tag{6.7}$$

$$\boldsymbol{\gamma} = \boldsymbol{\omega} + \boldsymbol{n}_\gamma \tag{6.8}$$

where $\boldsymbol{\alpha}$ is the measurement of accelerometer and $\boldsymbol{n}_\alpha$ is accelerometer's measurement noise. Similarly, $\boldsymbol{\gamma}$ is the measurement of the rate gyroscope, and $\boldsymbol{n}_\gamma$ is rate gyroscope's noise. The variance of the noises are $\sigma_\alpha^2 \boldsymbol{I}$ and $\sigma_\gamma^2 \boldsymbol{I}$ for the acceleromenter and rate gyroscope respectively, both noises are assumed to be isotropic.

## 6.3 State estimation

An extended Kalman Filter (EKF) is used for the state estimation of the vehicle according to the method proposed in [113]. The estimator's state vector $\hat{\boldsymbol{\xi}}$ consists of 9 elements:

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} \hat{\boldsymbol{p}} \\ \hat{\boldsymbol{v}} \\ \hat{\boldsymbol{\delta}} \end{bmatrix} \tag{6.9}$$

where $\hat{\boldsymbol{p}}$ is vehicle's estimated position, $\hat{\boldsymbol{v}}$ is vehicle's estimated velocity and $\hat{\boldsymbol{\delta}}$ is estimated three-dimensional attitude error with respect to the reference orientation $\hat{\boldsymbol{R}}_{ref}$. The estimated orientation of the vehicle is represented by

$$\hat{\boldsymbol{R}} = \hat{\boldsymbol{R}}_{\text{ref}}(t) \exp\left( S\left(\hat{\boldsymbol{\delta}}\right)\right) \tag{6.10}$$

where $\exp(\cdot)$ is matrix exponential so that $\exp\left( S\left(\hat{\boldsymbol{\delta}}\right)\right)$ represents rotation matrix.

## Prediction

During the prediction step, the estimator does not use dynamics equations of the vehicle, but uses measurements from the accelerometer and rate gyroscope instead. Thus, it does not require knowing any dynamics parameters of the vehicle. The prediction step follows the following difference equations

$$\begin{aligned}\hat{\boldsymbol{\xi}}_p(t+\Delta t) &= \hat{\boldsymbol{\xi}}(t) + \begin{bmatrix} \frac{d}{dt}\hat{\boldsymbol{p}}(t) \\ \frac{d}{dt}\hat{\boldsymbol{v}}(t) \\ \frac{d}{dt}\hat{\boldsymbol{\delta}}(t) \end{bmatrix} \Delta t \\ &= \begin{bmatrix} \hat{\boldsymbol{p}}(t) + \hat{\boldsymbol{v}}(t)\Delta t \\ \hat{\boldsymbol{v}} + \left( \left( \hat{\boldsymbol{R}}_{\text{ref}}(t) \exp\left( S\left( \hat{\boldsymbol{\delta}}(t) \right) \right) \right)^{-1} \boldsymbol{\alpha}(t) + \boldsymbol{g} \right) \Delta t \\ \boldsymbol{\delta}(t) + \left( \boldsymbol{\gamma}(t) - \frac{1}{2} S(\boldsymbol{\gamma}(t))\, \boldsymbol{\delta}(t) \right) \Delta t \end{bmatrix}\end{aligned} \tag{6.11}$$

The estimated variance is given by

$$\boldsymbol{P}_{\xi\xi,p}(t+\Delta t) = \boldsymbol{A}(t)\boldsymbol{P}_{\xi\xi}(t)\boldsymbol{A}(t)^T + \boldsymbol{Q} \tag{6.12}$$

where

$$\begin{aligned}\boldsymbol{A}(t) &= \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I}\Delta t & 0 \\ 0 & \boldsymbol{I} & S\left( \hat{\boldsymbol{R}}_{\text{ref}}(t)^{-1} \boldsymbol{\alpha}(t) \right) \Delta t \\ 0 & 0 & \boldsymbol{I} - \frac{1}{2} S(\boldsymbol{\gamma}(t))\, \Delta t \end{bmatrix} \\ \boldsymbol{Q} &= \operatorname{diag}\left[ 0, \sigma_\alpha^2 \boldsymbol{I}, \sigma_\gamma^2 \boldsymbol{I} \right]\end{aligned}$$

The attitude error $\boldsymbol{\delta}$ is then set to zero after each prediction, by updating the reference attitude $\boldsymbol{R}_{\mathbf{ref}}$ and covariance matrix. The details of this update can be found in [112].

## Zero-velocity update

When the vehicle stays still on the ground, the knowledge that the vehicle is not moving can be utilized to improve state estimation performance. The proposed zero-velocity detector is based on rate gyroscope only for computational simplicity, since [110] has shown that using both measurements from the accelerometer and rate gyroscope only gives marginal performance improvement compared to using rate gyroscope's measurements only. The zero velocity detector has two tuning parameters, $N_{\text{threshold}}$ and $\gamma_{\text{threshold}}$. If the magnitude of the rate gyroscope's measurement is below $\gamma_{\text{threshold}}$ for more than $N_{\text{threshold}}$ continuous time steps, the vehicle is considered stationary and it is considered to be moving otherwise. For the hardware we use, $N_{\text{threshold}}$ was tuned to be 20 and $\gamma_{\text{threshold}}$ was tuned to be 0.2 rad/s. A demonstration of the zero velocity detector is shown in Fig. 6.3.

When the vehicle is detected to be stationary, zero-velocity updates as pseudo measurements are introduced to the state estimator. The observation matrix H is

$$\boldsymbol{H} = \left[ 0, \boldsymbol{I}_{3\times 3}, 0 \right]$$

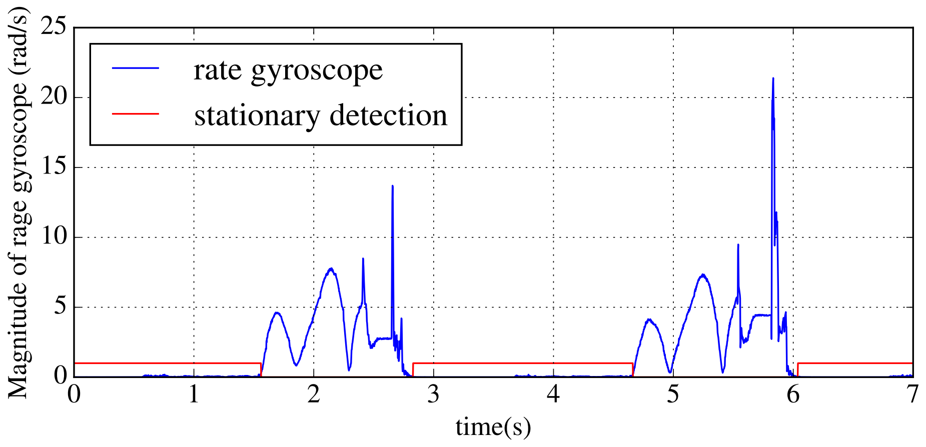


Figure 6.3: A demonstration of the implemented zero velocity detector based on the magnitude of rate gyroscope measurements. The value of zero velocity detection state is either 1 (stationary) or 0 (moving). The vehicle is "hopping" from 1.55s to 2.75s and from 4.7s to 6.0s. It remains stationary on the ground for the rest of the time.

For notational convenience, we set $t \leftarrow t + \Delta t$. Follow the standard EKF formalism and we can get

$$\boldsymbol{K}(t) = \boldsymbol{P}_{\xi\xi,p_+}(t)\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{\xi\xi,p_+}(t)\boldsymbol{H}^T\right)^{-1} \tag{6.13}$$

$$\hat{\boldsymbol{\xi}}_m(t) = \hat{\boldsymbol{\xi}}_{p_+}(t) + \boldsymbol{K}(t)\left(-\hat{\boldsymbol{v}}_{p_+}(t)\right) \tag{6.14}$$

$$\boldsymbol{P}_{\xi\xi,m}(t) = \left(\boldsymbol{I} - \boldsymbol{K}(t)\boldsymbol{H}\right)\boldsymbol{P}_{\xi\xi,p_+} \tag{6.15}$$

$$\hat{\boldsymbol{R}}_{\text{ref},m}(t) = \hat{\boldsymbol{R}}_{\text{ref},p_+}(t) \tag{6.16}$$

Note that the zero-velocity pseudo measurements has zero variance. The $\hat{\boldsymbol{\delta}}_m(t)$ is then reset to zero by updating the reference attitude $\boldsymbol{R}_{\text{ref}}$ and covariance matrix. The details of this update can be found in [112].

## 6.4 Multicopter inertial navigation

In this section, we provide a simplified analysis of the state estimation error of inertial navigation and based on the analysis propose a special flight motion to the state estimation error.

### Error analysis

State estimation by directly integrating the measurements from IMU would cause the estimation error to grow rapidly. To illustrate rapid growth of error, a simplified analysis of the position estimation error on one direction is given below. Linearize the attitude at hovering, where the proper acceleration is equal to $-\boldsymbol{g}$. We define $\hat{\boldsymbol{\xi}} = \left[p, v, \theta\right]^\top$, where $p$, $v$ and $\theta$ represent position, velocity and attitude respectively, from (6.1), (6.2), (6.7) and (6.8), we can get

$$\frac{d}{dt}\hat{\boldsymbol{\xi}}(t) = \boldsymbol{A}\hat{\boldsymbol{\xi}}(t) + \begin{bmatrix} 0 \\ \sigma_\alpha \\ \sigma_\gamma \end{bmatrix} \tag{6.17}$$

$$\frac{d}{dt}\boldsymbol{P}(t) = \boldsymbol{A}\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}^\top + \boldsymbol{Q} \tag{6.18}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & g \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{Q} = \operatorname{diag}\left[0, \sigma_\alpha^2, \sigma_\gamma^2\right].$$

We define the initial variance of the states to be $\boldsymbol{P}(0) = \operatorname{diag}[P_{pp}(0), P_{vv}(0), P_{\theta\theta}(0)]$. The variance of the states at time $t$ are given by

$$P_{\theta\theta}(t) = \sigma_\gamma^2 t + P_{\theta\theta}(0) \tag{6.19}$$

$$P_{vv}(t) = \frac{\sigma_\gamma^2 g^2}{3}t^3 + g^2 P_{\theta\theta}(0)t^2 + \sigma_\alpha^2 t + P_{vv}(0) \tag{6.20}$$

$$P_{pp}(t) = \frac{\sigma_\gamma^2 g^2}{20}t^5 + \frac{g^2 P_{\theta\theta}(0)}{4}t^4 + \frac{\sigma_\alpha^2}{3}t^3 + P_{vv}(0)t^2 + P_{pp}(0) \tag{6.21}$$

which shows that after time $t$ the additive noise from the IMU causes the variance of position grow on the order of $t^5$.

## Motion planning

The simplified error analysis in section 6.4 points towards a motion planning strategy to reduce state estimation variance: planning a path with many short duration "hops". Between the hops the vehicle remains stationary on the ground and we can partially reset the state uncertainty. A single flight of duration $t$ is broken into $N$ hops of time $t/N$. After each hop the vehicle stays stationary and the estimation variance of velocity estimation is set to zero. So the velocity estimation variance at the beginning of each hop flight is zero. From (6.19) - (6.21), the estimation variance of $\theta$ at the end of short flight $i$ ($i \in \{1, 2, ..., N\}$) is given by

$$P_{\theta\theta}\left(\frac{i}{N}t\right) = P_{\theta\theta}(0) + \sigma_\gamma^2 \frac{i}{N}t \tag{6.22}$$

and estimation variance of position has the following relationship

$$P_{pp}\left(\frac{i+1}{N}t\right) = \frac{\sigma_\gamma^2 g^2 t^5}{20N^5} + \frac{g^2 P_{\theta\theta}(\frac{i}{N}t)t^4}{4N^4} + \frac{\sigma_\alpha^2 t^3}{3N^3} + P_{pp}(\frac{i}{N}t) \quad (i{\geq}1) \tag{6.23}$$

$$P_{pp}\left(\frac{t}{N}\right) = \frac{\sigma_\gamma^2 g^2 t^5}{20N^5} + \frac{g^2 P_{\theta\theta}(0)t^4}{4N^4} + \frac{\sigma_\alpha^2 t^3}{3N^3} + \frac{P_{vv}(0)t^2}{N^2} + P_{pp}(0) \quad (i=0) \tag{6.24}$$

From (6.22) - (6.24) the position estimation variance after $N$ short motions is

$$P_{pp}(t) = \frac{(5N-3)\sigma_\gamma^2 g^2 t^5}{40N^4} + \frac{g^2 P_{\theta\theta}(0)t^4}{4N^3} + \frac{\sigma_\alpha^2 t^3}{3N^2} + \frac{P_{vv}(0)t^2}{N^2} + P_{pp}(0) \tag{6.25}$$

The effect of this motion planning strategy can be seen by comparing (6.25) to (6.21). Assume, for example, the initial condition of the system is perfectly known, and the system has a perfect rate gyroscope, such that the state estimation uncertainty only comes from the the accelerometer, the final position estimation variance is reduced by a factor of $N^{-2}$ when $t$ is large.

In Section 6.3, we use an extended Kalman Filter for state estimation. The zero-velocity pseudo measurement would provide a even greater reduction in position estimation variance compared to the error analysis in this section because of the correlation between velocity and position as well as the correlation between velocity and attitude.

For trajectory planning of each hop, we use the method proposed in [98], a trajectory generation method which minimizes the jerk (third derivative of position) of the multicopter given the initial and desired final states. The solution of minimum-jerk trajectory is provided in closed form and is computationally inexpensive, which makes it suitable to be implemented on embedded flight controllers. In addition, the method verifies if the planned trajectory satisfies the vehicle's actuation constraints (e.g. maximum thrust of the motor) and does not collide with known planar obstacles (e.g. the ground).

The vehicle's initial velocity and acceleration are zero because of the stationary state between the hops. The final velocity of the hop trajectory is zero such that the vehicle stops moving after each hop. Between each hop, the vehicle remains stationary on the ground, and the zero-velocity pseudo measurements are introduced to reduce the estimation variance of the state estimator. Although a shorter hop duration is helpful to reduce state estimation error, too aggressive trajectories will make the trajectory tracking difficult. As a result, the trade-off between the aggressiveness of the hop trajectory and trajectory tracking should be considered when choosing the hop time. The height of the hop trajectory can be adjusted by changing the final acceleration on the vertical direction.

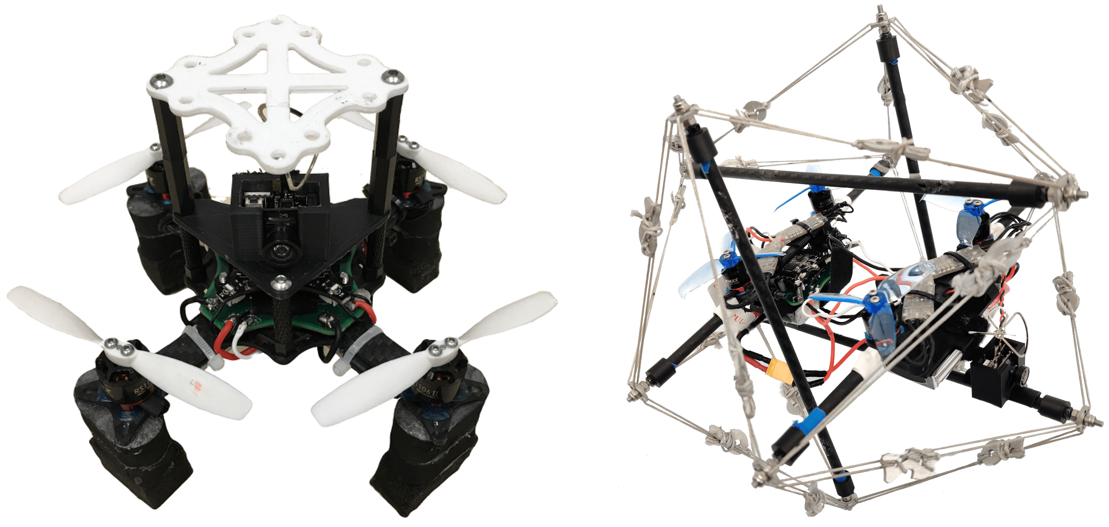
## 6.5 Experimental results

### Experimental setup

Experiments were conducted to evaluate the performance of the proposed inertial navigation method. In the experiments, two custom-built quadcopters, as shown in Fig. 6.4 were used.

The first quadcopter is a normal quadcopter (Fig. 6.4(a)) that weighs 165 grams. The distance between the hubs of two diagnal motors is 117.6 mm and the propeller is 76.2mm in diameter. A small analog camera is installed on the vehicle for video streaming. The second quadcopter is a special tensegrity quadcopter (Fig. 6.4(b)) that weighs 315 grams. The length of each rod in the icosahedron tensegrity shell is 200 mm. The tensegrity drone design was proposed in [114]. It has high collision tolerance, enabling it to work in challenging environments with hard-to-avoid obstacles. Details of unique static and dynamic features of tensegrity structures can be found at e.g. [115, 116].

For both of the vehicles, a Crazyflie 2.0 [36] running a modified version of the PX4 firmware was used as a low-level flight controller for the quadcopter. It is equipped with a consumer-level MEMS IMU (InvenSense MPU-9250) with a measurement frequency of 500Hz. The trajectory tracking flight controller and the proposed inertial state estimator ran on this micro-controller at 500Hz.



(a) A normal quadcopter. (b) A tensegrity quadcopter.

Figure 6.4: The quadcopters used in the experiments.

The indoor experiments were done in a flight space of size $7 \times 6 \times 5$m. A commercial motion capture system, which provides high-accuracy, high-rate state information, was used during the experiments to provide ground truth of the vehicle's states. In addition, we show the ability of the method to navigate the tensegrity quadcopter in a complex forest environment.

## Performance evaluation

### Tests with a normal quadcopter

The proposed inertial state estimator's performance was quantified by comparing with state estimation from the motion capture system. Note that during the experiments, the motion capture system is only used for ground truth, not for control of the vehicle. In the experiments, the vehicle was commanded to fly in a constant direction for 5 meters in 5 hops, each hop had length of 1 meter and took 1 second. There was a two-second interval between two hops, when the vehicle is stationary on the ground. The state estimator introduced in Section 6.3 was used to estimate the state of the vehicle and a cascaded PD controller was used for trajectory tracking.

The experiment was repeated eight times. The state estimation error of the proposed inertial state estimator is shown in Fig. 6.5. At the end of the flight, the root mean square error (RMSE) for position estimation was 0.13m, 0.12m and 0.03m for the downrange direction (flight direction), crossrange direction (perpendicular to the flight direction and in the horizontal plane) and vertical direction, respectively. The mean absolute position estimation error was 0.17m, which was 3.4% of the total flight distance. The trajectory of the vehicle (measured by the motion capture system), is compared with the reference trajectory in Fig. 6.6. At the end of the flight, the RMSE of position tracking was 0.19m and 0.06m for the crossrange and downrange direction. The position tracking error was zero for the vertical direction because the floor in the flight space is horizontal and flat. The mean absolute tracking error was 0.16m, which was 3.2% of the total flight distance.

For comparison, experiments with a long duration flight instead of multiple short-duration hops are conducted to compare with the proposed strategy. In the experiments, the vehicle was commanded to fly in a constant direction for 5 meters in 5 seconds. The motion capture system was used during the flight for state estimation for control because inertial navigation for 5 seconds would give a large state estimation error and can crash the vehicle. After the experiments, we run the inertial navigation off-board using the collected IMU data. We then compare the position estimation from inertial navigation with position estimation from the motion capture system, which is used as ground truth. The experiment was repeated eight times and the state estimation error is shown in Fig. 6.7. At the end of the flight, the root mean square error (RMSE) for position estimation was 0.85m, 0.56m and 0.25m for the downrange, crossrange and vertical direction, respectively. The mean absolute position estimation error was 0.88m, which is 17.6% of the total flight distance, and is 4.2 times larger than the proposed strategy. The position estimation error of these two methods are compared in table 6.1.

In addition to the experiments in the flight space, an additional experiment was conducted where the vehicle flew a longer distance around a corner. The experiment environment is shown in Fig. 6.8. This experiment demonstrates the ability of the proposed inertial navigation strategy to navigate a multicopter in real-world environment. The vehicle first flew forward for 6m in 6 hops and then made a left turn and flew 4m in 4 hops. Each hop

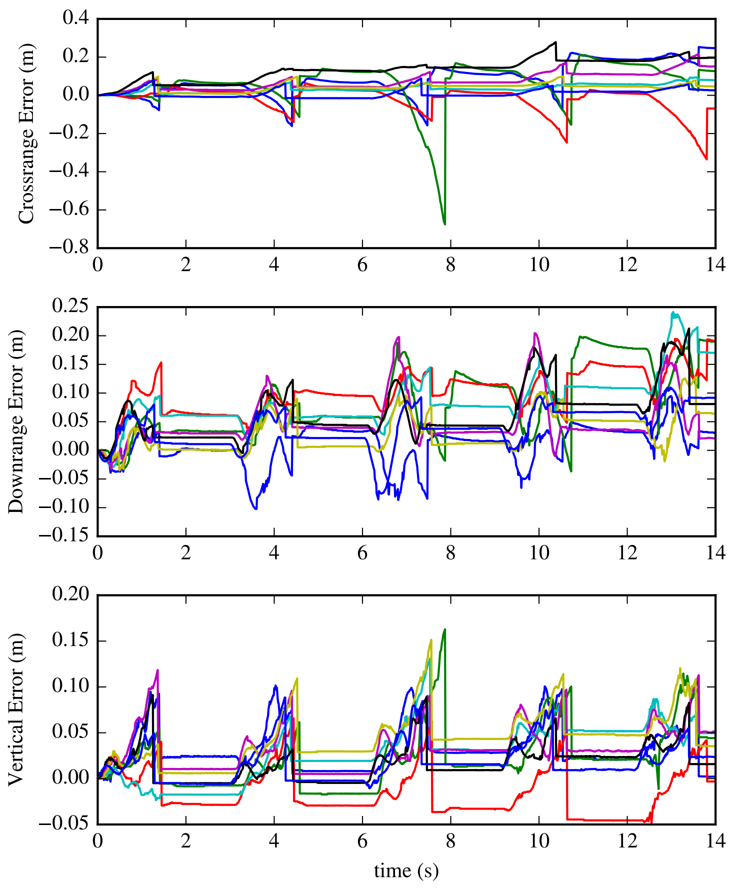


Figure 6.5: Position estimation of the vehicle, using only the rate gyroscope and accelerometer. No knowledge of the environment, except that the floor is not moving, is assumed. The effect of zero velocity measurement update could be seen at around 1.5s, 4.5s, 7.5s, 10.5s and 13.5s for reducing the estimation error.

Table 6.1: Inertial navigation position estimation error comparison between the proposed strategy and long-distance flight (with the normal quadcopter)

| position estimation error | proposed strategy | long-distance flight |
|---|---|---|
| downrange RMSE | 0.13m | 0.85m |
| crossrange RMSE | 0.12m | 0.56m |
| vertical RMSE | 0.03m | 0.25m |
| mean absolute error | 0.17m | 0.88m |

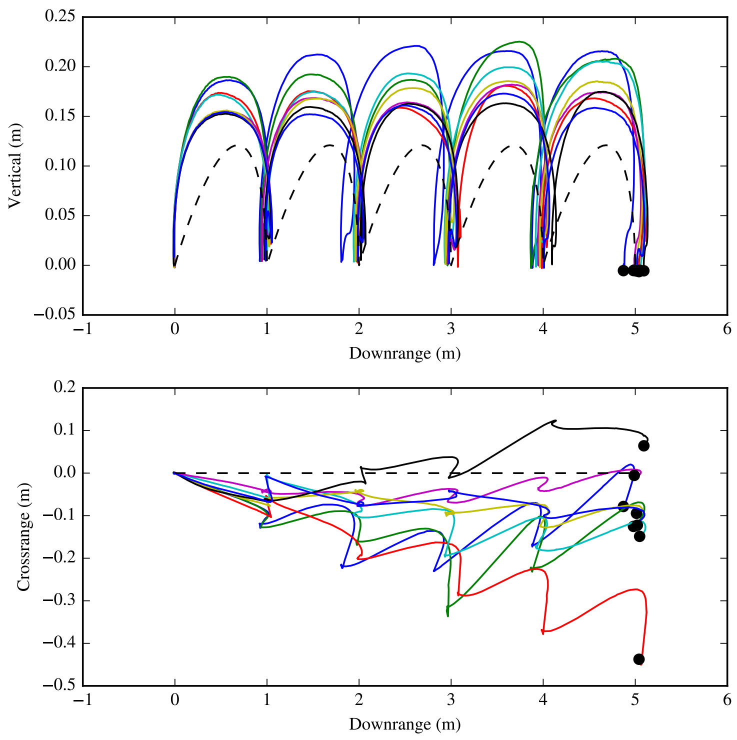


Figure 6.6: Closed-loop control of the vehicle using the proposed inertial navigation estimator. Trajectories of 8 separate flights are shown in solid lines with different color and the final position of each flight is marked by a solid circle. The reference trajectory is shown as a black dashed line.

was 1 second. The proposed strategy successfully navigates the vehicle and final position tracking error was about 0.3m, which is about 3% of the total flight distance, a similar result as before.

### Tests with the tensegrity quadcopter

In addition to the experiments with the normal quadcopter, we also tested the proposed inertial navigation strategy with the tensegrity quadcopter, to further validate its effectiveness.

With the proposed strategy, the tensegrity vehicle is able to traverse unknown environments and continue operation after a collision. If the vehicle encounters a collision, indicated by the norm of the accelerometer exceeding a given threshold, a recovery controller will be triggered and vehicle will mark the position where the collision takes place. The recovery controller increases the angular velocity control gains and commands a trajectory with a small constant negative velocity along the z-axis of the world frame, attempting to stabilize

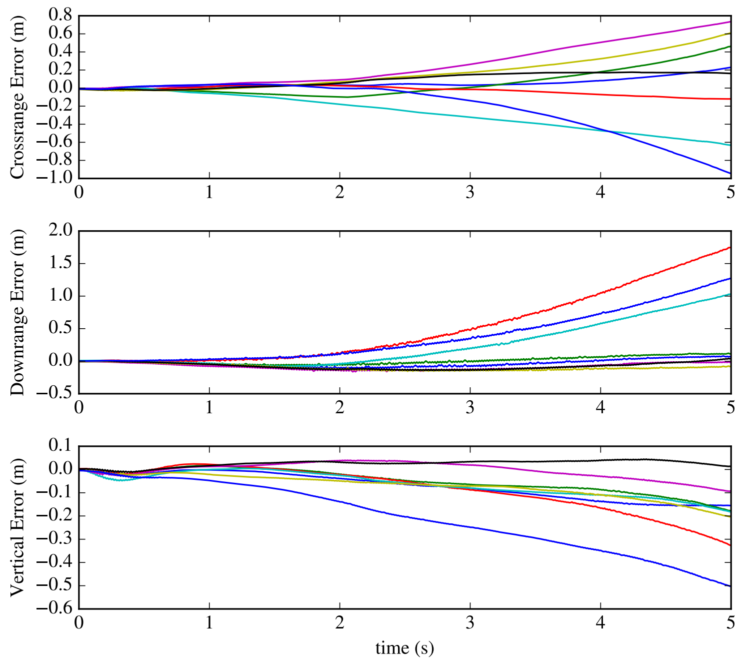


Figure 6.7: Position estimation error of the vehicle for a single long duration flight without using the proposed strategy.

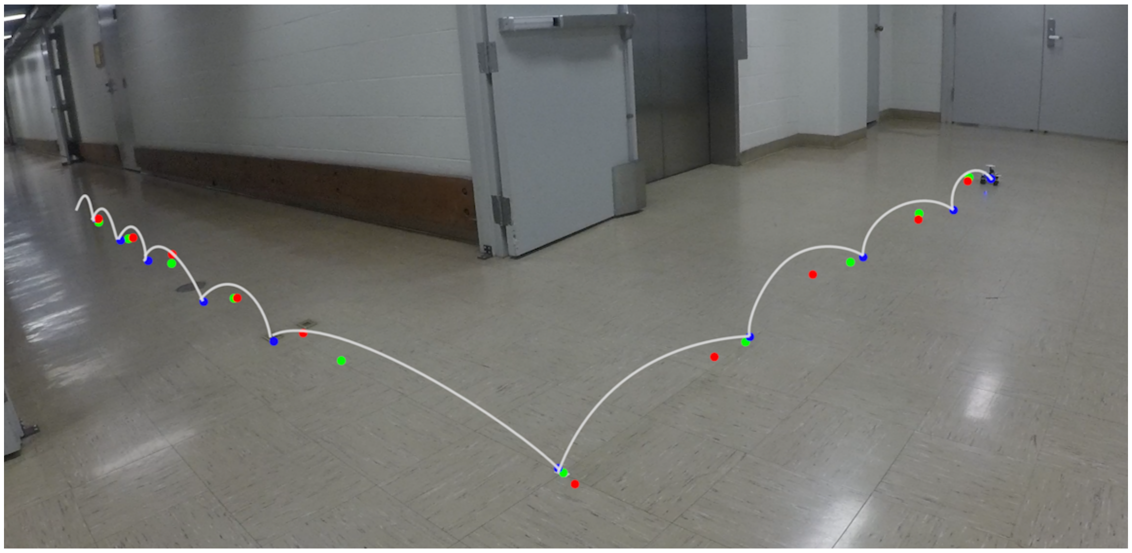


Figure 6.8: The vehicle flew 10m in 10 hops around a corner. The green dots mark the target positions, the blue dots mark the actual position of the vehicle after each hop, and the red dots mark the estimated positions. The white curves represents the actual flight path.

and land the vehicle softly in order to reduce inertial estimation error from large impacts. Once the vehicle has landed following a collision, it will attempt to hop around the obstacle it just collided with.

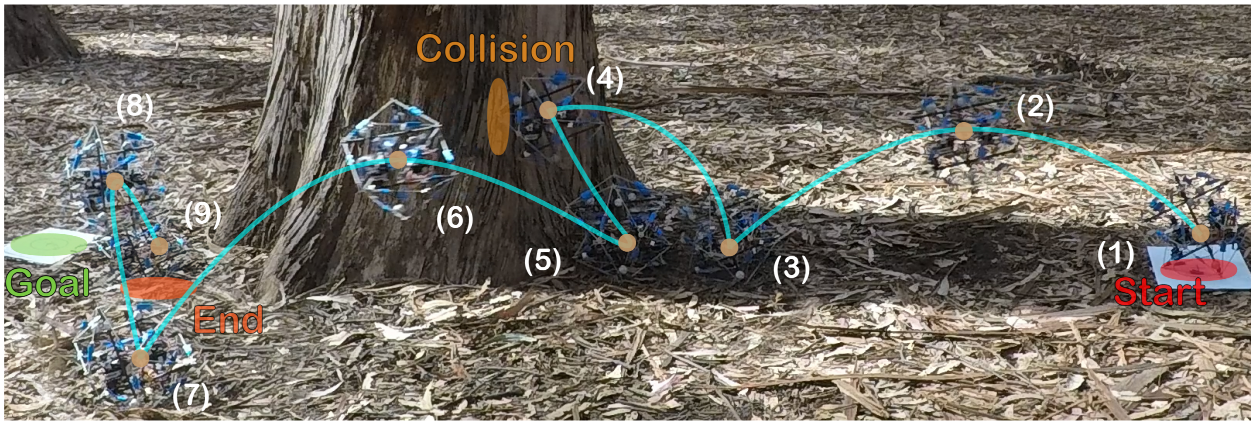


Figure 6.9: Image sequence of the tensegrity collision resilient vehicle using inertial navigation strategy to navigate in a previously unknown forest environment. The cyan curve marks the movement of the vehicle. The vehicle is ordered to move from the start point on the right side of the figure to the goal point on the left side of the figure. A tree obstacle exists in between the two points. The vehicle successfully survives a collision with the tree and arrives at an end point close to the goal. The distance between the goal point and the end point is 0.25m.

A composite image of a test in the forest is shown in Fig. 6.9. In this experiment, the vehicle is ordered to move in a given direction for 3m and there is a tree between the start position the goal. During the experiment, the vehicles collides with the tree during its second hop. It survives the collision, marks the position of the obstacle, hops sideways to avoid the obstacle, then continues to hop toward the final goal position.

## 6.6 Conclusion

In this work, an inertial navigation strategy for multicopters was proposed. The proposed method is based only on measurements from the on-board accelerometer and rate gyroscope and is especially suitable for challenging environments where other sensors are unavailable. An error analysis of the state estimation error of inertial navigation was introduced and based on this analysis a motion planning method of breaking a long time flight into multiple short time flight steps was proposed to reduce the estimation error.

Indoor experiments were repeated multiple times to evaluate the performance of this state estimation method, using a standard quadcopter equipped with a consumer-level IMU. The state estimator was used for closed-loop control of the quadcopter. The experiments showed that the mean absolute position estimation error of the proposed state estimator at the end of the flight was 0.17m for translation of 5m, which was 3.4% of the total distance. In addition, outdoor experiments with a tensegrity quadcopter demonstrate the vehicle's ability to navigate a quadcopter in real-world environments.

## Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge. The experimental testbed at the HiPeRLab is the result of contributions of many people, a full list of which can be found at `hiperlab.berkeley.edu/members/`.

# Chapter 7

# Conclusion and future work

## 7.1 Conclusion

This thesis presents several motion planning methods to mitigate the challenging problems of autonomous and energy-efficient flights of multicopters in complex environments.

In Chapter 2 and Chapter 3, we propose two model-free methods that can find the optimal flight speed and sideslip of quadcopters to achieve the longest flight time or range. Based on the extremum seeking controller, they are computationally efficient to run on low-cost embedded computers and can adapt to unknown payloads and wind disturbances. The proposed methods are beneficial to mitigate the limited flight time and range problem, which is common for quadcopters.

Next, in Chapter 4 and Chapter 5, we propose two sampling-based trajectory planners for the fast collision avoidance flight of quadcopters in cluttered environments. Visual inertial navigation is used for the state estimation of the vehicle, and a depth camera is used to sense obstacles in the environment. While in the first planner only fast flight and obstacle avoidance is considered, in the second planner we take the state-estimation quality from the VIO into account. The proposed perception-aware planner is able to guide the vehicle to areas with rich visual features and avoid overly aggressive flight. This is beneficial to improve the VIO's accuracy and reduce its divergence rate, improving the reliability of autonomous flight.

Finally, in Chapter 6, we propose a motion planning strategy for the inertial navigation of quadcopters when other state estimation methods (e.g. VIO, GPS) fail. Based only on the accelerometer and rate gyroscope, inertial navigation does not require any infrastructure in the environment and is usually unaffected by the environments. Our proposed motion planning strategy breaks a long-duration flight into multiple short-duration "hopping trajectories", and introduces zero-velocity update when the vehicle is stationary on the ground between two hops. This strategy dramatically reduces the state estimation uncertainty of the inertial navigation and can be used for the closed-loop control of the vehicle.

Extensive indoor and outdoor experiments are conducted to validation the effectiveness of the motion planning methods proposed in this thesis.

## 7.2 Future work

Regarding the challenging problems of autonomous flight of quadcopters in complex environments, there are several interesting topics for future work.

Related to the energy-efficient flight of quadcopters, Bayesian optimization or reinforcement learning methods could be utilized to search for the energy-efficient flight time and speed. These methods may achieve a faster convergence speed than the extremum seeking controller based methods proposed in Chapter 2 and Chapter 3. In addition, a deeper investigation into the aerodynamics and power consumption modeling of quadcopters will be helpful to predict the optimal flight speed and sideslip, as well as the planning of the energy-optimal trajectory in windy conditions. Furthermore, there are many recent designs of tilt-wing UAVs, which can take off and land vertically like a multicopter and switch to fixed-wing configuration for long-distance flight. These quadcopter-like vehicles are able to fly longer distances more efficiently than conventional quadcopters by taking advantage of their aerodynamic properties. Planning of energy-efficient trajectories for this kind of novel vehicles can further increase their flight range.

Related to the fast autonomous flight of quadcopters in complex environments, only static environments are considered for the methods proposed in Chapter 4 and Chapter 5. Taking moving objects into account would be a natural next step and is beneficial for the vehicle's safety. The quadcopter should have the ability to detect moving objects, predict their motion, and plan collision-free trajectories accordingly. In addition, a global planner taking into account of semantic information could be added to work together with the proposed local planners, to help avoid potentially dangerous areas (e.g. a crowd of people) and areas with poor VIO features (e.g. water surface).

The vehicle will crash when the state estimator fails and cause danger to people and properties nearby. In Chapter 6 we proposed an inertial navigation strategy using only the IMU as a backup state estimation plan. The method is helpful for indoor flight when the vehicle is close to the ground, but it cannot be used when the vehicle flies at high altitude outdoors. Using different and redundant sensors (e.g. GPS, range finder, camera, and lidar) on the vehicle and isolating the sensors when they fail is helpful for improving state estimation's robustness in these operations. Novel mechanical designs such as adding protective shells or parachutes to the vehicle are also helpful in reducing damage in case of crash.

In summary, there are still numerous opportunities for improving the autonomy of quadcopters in complex environments, and this thesis only mentions several aspects of them. There are also likely many new ideas to explore for the application, design, motion planning, and state estimation of quadcopters, which will help them become more widely used in our life.

# Bibliography

[1] Robert Mahony, Vijay Kumar, and Peter Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robotics Automation Magazine* 19.3 (2012), pp. 20–32.

[2] H. Kang et al. "FlyCam: Multitouch Gesture Controlled Drone Gimbal Photography". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3717–3724.

[3] Yunpeng Wu et al. "A UAV-Based Visual Inspection Method for Rail Surface Defects". In: *Applied Sciences* 8.7 (2018).

[4] Andrea Tagliabue et al. "Robust collaborative object transportation using multiple MAVs". In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1020–1044.

[5] Andrea Tagliabue, Xiangyu Wu, and Mark W. Mueller. "Model-free Online Motion Adaptation for Optimal Range and Endurance of Multicopters". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5650–5656.

[6] X. Wu and M. W. Mueller. "In-flight range optimization of multicopters using multivariable extremum seeking with adaptive step size". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1545–1550.

[7] Xiangyu Wu et al. "Model-free online motion adaptation for energy efficient flights of multicopters". In: *arXiv preprint arXiv:2108.03807* (2021).

[8] Xiangyu Wu et al. "Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry". In: *arXiv preprint arXiv:2204.03134* (2022).

[9] Xiangyu Wu and Mark W. Mueller. "Using multiple short hops for multicopter navigation with only inertial sensors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 8559–8565.

[10] R. Bähnemann et al. "A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects". In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 123–128.

[11] Kelly Steich et al. "Tree cavity inspection using aerial robots". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 4856–4862.

[12] Andrea Tagliabue et al. “Collaborative transportation using MAVs via passive force control”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 5766–5773.

[13] *Kitty Hawk*. `https://kittyhawk.aero/`. (Accessed on 08/27/2018).

[14] Andrea Tagliabue et al. “Shapeshifter: A Multi-Agent, Multi-Modal Robotic Platform for Exploration of Titan”. In: *2020 IEEE Aerospace Conference*. 2020, pp. 1–13.

[15] RD Lorenz et al. “Dragonfly: A Rotorcraft Lander Concept for Scientific Exploration at Titan”. In: *Johns Hopkins APL Technical Digest* (2018).

[16] Konstantinos Karydis and Vijay Kumar. “Energetics in robotic flight at small scales”. In: *Interface focus* 7.1 (2017), p. 20160088.

[17] Sebastian Verling and Julian Zilly. “Modeling and Control of a VTOL Glider”. In: *Bachelor Thesis, Autonomous Systems Lab, ETH Zurich* (2013).

[18] *Uber Elevate — The Future Of Urban Air Transport*. `https://www.uber.com/info/elevate/`. (Accessed on 08/27/2018).

[19] Sergei Lupashin and Raffaello D’Andrea. “Stabilization of a flying vehicle on a taut tether using inertial sensing”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 2432–2438.

[20] A. Kalantari and M. Spenko. “Modeling and Performance Assessment of the HyTAQ, a Hybrid Terrestrial/Aerial Quadrotor”. In: *IEEE Transactions on Robotics* 30.5 (2014), pp. 1278–1285.

[21] John Ware and Nicholas Roy. “An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1507–1514.

[22] Nicola Bezzo et al. “Online planning for energy-efficient and disturbance-aware uav operations”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 5027–5033.

[23] Fabio Morbidi, Roel Cano, and David Lara. “Minimum-energy path generation for a quadrotor UAV”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1492–1498.

[24] Christos Ampatis and Evangelos Papadopoulos. “Parametric design and optimization of multi-rotor aerial vehicles”. In: *Applications of Mathematics and Informatics in Science and Engineering*. Springer, 2014, pp. 1–25.

[25] Analiza Abdilla, Arthur Richards, and Stephen Burrow. “Power and endurance modelling of battery-powered rotorcraft”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 675–680.

[26] Yash Mulgaonkar et al. “Power and weight considerations in small, agile quadrotors”. In: *Micro-and Nanotechnology Sensors, Systems, and Applications VI*. Vol. 9083. International Society for Optics and Photonics. 2014, 90831Q.

[27] Haomiao Huang et al. “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering”. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 3277–3282.

[28] Zhilong Liu, Raja Sengupta, and Alex Kurzhanskiy. “A power consumption model for multi-rotor small unmanned aircraft systems”. In: *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*. IEEE. 2017, pp. 310–315.

[29] Gordon J Leishman. *Principles of helicopter aerodynamics*. Cambridge university press, 2006.

[30] Nadia Kreciglowa, Konstantinos Karydis, and Vijay Kumar. “Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation”. In: *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*. IEEE. 2017, pp. 656–662.

[31] M Krstić and HH Wang. “Design and stability analysis of extremum seeking feedback for general nonlinear systems”. In: *Proceedings of the 36th Conference on Decision and Control*.

[32] B. Calli et al. “Comparison of extremum seeking control algorithms for robotic applications”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 3195–3202.

[33] Paolo Binetti et al. “Formation flight optimization using extremum seeking feedback”. In: *Journal of Guidance, Control, and Dynamics* 26.1 (2003), pp. 132–142.

[34] Maximilian Schulz et al. “High-speed, steady flight with a quadrocopter in a confined environment using a tether”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 1279–1284.

[35] John H Mathews. *Numerical methods for mathematics, science and engineering*. Prentice-Hall, 1992.

[36] Bitcraze. *Crazyflie 2.0*. 2018. URL: `www.bitcraze.io/crazyflie-2` (visited on 02/28/2019).

[37] Lorenz Meier et al. “PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision”. In: *Autonomous Robots* 33.1-2 (2012), pp. 21–39.

[38] Yongbo Chen, Shoudong Huang, and Robert Fitch. “Active SLAM for mobile robots with area coverage and obstacle avoidance”. In: *IEEE/ASME Transactions on Mechatronics* 25.3 (2020), pp. 1182–1192.

[39] Scott Driessens and Paul Pounds. “The Triangular Quadrotor: A More Efficient Quadrotor Configuration”. In: *IEEE Transactions on Robotics* 31 (Oct. 2015), pp. 1–10.

[40] C. Holda, B. Ghalamchi, and M. W. Mueller. “Tilting multicopter rotors for increased power efficiency and yaw authority”. In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2018, pp. 143–148.

[41] Hüseyin Turan Arat and Meryem Gizem Sürer. “Experimental investigation of fuel cell usage on an air Vehicle’s hybrid propulsion system”. In: *International Journal of Hydrogen Energy* 45.49 (2020). Progress in Hydrogen Production and Utilization, pp. 26370–26378.

[42] K. P. Jain and M. W. Mueller. “Flying batteries: In-flight battery switching to increase multirotor flight time”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3510–3516.

[43] Robert TN Chen and Yiyuan Zhao. *Optimal trajectories for the helicopter in one-engine-inoperative terminal-area operations*. National Aeronautics and Space Administration, Ames Research Center, 1996.

[44] Carmelo Di Franco and Giorgio Buttazzo. “Energy-aware coverage path planning of UAVs”. In: *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*. IEEE. 2015, pp. 111–117.

[45] T. M. Cabreira et al. “Energy-Aware Spiral Coverage Path Planning for UAV Photogrammetric Applications”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3662–3668.

[46] Nicolas Michel et al. “Multiphysical Modeling of Energy Dynamics for Multirotor Unmanned Aerial Vehicles”. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 738–747.

[47] D. Hong et al. “Least-Energy Path Planning With Building Accurate Power Consumption Model of Rotary Unmanned Aerial Vehicle”. In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 14803–14817.

[48] Patrick Bouffard, Anil Aswani, and Claire Tomlin. “Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 279–284.

[49] Guillem Torrente et al. “Data-Driven MPC for Quadrotors”. In: *IEEE Robotics and Automation Letters* (2021).

[50] Y. Tan et al. “Extremum seeking from 1922 to 2010”. In: *Proceedings of the 29th Chinese Control Conference*. 2010, pp. 14–26.

[51] Justin Creaby, Yaoyu Li, and John E. Seem. “Maximizing Wind Turbine Energy Capture Using Multivariable Extremum Seeking Control”. In: *Wind Engineering* 33.4 (2009), pp. 361–387.

[52] Scott J. Moura and Yiyao A. Chang. “Lyapunov-based switched extremum seeking for photovoltaic power maximization”. In: *Control Engineering Practice* 21.7 (2013), pp. 971–980.

[53] Hsin-Hsiung Wang, S. Yeung, and M. Krstic. “Experimental application of extremum seeking on an axial-flow compressor”. In: *IEEE Transactions on Control Systems Technology* 8.2 (2000), pp. 300–309.

[54] Mario A Rotea. “Analysis of multivariable extremum seeking algorithms”. In: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334).* Vol. 1. 6. IEEE. 2000, pp. 433–437.

[55] Miroslav Krstić and Hsin-Hsiung Wang. “Stability of extremum seeking feedback for general nonlinear dynamic systems”. In: *Automatica* 36.4 (2000), pp. 595–601.

[56] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR*. 2015.

[57] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *4th International Conference on Learning Representations, ICLR*. 2016.

[58] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[59] Azad Ghaffari, Miroslav Krstić, and Dragan Nešić. “Multivariable Newton-based extremum seeking”. In: *Automatica* 48.8 (2012), pp. 1759–1767.

[60] H.K. Khalil. *Nonlinear Systems*. 3rd ed. Pearson Education. Prentice Hall, 2002. ISBN: 9780130673893.

[61] N.S. Nise. *Control Systems Engineering*. 7th ed. Wiley, 2015. ISBN: 9781118170519.

[62] Ying Tan, Dragan Nešić, and Iven Mareels. “On the choice of dither in extremum seeking systems: A case study”. In: *Automatica* 44.5 (2008), pp. 1446–1450.

[63] PX4. *PX4 Drone Autopilot*. 2021. URL: `https://github.com/PX4/PX4-Autopilot/tree/v1.10.1` (visited on 04/30/2021).

[64] Tanja Baumann. “Obstacle Avoidance for Drones Using a 3DVFH* Algorithm”. Master thesis. ETH Zurich, 2015.

[65] Basak Sakcak et al. “Sampling-Based Optimal Kinodynamic Planning with Motion Primitives”. In: 43 (Oct. 2019), pp. 1715–1732.

[66] Sara Spedicato and Giuseppe Notarstefano. “Minimum-Time Trajectory Generation for Quadrotors in Constrained Environments”. In: 26 (July 2018), pp. 1335–1344.

[67] Junghee Park, Sisir Karumanchi, and Karl Iagnemma. “Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming”. In: 31 (Oct. 2015), pp. 1101–1115.

[68] Yuncheng Lu et al. “A survey on vision-based UAV navigation”. In: *Geo-spatial Information Science* 21.1 (2018), pp. 21–32.

[69] H. Oleynikova et al. “Continuous-time trajectory optimization for online UAV replanning”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 5332–5339.

[70] Sikang Liu et al. "High speed navigation for quadrotors with limited onboard sensing". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1484–1491.

[71] Jing Chen, Tianbo Liu, and Shaojie Shen. "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1476–1483.

[72] J. Tordesillas, B. T. Lopez, and J. P. How. "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1934–1940.

[73] B. Zhou et al. "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3529–3536.

[74] Boyu Zhou et al. "Robust real-time UAV replanning using guided gradient-based optimization and topological paths". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1208–1214.

[75] Pete Florence, John Carter, and Russ Tedrake. "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps". In: *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 304–319.

[76] B. T. Lopez and J. P. How. "Aggressive 3-D collision avoidance for high-speed navigation". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 5759–5765.

[77] J. Zhang et al. "Maximum Likelihood Path Planning for Fast Aerial Maneuvers and Collision Avoidance". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 2805–2812.

[78] Nathan Bucki, Junseok Lee, and Mark W Mueller. "Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4626–4633.

[79] N. Bucki and M. W. Mueller. "Rapid Collision Detection for Multicopter Trajectories". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 7234–7239.

[80] Armin Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: 34 (Apr. 2013), pp. 189–206.

[81] Davide Scaramuzza and Zichao Zhang. "Visual-inertial odometry of aerial robots". In: *arXiv preprint arXiv:1906.03289* (2019).

[82] Jur van den Berg, Pieter Abbeel, and Ken Goldberg. "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 895–913.

[83] Sachin Patil et al. "Scaling up Gaussian Belief Space Planning Through Covariance-Free Trajectory Optimization and Automatic Differentiation". In: *Eleventh International Workshop on the Algorithmic Foundations of Robotics, WAFR 2014, August 2014.*

[84] Mark W. Mueller, Seung Jae Lee, and Raffaello D'Andrea. "Design and Control of Drones". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), null.

[85] Bryan Penin et al. "Vision-based minimum-time trajectory generation for a quadrotor UAV". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2017, pp. 6199–6206.

[86] Igor Spasojevic, Varun Murali, and Sertac Karaman. "Perception-aware time optimal path parameterization for quadrotors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA).* 2020, pp. 3213–3219.

[87] Igor Spasojevic, Varun Murali, and Sertac Karaman. "Joint feature selection and time optimal path parametrization for high speed vision-aided navigation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2020, pp. 5931–5938.

[88] Varun Murali et al. "Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness". In: *2019 American Control Conference (ACC).* 2019, pp. 3936–3943.

[89] Davide Falanga et al. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS).* 2018.

[90] Luca Bartolomei, Lucas Teixeira, and Margarita Chli. "Perception-aware Path Planning for UAVs using Semantic Segmentation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2020, pp. 5808–5815.

[91] Christos Papachristos, Shehryar Khattak, and Kostas Alexis. "Uncertainty-aware receding horizon exploration and mapping using aerial robots". In: *2017 IEEE International Conference on Robotics and Automation (ICRA).* 2017, pp. 4568–4575.

[92] Christos Papachristos et al. "Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning". In: *Autonomous Robots* 43.8 (2019), pp. 2131–2161.

[93] Steven M LaValle. *Planning algorithms.* Cambridge university press, 2006.

[94] Zichao Zhang and Davide Scaramuzza. "Perception-aware receding horizon navigation for MAVs". In: *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2018, pp. 2534–2541.

[95] Junseok Lee et al. "Autonomous flight through cluttered outdoor environments using a memoryless planner". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS).* IEEE. 2021, pp. 1131–1138.

[96] Patrick Geneva et al. "OpenVINS: A Research Platform for Visual-Inertial Estimation". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Paris, France, 2020.

[97] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2520–2525.

[98] M. W. Mueller, M. Hehn, and R. D'Andrea. "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation". In: *IEEE Transactions on Robotics* 31.6 (Dec. 2015), pp. 1294–1310.

[99] Sergei Lupashin et al. "A platform for aerial robotics research and demonstration: The Flying Machine Arena". In: *Mechatronics* 24.1 (2014), pp. 41–54.

[100] Matthew Field et al. "Motion capture in robotics review". In: *2009 IEEE International Conference on Control and Automation*. 2009, pp. 1697–1702.

[101] Mark W. Mueller, Michael Hamer, and Raffaello D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1730–1736.

[102] Saman Fahandezh-Saadi and Mark W. Mueller. "Optimal measurement selection algorithm and estimator for ultra-wideband symmetric ranging localization". In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2018, pp. 229–235.

[103] Antoni Rosinol Vidal et al. "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 994–1001.

[104] Gabriel Nützi et al. "Fusion of IMU and vision for absolute scale estimation in monocular SLAM". In: *Journal of intelligent & robotic systems* 61.1 (2011), pp. 287–299.

[105] Cesar Cadena et al. "Simultaneous Localization And Mapping: Present, Future, and the Robust-Perception Age". In: *CoRR* abs/1606.05830 (2016).

[106] Martin Brossard, Axel Barrau, and Silvère Bonnabel. "AI-IMU dead-reckoning". In: *IEEE Transactions on Intelligent Vehicles* 5.4 (2020), pp. 585–595.

[107] Guillaume Allibert et al. "Estimating body-fixed frame velocity and attitude from inertial measurements for a quadrotor vehicle". In: *2014 IEEE Conference on Control Applications (CCA)*. 2014, pp. 978–983.

[108] James Svacha et al. "Inertial Velocity and Attitude Estimation for Quadrotors". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9.

[109] Oliver J Woodman. *An introduction to inertial navigation*. Tech. rep. University of Cambridge, Computer Laboratory, 2007.

[110] Isaac Skog et al. “Zero-Velocity Detection—An Algorithm Evaluation”. In: *IEEE Transactions on Biomedical Engineering* 57.11 (2010), pp. 2657–2666.

[111] E. Foxlin. “Pedestrian tracking with shoe-mounted inertial sensors”. In: *IEEE Computer Graphics and Applications* 25.6 (2005), pp. 38–46.

[112] Mark W Mueller. “A dynamics-agnostic state estimator for unmanned aerial vehicles using ultra-wideband radios”. In: *Dynamic Systems and Control Conference.* Vol. 51913. American Society of Mechanical Engineers. 2018, V003T36A002.

[113] Mark W. Mueller, Markus Hehn, and D’Andrea. Raffaello. “Covariance Correction Step for Kalman Filtering with an Attitude”. In: *Journal of Guidance, Control, and Dynamics* 40.9 (Nov. 2016), pp. 2301–2306.

[114] Jiaming Zha et al. “A collision-resilient aerial vehicle with icosahedron tensegrity structure”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2020, pp. 1407–1412.

[115] Joono Cheong and Robert E Skelton. “Nonminimal dynamics of general class k tensegrity systems”. In: *International Journal of Structural Stability and Dynamics* 15.02 (2015), p. 1450042.

[116] Hidenori Murakami. “Static and dynamic analyses of tensegrity structures. Part 1. Nonlinear equations of motion”. In: *International Journal of Solids and Structures* 38.20 (2001), pp. 3599–3613.