Implementing Tracking Error Control for Quadrotor UAV

Master's Thesis
DC 2021.044

Author
Xinyu Zeng

Supervisors
prof. dr. H. Nijmeijer
Coach
dr.ir. A.A.J. Lefeber

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control

Eindhoven, 4th May 2021

Abstract

In previous research, uniform almost global asymptotic stability of a controller for Autonomous Unmanned Areal Vehicles (UAVs or drones) has been proved. This project aims at implementing this controller on a Parrot Mambo drone to validate its stability in reality. By derivating this controller, we detail the connection between position control and attitude control via a virtual input. Besides, a flat-out-based reference is determined for smooth tracking. For successful implementation, we first introduce the drone hardware and analyze the default controller and estimator of the Parrot Simulink Package. Due to the broad range of parameters in [26] and the robust performance of the default controller, we decide to resemble the linearized default controller to settle a set of robust gains for our controller. Subsequently, our controller achieves zero-error tracking in simulations via the Parrot Simulink Package. However, the experiments illustrate undesired oscillation. Through the eigenvalue checking and estimator validating, one reason comes from the mini-size of the drone, which cannot stand the robust but high gains. Another reason is the inaccurate horizontal estimation of the optical flow sensor. A Kalman filter brings the drifting acceleration into the horizontal velocity estimator. And the crude default optical flow estimator ignores the rotation in the optical flow field. Facilitating the optical flow theory, we obtain a theoretical horizontal estimator excluding filters. Its accuracy improves $4\% - 33\%$ in pendulum validations.

Table of Contents

Nomenclature

Reference frames

$\mathcal{I}$	Right-handed orthonormal earth-fixed frame in North-East-Down (NED) configuration
$\mathcal{B}$	Right-handed orthonormal body-fixed frame in North-East-Down (NED) configuration

Number sets

$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	The n-dimensional Euclidian space
SO(3)	The 3D rotation special orthogonal group

Operators

A^T	The transpose of the vector or matrix A
$\dot{x}$	Time derivative of a state x
$\text{Conj}(\boldsymbol{q}), q^*$	The conjugate of a quaternion $\boldsymbol{q}$
$\text{norm}(\boldsymbol{q}), \|q\|$	The square root of the product of a quaternion q with its conjugate
$\otimes$	Quaternion product
$\odot$	Quaternion rotation
$\text{eig}(A)$	The eigenvalues of matrix A
$\det(A)$	The determinant of matrix A
$\text{diag}(A)$	The diagonal of matrix A
$S(x)$	The skew symmetric matrix of a vector x

Acronyms

UAV	Unmanned Areal Vehicle
CPU	Central Processing Unit
IMU	Internal Measurement Unit
PWM	Pulse-Width Modulation

PSP	Parrot Simulink Package
VGA	Video Graphics Array

Constants and variables

ρ	Earth-fixed position vector
ν	Body-fixed velocity vector
a	Body-fixed acceleration vector
ϕ, θ, ψ	Roll, Pitch, Yaw angle of a body
R	Rotation matrix
q	quaternion
ω	Body-fixed angular velocity vector
f	Upward force generated by the rotors
τ	Totally torque generated by the rotors
m	Mass
J	Inertial tensor
g	Gravity acceleration
(du, dv)	Measured optical flow
λ	Eigenvalue of considered system
I	3 by 3 identity matrix

Chapter 1

Introduction

1.1 Background

The quadrotor(drone) is a kind of unmanned aerial vehicle(UAV) with two pairs of opposite propellers with clockwise and counter-clockwise rotation to balance the torque [41]. Its position and attitude are controlled by changing thrusts provided by four propellers using pulse width modulation (PWM) [41]. From the last century, UAVs gradually developed for the civilian market. As an aircraft without a human pilot, it provides a possible choice to perform more subtle tasks in different environments. Nowadays, agriculture drones can observe the wild field issues from a bird's-eye view and carry pesticides to spray evenly. In delivering and online shopping companies, delivery drones are considered a fast and private way for customers.

Even though the robustness, low weight, and small size gain attention to the quadrotor, in the past decades few commercial quadrotor platforms are available for customization and expensive for research [28]. At this point, some research groups build specific platforms referring to their needs [28]. For example, ETH Zurich's quadrotor embeds a Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications [18]. Or the OS4 project from Ecole Polytechnique Fédérale de Lausanne develops a vehicle with a full state backstepping controller [6]. However, recently with the expansion of the market, more and more companies provide cheaper and nearly open platforms with low-cost onboard sensors, whose utilization now is a favorable basement for control design.

As four individual rotors of quadrotor equivalent to one main rotor on a helicopter, quadrotor can realize smaller airframe size and Pitch-fixed parallels without complex mechanical linkages [20]. However, its advantages are costed by coupled dynamics and under-actuated tracking because of only four degrees of freedom [29]. Furthermore, another challenge is how to overcome the global asymptotic stability [26] under several uncertainties of UAVs' highly nonlinear flight dynamics [24][41]. Therefore, designing a better auto-navigation algorithm of the quadrotor is a popular issue in control research.

In previous studies of the Dynamics and Control research group at the Eindhoven University of Technology, Jeurgen [21] and van den Eijnden [38] have identified the dynamics of an AR. Drone 2.0 and designed a cascade-based controller for it. Brekelmans [7] extended this controller from simulation to experiments with unit quaternions and identified vehicle parameters of the Parrot Mambo drone. Following these works, this report focuses on implementing the feedback controller of [26] on the Parrot Mambo drone, where the steps involving the combination of controllers and its parameters tuning from simulation to an actual drone. The bridge between position and attitude controller, however, is rarely discussed in [26]. For such a case, this report aims to detail the derivation of the controller in [26], the gains tuning by linear approximation with a robust

proportional–integral–derivative controller, and the estimation improvement for the optical flow sensor. Specifically, to avoid delay from the overlap between the default filtered estimator and designed observer [26] a raw estimator without filters should be redesigned.

1.2 Objective

Enabling autonomous navigation with the controller of previous research in Eindhoven [26] on a Parrot Mambo drone involves several problems. Although [26] have theoretically proved the uniform almost global asymptotic stability for this controller, it still waits for experimental validations. Since the given theoretical constraints of gains are too broad for experimental tuning, the principal question of this project is to find out the feasible and robust range for the gains, which relate to the vehicle characters and the rotor limitation. Moreover, due to the designed combination of observer and controller in [26], the unfiltered states from raw data are expected from the state estimator, where the default estimation algorithm in PSP needs to adjust even redesign. Specifically, the continuous horizontal drift from the optical flow estimator mentioned in Brekelmans' report [7] would be a focus, and this project would provide a solution. Therefore, given the ultimate goal, this project is separated into several sub-objectives, from theoretical derivation to final validation.

A quadcopter dynamics model should be first explained for flight relating to reference choice and control design. Following this model, a sufficient smooth reference based on flat-output is derived. As the first control attempt, we detail the combination of transition and attitude control laws in [26]. Then we explore the flight platforms, Parrot mambo drone, and its Simulink package, to prepare for experiments. And to avoid undesired performance in the experiments, a linear approximation between the default controller and the designed controller is necessary for feedback gains tuning. Besides tuning for stability, excluding unnecessary filters from the default estimator and tuning the designed observer become the next assignment. Moreover, a new horizontal motion algorithm based on optical flow theory solves the drift mentioned in [7], which is validated by pendulum experiments. With above preparation, this time, we were able to simulate and implement stably with suitable gains for hover condition.

1.3 Thesis outline

This thesis consists of six chapters including the current background chapter introduced above. Chapter 2 outlines the notation, definition [26], and formula for the remainder of this report. It contains three attitude representations, time differentiation formulas, and quadcopter flight dynamics. Chapter 3 derives a flat-output based reference model and the controller in [26]. It explains each sufficient smooth reference state without integral and the combination for the position and attitude control. Furthermore, a unit quaternion form of the designed controller is derived from the rotation matrix form in [26] for faster computation at the end of this chapter. Chapter 4 briefly presents the vehicle parameters and on-board sensor of the Parrot Mambo drone for simulations and experiments. Using the detected data from the on-board sensors, this chapter explains the default algorithms of state estimator and controller in PSP. These default algorithms are then linearized in Chapter 5 to approximate the designed controller for gains tuning in simulation and experiments. Additionally, a pure state estimator correcting for horizontal drift without filters is validated. Finally, Chapter 6 concludes this thesis work and summarises the recommendations for future research.

Chapter 2

Preliminaries

In this chapter, we begin with the comparison of orientational representation forms: rotation matrix, unit quaternions, and Euler angles, which are discussed for efficiency and intuitiveness. To be convenient, two specific time differential forms are derived for the quadcopter model. The flight dynamics model is given in section 2.3 [25] [26] as an overview of UAV's flight states. These specifications involve the model expressions in the remainder of this report.

2.1 Attitude representation

An attitude representation defines the rotational relationship of one given frame relative to another frame. In this report, all frames are considered as North-East-Down(NED). We consider three representations: rotation matrix, unit quaternions and Euler angles and compare their singularity, calculational complexity and storage to decide the implemented form in this section.

2.1.1 Rotation matrix

As a common affine form, the rotation matrix delivers the rotation transformation under homogeneous coordinates [40] to readers intuitively and accurately. In Euclidean space, a square 3 by 3 matrix R is a rotation matrix, if and only if R satisfies

$$R^{-1} = R^T \tag{2.1a}$$

$$\det(R) = 1, \tag{2.1b}$$

where the set of all matrices satisfying (2.1) is denoted as the 3D rotation special orthogonal group, SO(3).

In literature, a rotation of a vector is represented as

$$\rho_{rot} = R\rho, \tag{2.2}$$

where $\rho, \rho_{rot} \in \mathbb{R}^{3\times 1}$ are the pre- and post-rotated vector; $R \in \mathbb{R}^{3\times 3}$ is the rotation matrix. To calculate a rotated vector, 6 additions and 9 multiplications are needed. A sequential rotation is simply multiplied as

$$R_{rot} = R_2 R_1 \tag{2.3}$$

where R_i is the i_{th} rotation in sequence. To calculate a sequenced rotation, 18 additions and 27 multiplications are required.

2.1.2 Unit quaternion

An alternative representation of attitude is unit quaternion. In 1843, William Rowan Hamilton extended the complex numbers to four dimensions as quaternions;

$$\boldsymbol{q} = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} = [q_0, \boldsymbol{q_{1:3}}]^T \tag{2.4}$$

where i, j and k hold similar properties as complex numbers

$$i^2 = j^2 = k^2 = ijk = -1, \tag{2.5a}$$
$$ij = k, \qquad ji = -k, \tag{2.5b}$$
$$jk = i, \qquad kj = -i, \tag{2.5c}$$
$$ki = j, \qquad ik = -j. \tag{2.5d}$$

For convenience, in the remainder a quaternion is expressed as

$$\boldsymbol{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T. \tag{2.6}$$

Furthermore, we have

$$\text{Conj}(\boldsymbol{q}) = \boldsymbol{q}^* = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T, \tag{2.7a}$$
$$\text{Norm}(\boldsymbol{q}) = \|\boldsymbol{q}\| = \sqrt{q^*q} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \tag{2.7b}$$
$$\boldsymbol{q}^{-1}\boldsymbol{q} = \frac{\boldsymbol{q}^*\boldsymbol{q}}{\boldsymbol{q}^*\boldsymbol{q}} = 1, \tag{2.7c}$$
$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|^2}. \tag{2.7d}$$

Based on (2.5), the multiplication $\otimes$ of two quaternions is denoted as

$$\begin{aligned} q \otimes p &= (q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k})(p_0 + p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k}) \\ &= (q_0p_0 - (q_1p_1 + q_2p_2 + q_3p_3)) + (q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2)\boldsymbol{i} + (q_0p_2 + q_2p_0 - q_1p_3 + q_3p_1)\boldsymbol{j} + (q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1)\boldsymbol{k} \\ &= \begin{bmatrix} q_0p_0 - (q_1p_1 + q_2p_2 + q_3p_3) \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 + q_2p_0 - q_1p_3 + q_3p_1 \\ q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1 \end{bmatrix}. \end{aligned} \tag{2.8}$$

Note that there are only 16 multiplications and 12 additions in the quaternions multiplication calculation (2.8). And because the conjugation is its own inverse [5], conjugating the product (2.8) gives

$$(q \otimes p)^* = p^* \otimes q^* \tag{2.9}$$

To express the rotation, Hamilton denoted a quaternion with norm one $\{q \mid \|q\| = 1\}$ as a versor(unit quaternion)

$$q = \begin{bmatrix} \cos\frac{\theta}{2} \\ \vec{n}\sin\frac{\theta}{2} \end{bmatrix}, \tag{2.10}$$

where by the axis–angle method [11] the Euler axis $\vec{n}$ satisfies $\|\vec{n}\| = 1$ and $\theta \in [0, 2\pi)$ denotes the clockwise rotation angle around $\vec{n}$.

Assuming that a vector $z_{rot} \in \mathbb{R}^3$ is rotated from initial vector $z \in \mathbb{R}^3$ by a versor q, the rotation

$$\begin{bmatrix} 0 \\ z_{rot} \end{bmatrix} = q \odot z \tag{2.11}$$

is pre- and post-multiplied by the quaternion and its conjugate (see Rodrigues' rotation formula [32])

$$
\begin{aligned}
q \odot z = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ z \end{bmatrix} \otimes \boldsymbol{q}^* &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 0 & 2(q_0q_3+q_1q_2) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_1q_3-q_0q_1) \\ 0 & 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \begin{bmatrix} 0 \\ z \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0_{1\times 3} \\ 0_{3\times 1} & Q(q) \end{bmatrix} \begin{bmatrix} 0 \\ z \end{bmatrix}
\end{aligned} \tag{2.12}
$$

This driving expression (2.12) shows 18 multiplications and 12 additions.

Note that due to the existence of the quadratic term in (2.12), $Q(q)$ implies the same rotation from either positive or negative versor. However, the ambiguous mapping is unidirectional from rotated vector to versor, where converting the whole system into versor before calculation would avoid the uncertainty.

Additionally, the time derivative of a versor would be easily derived from (2.10) [13] [33]

$$
\begin{aligned}
\dot{q} &= \lim_{\Delta t \to 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{q(t) \otimes q(\Delta t) - q(t)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{q \otimes (\begin{bmatrix} 1 \\ \vec{n}\frac{\theta}{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 0_{3\times 1} \end{bmatrix})}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{q \otimes \begin{bmatrix} 0 \\ \vec{n}\frac{\theta}{2} \end{bmatrix}}{\Delta t} \\
&= \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix},
\end{aligned} \tag{2.13}
$$

where θ is a small disturbance of the rotation angle around Euler axis $\boldsymbol{n}$; $\omega = \vec{n}\dot{\theta}$ is the angular rate.

2.1.3 Euler angles

As the most common way, Euler angles are intuitive and brief for readers and broadly used in the IMU sensor. It uses three intrinsic rotations angles(ϕ, θ, ψ) around XYZ frame axes. In the aircraft field, the Roll, Pitch, Yaw angles(RPY) are widely used in the North-East-Down(NED) frame.

In Euclidean space, following the ZYX rotation sequence the conversion from Euler angles to rotation matrix is given by

$$
R_{ZYX} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.14}
$$

and from Euler angles to unit quaternion

$$
q = \begin{bmatrix} \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \end{bmatrix}. \tag{2.15}
$$

Both can be computed uniquely. However, there is a unidirectional disadvantage of Euler angles named as Gimbal Lock leading to singularity and uncertainty in mathematics, when Euler angles [17] are decomposed from rotation matrix or unit quaternion; for instance, once Pitch angle $\theta = \frac{\pi}{2}$ in (2.14), the Roll and Yaw angles cannot be derived uniquely from one rotation matrix.

2.1.4 Comparison of attitude representations

As was shown in the above sections, to avoid the singularity of attitude it must choose rotation matrix or unit quaternion for flight control, even though Euler angles is still used in derivations. In sight the calculations between the rotation matrix(2.2,2.3) and unit quaternion(2.8,2.12) as Table.2.1, unit quaternions shows less 17 calculations in itself multiplication but 15 more steps for its rotation. Moreover considering only four elements of storage of a versor rather than nine

Action	Additions	Multiplications
Rotation matrix multiplication	18	27
Unit quaternion multiplication	12	16
Rotation matrix rotation	6	9
Unit quaternion rotation	12	18

Table 2.1: The computation amount of quaternions and rotation matrix [14] [7]

elements of a rotation matrix the implementation chooses the unit quaternion as the final form; though to avoid ambiguity the control design is exhibited by a rotation matrix.

2.2 Time differentiation

In this section, there are two particular time differential forms introduced as pre-knowledge. One is for rotation dynamics, and another one is for 2-norms.

If R is a rotation matrix and $\omega = [\omega_1, \omega_2, \omega_3]^T$ is the angular velocity, let $S(\omega)$ denote the skew-symmetric map

$$S(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{2.16}$$

The equivalence that the cross product $a \times b = S(a)b$ [26] can be used for the time differentiation of rotation dynamics

$$\dot{R} = R \times \omega = RS(\omega). \tag{2.17}$$

In the remainder, we define 2-norm of $x = [x_1, x_2, x_3]^T \in \mathbb{R}^3$ as

$$\|x\| = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + x_3^2}. \tag{2.18}$$

Its time derivative is

$$\frac{d}{dt}\|x\| = \frac{1}{2\sqrt{x_1^2+x_2^2+x_3^2}}(2x_1\dot{x}_1 + 2x_2\dot{x}_2 + 2x_3\dot{x}_3) = \frac{x^T\dot{x}}{\|x\|}. \tag{2.19}$$

2.3 Quadcopter flight dynamics

As was shown in Figure 2.1, the quadcopter can be regarded as a rigid body and only provides thrust perpendicular to its upper surface. Following [26], two frames are introduced here to

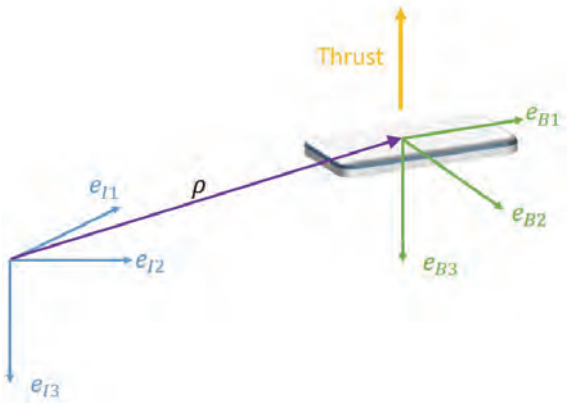


Figure 2.1: Physical schematic model of drone

describe translation and rotation of the drone. Inertial frame $\mathcal{I} = [e_{I1}, e_{I2}, e_{I3}]$ is a North-East-Down(NED) earth-fixed frame and body-fixed frame $\mathcal{B} = [e_{B1}, e_{B2}, e_{B3}]$, is fixed in the drone's center of mass, whose z-axis e_{B3} is always in opposite direction of the thrust. The translation vector between two frames, $\mathcal{B}$ with respect to $\mathcal{I}$, is presented as $\rho = [x, y, z]^T \in \mathbb{R}^3$., and the rotation matrix $R \in SO(3)$ denotes the intrinsic rotation from $\mathcal{B}$ to $\mathcal{I}$. Furthermore, the angular velocity $\omega \in \mathbb{R}^3$ and linear velocity $\nu \in \mathbb{R}^3$ are both denoted in $\mathcal{B}$. The UAV flight dynamics can be described in the following way [25];

$$\dot{\rho} = R\nu, \tag{2.20a}$$

$$\dot{\nu} = -S(\omega)\nu + gR^T e_3 - \frac{f}{m}e_3, \tag{2.20b}$$

$$\dot{R} = RS(\omega), \tag{2.20c}$$

$$J\dot{\omega} = S(J\omega)\omega + \tau, \tag{2.20d}$$

where m is the mass of drone $J = \text{diag}[J_{11}, J_{22}, J_{33}]$ is the drone's moment of inertia in $\mathcal{B}$, f is the thrust magnitude, $\tau = [\tau_1, \tau_2, \tau_3]^T$ is the torque vector, and g is the gravity acceleration.

2.4 Concluding remarks

This chapter has described three attitude presentations: rotation matrix, unit quaternion, and Euler angles. Given the earlier comparison and discussion in section 2.1, we decide the unit quaternions as the final form in code and the rotation matrix form in literature for intuition. The two crucial time differentiating rules have been introduced for the remainder. In the end, the introduction of quadcopter dynamics takes a insight into the system and prepares the theoretical analysis for control design in the next chapter.

Chapter 3

Quadcopter model

This chapter introduces the quadcopter UAV's model for the Parrot mambo drone, which gathers a flat-output-based reference and a controller designed in [26]. The standard derivation of the flat-output-based reference is given in section 3.2, which is sufficiently smooth without integration [26] used to exam the controller by highly demanding state-trajectory. Next, section 3.2 details the very information about the controller [26]: a combination of position and attitude feedback control with a denoising observer. Finally, for convenience, a versor form of the full controller used in implementation is presented in section 3.2.3.

3.1 Flat-output based reference

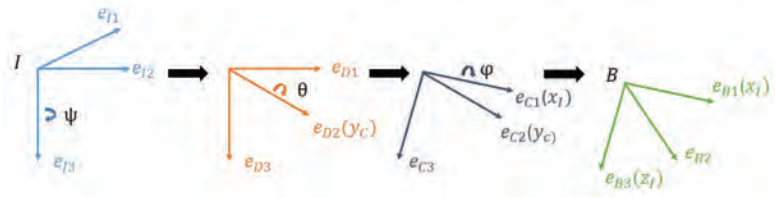


Figure 3.1: Rotation order from inertial frame to body frame

As introduced in Chapter 2, a feasible continuous reference dynamics should satisfy (2.20) as

$$\dot{\rho}_r = R_r \nu_r \tag{3.1a}$$
$$\dot{\nu}_r = -S(\omega_r)\nu_r + gR_r^T e_3 - (f_r/m)e_3 \tag{3.1b}$$
$$\dot{R}_r = R_r S(\omega_r) \tag{3.1c}$$
$$J\dot{\omega}_r = -S(J\omega_r)\omega_r + \tau_r. \tag{3.1d}$$

A desired trajectory for UAV's controller tracking should be sufficiently smooth even given aggressive requirements [26]. Considering there are only four degrees of freedom for the drone, the given reference γ only needs four dimensions. That is

$$\begin{bmatrix}\gamma_1(t)\\ \gamma_2(t)\\ \gamma_3(t)\end{bmatrix} = \begin{bmatrix}x_r(t)\\ y_r(t)\\ z_r(t)\end{bmatrix} = \rho_r(t) \tag{3.2a}$$
$$\gamma_4(t) = \psi_r(t) \tag{3.2b}$$

where all γ are four times differentiable; if $\ddot{z}_r = g$, $-\ddot{x}_r \sin\psi_r + \ddot{y}_r \cos\psi_r \neq \pm\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2}$ (see (3.8)); the first three dimensions describe the position of the UAV and satisfy $\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2 \neq 0$(see (3.5)); the fourth one γ_4 is defined as the yaw angle denoting the ZYX extrinsic rotation order (see Figure 3.1,yaw angle ψ, pitch angle θ, roll angle ϕ) from inertial frame to body frame.

Besides the given position ρ_r, the rest states $[f_r, \dot{f}_r, \ddot{f}_r, R_r, \dot{R}_r, v_r, \omega_r, \dot{\omega}_r, \tau_r]$ of the reference dynamics are derived below.

3.1.1 Force scale

Observing equations (3.1), the force scale f_r of the reference may be normed and rearranged from the time derivative of (3.1a). That is

$$\ddot{\rho}_r = \dot{R}_r \nu_r + R_r \dot{\nu}_r = g e_3 - (f_r/m) R_r e_3. \tag{3.3a}$$

$$f_r = m\|g e_3 - \ddot{\rho}_r\| = m\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2}. \tag{3.3b}$$

According to (2.19), the first order and second order time derivatives of f_r can be written as

$$\dot{f}_r = -m\frac{(g e_3 - \ddot{\rho}_r)^T \rho_r^{(3)}}{\|g e_3 - \ddot{\rho}_r\|} \tag{3.4a}$$

$$\ddot{f}_r = m\frac{(\rho_r^{(3)})^T \rho_r^{(3)}}{\|g e_3 - \ddot{\rho}_r\|} - m\frac{(g e_3 - \ddot{\rho}_r)^T \rho_r^{(4)}}{\|g e_3 - \ddot{\rho}_r\|} - \frac{\dot{f}_r^2}{\|g e_3 - \ddot{\rho}_r\|}, \tag{3.4b}$$

where $\rho_r^{(i)}$ is the i-th order time derivative of position reference ρ_r.

3.1.2 Rotation matrix

Determining a unique rotation matrix in SO(3) requires four independent parameters. From equation (3.3a), the first three parameters are obtained

$$R_r e_3 = \frac{m}{f_r}(g e_3 - \ddot{\rho}_r) = \frac{1}{\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2}} \begin{bmatrix} -\ddot{x}_r \\ -\ddot{y}_r \\ g - \ddot{z}_r \end{bmatrix} := \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} := r, \tag{3.5}$$

provided $\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2 \neq 0$.

Referring to (3.2b), it denotes a yaw angle ψ_r in a ZYX rotation sequence from the inertial frame to the body frame(see (2.14)), which leads to

$$R_r = R_{ZYX}^T(\psi_r, \theta_r, \phi_r) = \begin{bmatrix} \cos\psi_r & -\sin\psi_r & 0 \\ \sin\psi_r & \cos\psi_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_r & 0 & \sin\theta_r \\ 0 & 1 & 0 \\ -\sin\theta_r & 0 & \cos\theta_r \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_r & -\sin\phi_r \\ 0 & \sin\phi_r & \cos\phi_r \end{bmatrix}, \tag{3.6}$$

where ϕ_r, θ_r are temporally introduced as roll and pitch angle.

Substitution of (3.6) into (3.5) gives

$$\begin{aligned} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} &= \begin{bmatrix} \cos\psi_r & -\sin\psi_r & 0 \\ \sin\psi_r & \cos\psi_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_r & 0 & \sin\theta_r \\ 0 & 1 & 0 \\ -\sin\theta_r & 0 & \cos\theta_r \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_r & -\sin\phi_r \\ 0 & \sin\phi_r & \cos\phi_r \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \sin\phi_r \sin\psi_r + \cos\phi_r \cos\psi_r \sin\theta_r \\ \cos\phi_r \sin\psi_r \sin\theta_r - \cos\psi_r \sin\phi_r \\ \cos\phi_r \cos\theta_r \end{bmatrix}. \end{aligned} \tag{3.7}$$

Then it is possible to write the corresponding cosine and sine functions of the unknown roll and pitch angles ϕ_r, θ_r as

$$\sin\phi_r = r_1 \sin\psi_r - r_2 \cos\psi_r \tag{3.8a}$$

$$\cos\phi_r = \sqrt{1-(r_1 \sin\psi_r - r_2 \cos\psi_r)^2} \tag{3.8b}$$

$$\cos\theta_r = -\frac{r_3}{\sqrt{1-(r_1 \sin\psi_r - r_2 \cos\psi_r)^2}} \tag{3.8c}$$

$$\sin\theta_r = \frac{r_1 \sin\psi_r + r_2 \cos\psi_r}{\sqrt{1-(r_1 \sin\psi_r - r_2 \cos\psi_r)^2}}. \tag{3.8d}$$

Substituting (3.8) into (3.6) we have

$$R_r = \begin{bmatrix} \frac{r_3\cos\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & -\frac{r_1 r_2\cos\psi+(1-r_1^2)\sin\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & r_1 \\ \frac{r_3\sin\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & \frac{r_1 r_2\sin\psi+(1-r_2^2)\cos\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & r_2 \\ -\frac{r_1\cos\psi+r_2\sin\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & \frac{(r_1\sin\psi-r_2\cos\psi)r_3}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}} & r_3 \end{bmatrix}. \tag{3.9}$$

Its time derivative is

$$\dot{R}_r = \begin{bmatrix} \frac{\dot{r}_3\cos\psi - r_3\sin\psi\dot{\psi}}{C} - \frac{r_3\cos\psi}{C^2}\dot{C} & dR_{12} & \dot{r}_1 \\ \frac{\dot{r}_3\cos\psi + r_3\cos\psi\dot{\psi}}{C} - \frac{r_3\sin\psi}{C^2}\dot{C} & dR_{22} & \dot{r}_2 \\ -\frac{\dot{r}_1\cos\psi - r_1\sin\psi\dot{\psi} + \dot{r}_2\sin\psi + r_2\cos\psi\dot{\psi}}{C} + \frac{r_1\cos\psi + r_2\sin\psi}{C^2}\dot{C} & dR_{32} & \dot{r}_3 \end{bmatrix} \tag{3.10}$$

$$dR_{12} = -\frac{\dot{r}_1 r_2\cos\psi + r_1\dot{r}_2\cos\psi - r_1 r_2\sin\psi\dot{\psi} - 2r_1\dot{r}_1\sin\psi + (1-r_1^2)\cos\psi\dot{\psi}}{C} + \frac{r_1 r_2\cos\psi + (1-r_1^2)\sin\psi}{C^2}\dot{C}$$

$$dR_{22} = \frac{\dot{r}_1 r_2\sin\psi + r_1\dot{r}_2\sin\psi + r_1 r_2\cos\psi\dot{\psi} - 2r_1\dot{r}_2\cos\psi - (1-r_2^2)\sin\psi\dot{\psi}}{C} + \frac{r_1 r_2\sin\psi + (1-r_2^2)\cos\psi}{C^2}\dot{C}$$

$$dR_{32} = \frac{(r_1\sin\psi - r_2\cos\psi)\dot{r}_3 + (\dot{r}_1\sin\psi - r_1\cos\psi\dot{\psi} - \dot{r}_2\cos\psi + r_2\sin\psi\dot{\psi})r_3}{C} + \frac{(r_1\sin\psi - r_2\cos\psi)r_3}{C^2}\dot{C},$$

where $C = \sqrt{1-(r_1\cos\psi - r_2\sin\psi)^2}$ and $\dot{C} = \frac{(-2(r_1\cos\psi - r_2\sin\psi))(\dot{r}_1\cos\psi - r_1\sin\psi\dot{\psi} - \dot{r}_2\sin\psi - r_2\cos\psi\dot{\psi})}{2\sqrt{1-(r_1\cos\psi - r_2\sin\psi)^2}}$.

Note that once $\ddot{z}_r$ equals to 0, i.e. $r = [r_1, r_2, 0]^T$, there is a risk that $\cos\phi_r$ becomes 0, i.e. $\phi_r = \pm\frac{\pi}{2}$, leading to two conditions: $r = [-\sin\psi, \cos\psi, 0]^T$ and $r = [\sin\psi, -\cos\psi, 0]^T$. Taking $\phi_r = \frac{\pi}{2}$ as an example, (3.7) is rewritten as

$$\begin{bmatrix} r_1 \\ r_2 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2}} \begin{bmatrix} -\ddot{x}_r \\ -\ddot{y}_r \\ 0 \end{bmatrix} = \begin{bmatrix} \sin\psi_r \\ -\cos\psi_r \\ 0 \end{bmatrix}, \tag{3.11}$$

where θ_r cannot be determined in this case. Therefore, to ensure the accessibility of θ_r, we have a constraint that if $\ddot{z}_r = g$, $-\ddot{x}_r\sin\psi_r + \ddot{y}_r\cos\psi_r \neq \pm\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2}$ to avoid $\cos\phi_r = 0$.

3.1.3 Linear velocity, angular rate and torque

With a complete rotation matrix R_r, the linear velocity v_r can be derived from (3.1a) as

$$v_r = R_r^T \dot{\rho}_r \tag{3.12}$$

with $\dot{\rho}_r = [\dot{x}_r, \dot{y}_r, \dot{z}_r]^T$.

And the angular velocity $\omega_r = [\omega_1, \omega_2, \omega_3]^T$ follows from (3.1c):

$$S(\omega_r) = R_r^T \dot{R}_r. \tag{3.13}$$

Substituting (3.9) and (3.11) into (3.13) gives

$$\omega_r = \begin{bmatrix} \frac{\dot{r}_1 \cos\psi - \dot{r}_2 \sin\psi}{\sqrt{1-(r_1\cos\psi + r_2\sin\psi)^2}} \\ \frac{r_3\cos\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}}\dot{r}_1 + \frac{r_3\sin\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}}\dot{r}_2 - \frac{r_1\cos\psi + r_2\sin\psi}{\sqrt{1-(r_1\cos\psi+r_2\sin\psi)^2}}\dot{r}_3 \\ -\frac{r_3\cos\psi(r_1\sin\psi - r_2\cos\psi)}{1-(r_1\cos\psi+r_2\sin\psi)^2}\dot{r}_1 - \frac{r_3\sin\psi(r_1\sin\psi - r_2\cos\psi)}{1-(r_1\cos\psi+r_2\sin\psi)^2}\dot{r}_2 + \frac{(r_1\cos\psi+r_2\sin\psi)(r_1\sin\psi - r_2\cos\psi)}{1-(r_1\cos\psi+r_2\sin\psi)^2}\dot{r}_3 + \frac{r_3\dot{\psi}}{1-(r_1\sin\psi - r_2\cos\psi)^2} \end{bmatrix} \tag{3.14}$$

which can be simplified as

$$\omega_r = \begin{bmatrix} \frac{1}{r_3}(R_{r21}\dot{r}_1 - R_{r11}\dot{r}_2) \\ R_{r11}\dot{r}_1 + R_{r21}\dot{r}_2 + R_{r31}\dot{r}_3 \\ -\frac{R_{r32}}{r_3}\omega_2 + \frac{r_3}{\cos^2\phi}\dot{\psi} \end{bmatrix}, \tag{3.15}$$

where R_{rij} is the corresponding element in row i and column j of rotation matrix R_r (3.9). Subsequently, the angular acceleration can be determined by differentiating (3.15):

$$\begin{aligned} \dot{\omega}_r &= \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{r_3}\dot{r}_3\omega_1 + \frac{1}{r_3}(\dot{R}_{r21}\dot{r}_1 + R_{r21}\ddot{r}_1 - \dot{R}_{r11}\dot{r}_2 - R_{r11}\ddot{r}_2) \\ \dot{R}_{r11}\dot{r}_1 + R_{r11}\ddot{r}_1 + \dot{R}_{r21}\dot{r}_2 + R_{r21}\ddot{r}_2 + \dot{R}_{r31}\dot{r}_3 + R_{r31}\ddot{r}_3 \\ -(\frac{\dot{R}_{r32}}{r_3}\omega_2 - \frac{R_{r32}}{r_3^2}\dot{r}_3\omega_2 + \frac{R_{r32}}{r_3}\dot{\omega}_2) + (\frac{\dot{r}_3}{\cos^2\phi_r}\dot{\psi}_r - 2\frac{\dot{r}_3}{\cos^3\phi_r}\cos\phi_r\sin\phi_r\dot{\phi}_r\dot{\psi}_r + \frac{r_3}{\cos^2\phi_r}\ddot{\psi}_r) \end{bmatrix}, \end{aligned} \tag{3.16}$$

where $\dot{R}_{rij}$ is the corresponding element in row i and column j of the time derivative of rotation matrix $\dot{R}_r = R_r S(\omega_r)$ (3.1c).

Consequently, the torque τ_r of the reference can be rearranged from (3.1d) as

$$\tau_r = J\dot{\omega}_r - S(J\omega_r)\omega_r. \tag{3.17}$$

3.2 Controller by E. Lefeber(2020)

The study by [26] innovates a filtered output feedback tracking control for quadcopters, which has proved uniform almost global asymptotic stability(UaGAS) in literature [26]. In section 3.2.1 and 3.2.2 respectively a position controller and an attitude controller are presented. In the end, the versor form of the combined controller is stated in section 3.2.3.

3.2.1 Position tracking error control

This section addresses a feedback control law, which is driven by the translational error dynamics. It also designs a virtual input u to avoid the untraceable attitude in this part. Herein let the difference between the reference (ρ_r, ν_r) and the estimated states (ρ, ν) denote the translational error

$$\begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} = \begin{bmatrix} R_r^T(\rho_r - \rho) \\ \nu_r - R_r^T R\nu \end{bmatrix}. \tag{3.18}$$

Facilitated by general dynamics(2.20), we obtain the translational error dynamics

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{3.19a}$$

$$\dot{\nu}_e = -S(\omega_r)\nu_e + \frac{f}{m}R_r^T R e_3 - \frac{f_r}{m}e_3. \tag{3.19b}$$

Considering that the only controllable thrust magnitude f is unable to affect attitude R, a substitution as a virtual input is constructed

$$u = \frac{f}{m} R_r^T R e_3 - \frac{f_r}{m} e_3, \tag{3.20}$$

which leads to the translational error dynamics

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{3.21a}$$
$$\dot{\nu}_e = -S(\omega_r)\nu_e + u, \tag{3.21b}$$

in closed-loop with observed output feedback

$$u = -\sigma(k_\rho \hat{\rho}_e + k_\nu \hat{\nu}_e), \tag{3.22a}$$
$$\dot{\hat{\rho}}_e = -S(\omega_r)\hat{\rho}_e + \nu_e + L_1 z \tag{3.22b}$$
$$\dot{\hat{\nu}}_e = -S(\omega_r)\hat{\nu}_e + u + L_2 z \tag{3.22c}$$
$$\dot{z} = -S(\omega_r)z - (L_1 + L_3)z + (L_1 + L_3)\tilde{\rho}_e, \tag{3.22d}$$

where $k_\rho > 0, k_\nu > 0$ are the feedback gains; $L_1 > 0$, $L_2 > 0$ and $L_3 > 2L_2/L_3$ are filter parameters; $\tilde{\rho}_e = \rho_e - \hat{\rho}_e$ is the difference between estimated and filtered error; $\sigma(*)$ is a saturation function

$$\sigma(x) = \begin{cases} \gamma \tanh(||x||_2/\gamma)||x||_2^{-1}x, & \text{if } \sigma(x) \neq 0 \\ 0, & x = 0 \\ x, & \text{elsewhere} \end{cases}, \tag{3.23}$$

to avoid reaching the maximum motor ratio, which might trigger instability.

From (3.20), a thrust magnitude f, can be determined:

$$mu + f_r e_3 = f R_r^T R e_3 \tag{3.24a}$$
$$f = \|mu + f_r e_3\|, \tag{3.24b}$$

3.2.2 Attitude control

In the previous section, a stable position tracking was developed, however, depending on a virtual input u. In this section, by introducing *desired attitude dynamics* based on this virtual input, we can update the reference for attitude and finally bridge the position control law with attitude.

Let the desired thrust direction f_d and desired rotation matrix R_d (further details see AppendixA) denote the term $R_r^T R e_3$ and a non-unique rotation matrix in (3.24a)

$$f_d := R_r^T R e_3 = \begin{bmatrix} f_{d1} \\ f_{d2} \\ f_{d3} \end{bmatrix} = \frac{f_r e_3 + mu}{\|f_r e_3 + mu\|} = \frac{f_r e_3 + mu}{f} \tag{3.25}$$

$$R_d := \begin{bmatrix} 1 - \frac{f_{d1}^2}{(1+f_{d3})} & -\frac{f_{d1}f_{d2}}{(1+f_{d3})} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{(1+f_{d3})} & 1 - \frac{f_{d2}^2}{(1+f_{d3})} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3}, \end{bmatrix} \tag{3.26}$$

which derive a desired angular rate

$$\dot{R}_d = R_d S(\omega_d) \tag{3.27a}$$
$$\omega_d = \begin{bmatrix} -\dot{f}_{d2} + \frac{f_{d2}\dot{f}_{d3}}{1+f_{d3}} \\ \dot{f}_{d1} - \frac{f_{d1}\dot{f}_{d3}}{1+f_{d3}} \\ \frac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix}. \tag{3.27b}$$

From (3.25), as R_d and $R_r^T R$ share the same last column f_d, we can require $R_r^T R$ converging to R_d. With known R_d and R_r, R can be simply derived from $R_r R_d$; in other words, we should update the attitude tracking reference as the rotation matrix R_{rd} and angular rate ω_{rd} as

$$R_{rd} = R_r R_d \tag{3.28a}$$
$$\omega_{rd} = R_d^T \omega_r + \omega_d, \tag{3.28b}$$

which renders new reference dynamics

$$\dot{R}_{rd} = R_{rd} S(\omega_{rd}) \tag{3.29a}$$
$$J\dot{\omega}_{rd} = S(J\omega_{rd})\omega_{rd} + \tau_{rd} \tag{3.29b}$$
$$\tau_{rd} = -S(J\omega_{rd})\omega_{rd} + J(R_d S(\omega_d))^T \omega_r + J R_d^T \dot{\omega}_r + J\dot{\omega}_d. \tag{3.29c}$$

For the derivation of these dynamics see AppendixA. Thereafter, drawing into the similarity of position control, the filtered attitude controller can be achieved by the orientational dynamics

$$\dot{R} = R S(\omega) \tag{3.30a}$$
$$J\dot{\omega} = S(J\omega)\omega + \tau \tag{3.30b}$$

with closed-loop input

$$\tau = \tau_{rd} + S(J\hat{\omega}_e)\omega_{rd} + K_\omega \hat{\omega}_e + \sum_{i=1}^{3} k_i S(R_{rd}^T \nu_i)\hat{R}^T \nu_i, \tag{3.31a}$$

$$\dot{\hat{R}} = \hat{R} S(\omega - C_R \sum_{i=1}^{3} k_i S(\hat{R}^T \nu_i)(R_r^T \nu_i + R^T \nu_i)) \tag{3.31b}$$

$$J\dot{\hat{\omega}} = S(J\omega)\omega + \tau - c_\omega J S(\omega_r)\omega_e - c_\omega K_\omega \omega_e - C_\omega \tilde{\omega}, \tag{3.31c}$$

where the errors are defined as $R_e = R_{rd} R^T, \tilde{R} = \hat{R} R, \omega_e = \omega_{rd} - \omega, \hat{\omega}_e = \omega_{rd} - \hat{\omega}$ and $\tilde{\omega} = \hat{\omega} - \omega$; $v_i = e_i$ is the direction vector; $K_\omega = K_\omega^T > 0$ and $k_i > 0$ are the attitude feedback gain; $C_\omega = C_\omega^T > 0, c_R > 0$ and $c_\omega > 0$.

3.2.3 Combined control in quaternions

As illustrated in section 2.1.3, the calculation of unit quaternions is more efficient than the rotation matrix's, which is friendly for a limited CPU of the drone. However, to avoid the ambiguous mapping for the same rotation from either positive or negative versor, the general flight dynamics(2.20) are rearranged (details about quaternions see section 2.1.2) into

$$\dot{\rho} = q \odot \nu \tag{3.32a}$$
$$\dot{\nu} = -S(\omega)\nu_r + g q^* \odot e_3 - (f/m)e_3 \tag{3.32b}$$
$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{3.32c}$$
$$J\dot{\omega} = -S(J\omega)\omega + \tau, \tag{3.32d}$$

and the versor form of the reference dynamics is given by

$$\dot{\rho}_r = q_r \odot \nu_r \tag{3.33a}$$
$$\dot{\nu}_r = -S(\omega_r)\nu_r + g q_r^* \odot e_3 - (f_r/m)e_3 \tag{3.33b}$$
$$\dot{q}_r = \frac{1}{2} q_r \otimes \begin{bmatrix} 0 \\ \omega_r \end{bmatrix} \tag{3.33c}$$
$$J\dot{\omega}_r = -S(J\omega_r)\omega_r + \tau_r, \tag{3.33d}$$

which leads to

$$\begin{bmatrix}\rho_e\\ \nu_e\end{bmatrix} = \begin{bmatrix} q_r^* \odot (\rho_r - \rho)\\ \nu_r - (q_r^* \otimes q) \odot \nu\end{bmatrix}. \tag{3.34}$$

Then building a virtual input $u = -\sigma(k_\rho \hat{\rho}_e + k_\nu \hat{\nu}_e)$, stability is guaranteed for the close loop system

$$\dot{\hat{\rho}}_e = -S(\omega_r)\hat{\rho}_e + \nu_e + L_1 z \tag{3.35a}$$
$$\dot{\hat{\nu}}_e = -S(\omega_r)\hat{\nu}_e + u + L_2 z \tag{3.35b}$$
$$\dot{z} = -S(\omega_r)z - (L_1 + L_3)z + (L_1 + L_3)(\rho_e - \hat{\rho}_e). \tag{3.35c}$$

Corresponding to the desired rotation matrix R_d (3.26), a desired versor q_d can be derived (see Appendix A):

$$\boldsymbol{q}_d = \begin{bmatrix} \sqrt{\frac{(1+f_{d3})}{2}}\\ -\frac{f_{d2}}{\sqrt{2(1+f_{d3})}}\\ \frac{f_{d1}}{\sqrt{2(1+f_{d3})}}\\ 0\end{bmatrix}. \tag{3.36}$$

Updated attitude reference (3.28) may be rewritten to give

$$q_{rd} = q_r \otimes q_d \tag{3.37a}$$
$$\omega_{rd} = q_d^* \odot \omega_r + \omega_d, \tag{3.37b}$$

which satisfy

$$\dot{q}_{rd} = \frac{1}{2} q_{rd} \otimes \begin{bmatrix}0\\ \omega_{rd}\end{bmatrix} \tag{3.38a}$$
$$J\dot{\omega}_{rd} = S(J\omega_{rd})\omega_{rd} + \tau_{rd} \tag{3.38b}$$
$$\tau_{rd} = -S(J\omega_{rd})\omega_{rd} + J\dot{q}_d^* \odot \omega_r + Jq_d^* \odot \dot{\omega}_r + J\dot{\omega}_d. \tag{3.38c}$$

Finally, the overall objective of these equations is to rewrite the full feedback control (3.22) and (3.31) into unit quaternions. The particular shape of related model dynamics has been delivered above. It is possible to formulate the cascade controller for full states using quaternions as

$$u = -\sigma(k_\rho \hat{\rho}_e + k_\nu \hat{\nu}_e), \tag{3.39a}$$
$$\dot{\hat{\rho}}_e = -S(\omega_r)\hat{\rho}_e + \nu_e + L_1 z \tag{3.39b}$$
$$\dot{\hat{\nu}}_e = -S(\omega_r)\hat{\nu}_e + u + L_2 z \tag{3.39c}$$
$$\dot{z} = -S(\omega_r)z - (L_1 + L_3)z + (L_1 + L_3)\tilde{\rho}_e \tag{3.39d}$$
$$\tau = \tau_{rd} + S(J\hat{\omega}_e)\omega_{rd} + K_\omega \hat{\omega}_e + \sum_{i=1}^{3} k_i S(q_{rd}^* \odot \nu_i)\hat{q}^* \odot \nu_i \tag{3.39e}$$
$$\dot{\hat{q}} = \frac{1}{2}\hat{q} \otimes \begin{bmatrix}0\\ \omega + \delta_R\end{bmatrix} \tag{3.39f}$$
$$J\dot{\hat{\omega}} = S(J\omega)\omega + \tau + \delta_\omega, \tag{3.39g}$$

where

$$\delta_R = -c_R \sum_{i=1}^{3} k_i S(\hat{q}^* \odot \nu_i)[(q_{rd} + q)^{-1} \odot \nu_i] \tag{3.40a}$$
$$\delta_\omega = -c_\omega J S(\omega_{rd})\omega_e - c_\omega K_\omega \omega_e - C_\omega \tilde{\omega} \tag{3.40b}$$

3.3 Concluding remarks

As stated above, this chapter derives a smooth trajectory based on flat-output as a reference for controller validation. Following those reference states, the position tracking errors build a virtual input, which relates desired dynamics to update the attitude reference. Facilitated by this virtual input, we obtain a full state controller for both position and attitude and further rewrite it in versor form for quicker computation.

Chapter 4

Flying Parrot mambo drone

Before implementing the newly designed controller by [26] on the set-up, it is necessary to simulate this controller by the Aerospace Toolbox in Matlab. This toolbox allows customers to evaluate vehicle motion and orientation using built-in aerospace math operations and coordinate system and spatial transformations [31]. It also provides a platform for Parrot's specific Support Package based on the Aerospace Toolbox, to design a customized controller and access those sensors' navigation data [30]; thereafter, the host-computer can remotely deploy the controller on the set-up to track demanding references [30]. The insight of the default estimator and controller would provide clues for gains tuning of the controller by [26] in the next chapter.

4.1 Experimental Setup

To verify the previous theoretical design in Chapter 3, it is crucial to understand firmware and obtaining its parameters. The experimental setup contains a Parrot Mambo drone and a host computer with a Parrot Simulink package on Matlab (2019a version).

4.1.1 Aerial vehicle

Given by its official website and Brekelmans's work [7], Parrot mambo drone is a lightweight mini-drone with size 0.18×0.18 meter, whose weight is only 0.068 kilogram and the inertia is $\text{diag}[0.069, 0.0775, 0.150] \cdot 10^{-3}\text{kg}\cdot\text{m}^2$ (including four removable propeller bumpers as protection) designed for easy handling. As a classic quadrotor, it contains four fixed propellers to generate thrust and the carried battery supports a 10 minutes flight.

4.1.2 On-board sensors

The on-board sensors consist of a vertical camera, an inertial measurement unit(IMU), and an ultrasound sensor, which are detailed below.

Vertical camera

The vertical camera is used to measure optical flow for horizontal speed. Its lens is 80 degrees processing a VGA resolution (640x480p) signal at 60 frames per second. This camera can also be used to detect object characters, such as color, for further absolute position recognition.

IMU

The IMU includes a 3-axis accelerometer and a 3-axis gyroscope running at 200 Hz. The accelerometer provides body-fixed accelerations on xyz directions and the gyroscope measures Euler angle rates following ZYX order.

Ultrasound sensor

The ultrasound sensor is used to estimate the altitude and vertical velocity in the inertial frame. It provides a "distance" between the drone and any event as a relative quantity [37]. To avoid the attitude suddenly varying, the terrain should be horizontal without obstacles.

By obtaining the data from the above sensors, the real-time system embedded on the drone estimates states and controls flight simultaneously at a 200 Hz rate [8].

4.2 Parrot Simulink package

This section focuses on the autonomous flying kernel of the Parrot Simulink package(PSP): control strategy. It consists of two default modules: a state estimator and a Proportional Integral Derivative controller. As a decisive role, the state estimator promises timely accurate states for the controller to compute the desired force and torque within those states for the rotor commands.

4.2.1 Default state estimator

The structure of the default state estimator is shown in Figure 4.1. The system combines integrator and several predictive filters based on a flight dynamics model to fuse data, predict states and denoise [37]. It starts from extracting and tracking the sensor measurements(see Section 4.1.2): linear accelerations, angular rates, height, and a scaled horizontal motion. All features are initialized by necessary calibration data obtained in the steady set-off stage, and IMU data are filtered in sensor preprocessing.

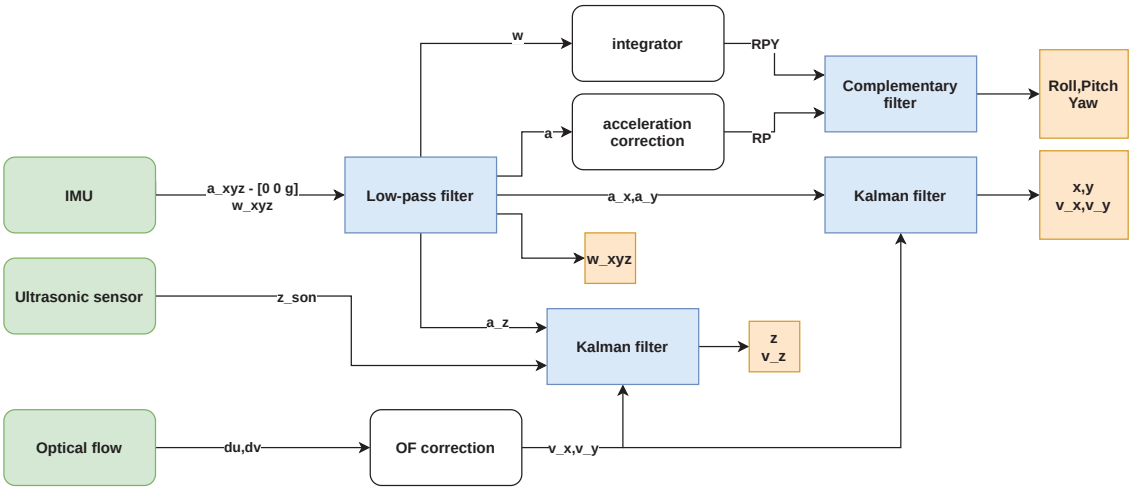


Figure 4.1: Schema of the default estimator in Parrot Simulink package

Before detailing the estimation processing, we denote $(*)$ as a measured raw feature or a certain final estimated state, $(\hat{*})$ as an estimation of state, and $(\hat{*})'$ as a prior estimation of $(\hat{*})$. And as in Chapter 3, R is the rotation matrix from body-fixed to the inertial frame; T_s is the sample time of sensors.

IMU based attitude estimation

As the outcome of gravity and flight dynamics, IMU measurements are affected by accelerator bias a_b, gyroscope bias ω_b, gravity tensor $G_b = [0, 0, g]^T$ and additive random noises a_a, n_ω [35]. The measured raw linear acceleration $A = [A_1, A_2, A_3]^T$ and body-fixed angular rate $\Omega = [\Omega_1, \Omega_2, \Omega_3]^T$ are given by

$$A = a_{real} + R^T G_b + a_b + n_a \tag{4.1a}$$

$$\Omega = \omega_{real} + \omega_b + n_\omega. \tag{4.1b}$$

Assuming that the additive random noises are Gaussian distributed and the IMU biases are random walk with Gaussian distributed deviates [35]. In such a case, the measurements are preprocessed by FIR filter and IIR filter

$$f_{FIR}(A(Z)) := a(Z) = \sum_{k=0}^{5} h(k) Z^{-k} A(Z) \tag{4.2a}$$

$$f_{IIR}(\Omega_3(Z)) := \omega_3(Z) = \frac{\sum_{k=0}^{5} b(k) Z^{-k}}{\sum_{k=0}^{5} a(k) Z^{-k}} \Omega_3(Z), \tag{4.2b}$$

where $*(Z)$ denotes the signals in Z-domain; $h(k) = [0.026, 0.14, 0.33, 0.33, 0.14, 0.026]$ are the coefficients of the FIR filter; $a(k) = [0.28, 1.27, 2.42, 2.42, 1.27, 0.28]$, $b(k) = [1, 2.23, 2.52, 1.58, 0.54, 0.08]$ are the coefficients of the IIR filter. Herein, the filtered acceleration and body-fixed angular rate are presented as

$$a = [a_1, a_2, a_3]^T \tag{4.3}$$

$$\omega = [\Omega_1, \Omega_2, \omega_3]^T. \tag{4.4}$$

To estimate the Euler angles, we use integrator for consecutive frames of the Euler rate $[\dot{\phi}_{gyr}, \dot{\theta}_{gyr}, \dot{\psi}_{gyr}]^T$ with zero initial condition, which can be transformed from the body-fixed angular rate as

$$\begin{bmatrix} \dot{\phi}_{gyr} \\ \dot{\theta}_{gyr} \\ \dot{\psi}_{gyr} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = W_{be}\omega \tag{4.5a}$$

$$\begin{bmatrix} \phi_{gyr} \\ \theta_{gyr} \\ \psi_{gyr} \end{bmatrix} = \int W_{be}\omega \, dt. \tag{4.5b}$$

The mapping relationship is given from [16] [10]

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} \dot{\phi}_{gyr} \\ 0 \\ 0 \end{bmatrix} + R(\phi) \begin{bmatrix} 0 \\ \dot{\theta}_{gyr} \\ 0 \end{bmatrix} + R(\phi)R(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_{gyr} \end{bmatrix}. \tag{4.6}$$

Note that, when the drone reaches a steady situation, an additional Roll and Pitch angle correction with acceleration are introduced

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = R^T \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = g \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{4.7a}$$

$$\theta_a = \arcsin -\frac{a_1}{g} \tag{4.7b}$$

$$\phi_a = \arctan \frac{a_2}{a_3};, \tag{4.7c}$$

where R^T is the transfer matrix from inertial frame to body frame following ZYX order

$$R^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Consequently, the attitude estimation algorithm is

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{cases} 0.999 \begin{bmatrix} \phi_{gyr} \\ \theta_{gyr} \\ \psi_{gyr} \end{bmatrix} + 0.001 \begin{bmatrix} \phi_a \\ \theta_a \\ 0 \end{bmatrix}, \text{if } (1+0.02) * g > \sqrt{\sum_i^3 a_i^2} > (1-0.02) * g \\ \begin{bmatrix} \phi_{gyr} \\ \theta_{gyr} \\ \psi_{gyr} \end{bmatrix}, \text{else} \end{cases} . \tag{4.9a}$$

Ultrasonic sensor based height and velocity

Due to the noisy altitude signal from the ultrasonic sensor, the default operation of the vertical motion estimator implements a Kalman filter with a state dynamics model

$$\begin{bmatrix} \dot{\hat{z}}' \\ \dot{\hat{v}}'_{zI} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{z}' \\ \hat{v}'_{zI} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ddot{z} \tag{4.10a}$$

$$z_{sonar} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{z} \\ \hat{v}_{zI} \end{bmatrix} \tag{4.10b}$$

where z_{sonar} is the raw altitude data from the ultrasonic sensor; $\hat{v}_{zI}$ is the prior estimated z-axis velocity in the **inertial frame**; $\ddot{z}$ is the z-axis acceleration in the **inertial frame**, which compromises the body-fixed acceleration a as an input to adapt the vertical linear velocity. That is

$$\begin{aligned} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} &= R \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ &= \begin{bmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\phi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \end{aligned} \tag{4.11a}$$

$$\ddot{z} = -a_1\sin\theta + a_2\cos\theta\sin\psi + a_3\cos\theta\cos\psi + g. \tag{4.11b}$$

This results in the final estimated altitude and vertical linear velocity

$$\begin{bmatrix} \hat{z} \\ \hat{v}_{zI} \end{bmatrix} = \begin{bmatrix} \hat{z}' \\ \hat{v}'_{zI} \end{bmatrix} + \begin{bmatrix} L_{zc} \\ 0 \end{bmatrix} (z_{sonar} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{z}' \\ \hat{v}'_{zI} \end{bmatrix}) \tag{4.12a}$$

where L_{zc} is the continuous constant Kalman gain. Note that the required linear velocity should be in the body-fixed frame, therefore $\hat{v}_{zI}$ can be written as;

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = R^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \hat{v}_{zI} \end{bmatrix}, \tag{4.13}$$

where $\dot{x}, \dot{y}$ are the horizontal linear velocities in the **inertial frame** given by the horizontal movement estimator (see next section).

Optical flow based horizontal position and velocity

The main role of the optical flow sensor is to reflect the vehicle motion on the image plane. The horizontal speed estimator from the image relates two interconnected algorithms [8]. Referring to Parrot Patents [15] [1], the first one is discrete-time digital processing to determine an optical flow field between each pair of successive images, which can be decomposed into image pyramids. As the distribution of the movement of brightness pattern [19], optical flow field provides motion of each pixel, then this algorithm combined Lucas-Kanade estimation method [27] and pyramid of images estimate [1] [12] a corresponding maximum detectable pixel displacement as a constant optical flow (du, dv) in a focal plane. However, from the output of firmware, the magnitude of optical flow data is 10^{-3} unmatched the possible displacement. Due to the inaccessible firmware, we assume the optical flow output is refined with the scale of focal length and pixel size experimentally validated in Chapter 5. For the same reason of the inaccessibility, the further detail of the optical flow detecting algorithm is needless to mention in this report.

Within the detected optical flow (du, dv), the second algorithm revealed in the Parrot Simulink Package utilizes a compensation gain and the altitude z for frame change and scale change between the image plane and the scene [12] to achieve the horizontal body-fixed linear velocity:

$$\dot{x} = -1.15z * du \tag{4.14a}$$
$$\dot{y} = -1.15z * dv. \tag{4.14b}$$

Additionally, because of the low-cost accelerator, the noisy acceleration is insufficient to be integrated twice for position [12] but introduced to filter linear velocities(4.14) with a state-space model

$$\begin{bmatrix} \dot{\hat{v}}'_x \\ \dot{\hat{v}}'_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{4.15a}$$
$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix} \tag{4.15b}$$

which leads to

$$\begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix} = \begin{bmatrix} \hat{v}'_x \\ \hat{v}'_y \end{bmatrix} + \begin{bmatrix} L_{xy} & 0 \\ 0 & L_{xy} \end{bmatrix} \left(\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{v}'_x \\ \hat{v}'_y \end{bmatrix} \right). \tag{4.16}$$

where L_{xy} is a constant Kalman gain.

Finally, the horizontal localization of the vehicle is derived from integrated horizontal velocities

$$\begin{bmatrix} x \\ y \end{bmatrix} = \int \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{v}_z \end{bmatrix} dt \tag{4.17}$$

where vertical velocity $\hat{v}_z$ is given by (4.13).

4.2.2 Default Proportional Integral Derivative controller

As was presented in section 4.2.1, the default state estimator evaluates and filters position, Euler angles, linear velocity, and angular rate from sensor data. The default controller, a Proportional Integral Derivative(PID) controller, receives said states and acquires desired references [36]. Such a PID controller promises good robustness and responses in a short time with a slight overshoot and nearly zero steady-state error [41]. Note that the Derivative module of the control strategy obscurely assumes all reference states except position are zero, which only performs for hover

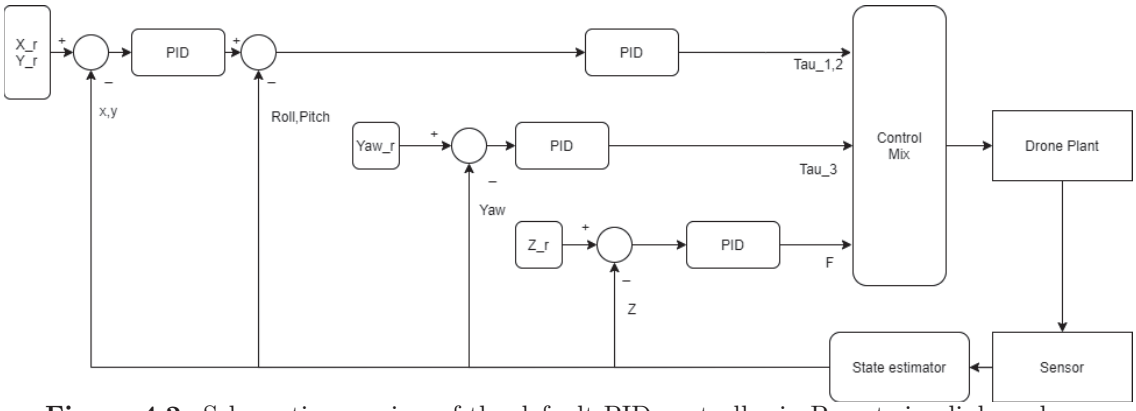


Figure 4.2: Schematic overview of the default PID controller in Parrot simulink package

condition to avoid the unstable behavior from its imprecise model. Further description of this controller is presented below.

The general PID algorithm separates controllers for the force scale f and torque vector τ corresponding to the altitude and attitude. In the first circuit, an prior PD controller of horizontal position error would adjust Roll and Pitch angle references, where the tilting vehicle obtains the horizontal component of thrust to correct the location. That is

$$\begin{bmatrix}\theta_r\\ \phi_r\end{bmatrix} = P_{eul}\mathrm{Sat}(R(\psi)\begin{bmatrix}x_r - x\\ y_r - y\end{bmatrix}) + D_{eul}\begin{bmatrix}v_1\\ v_2\end{bmatrix}, \tag{4.18}$$

where $P_{eul} = \begin{bmatrix}-0.24 & 0\\ 0 & 0.24\end{bmatrix}$ and $D_{eul} = \begin{bmatrix}0.1 & 0\\ 0 & -0.1\end{bmatrix}$; $\mathrm{Sat}(*)$ is a saturation function

$$\mathrm{Sat}(x) = \begin{cases} 3, & \text{if } x > 3\\ -3, & \text{if } x < 3\\ x, & \text{elsewhere;}\end{cases} \tag{4.19}$$

$R(\psi)$ is the rotation matrix

$$R(\psi) = \begin{bmatrix}\cos\psi & \sin\psi\\ -\sin\psi & \cos\psi\end{bmatrix} = \begin{bmatrix}1 & \psi\\ -\psi & 1\end{bmatrix} + h.o.t \tag{4.20}$$

from the inertial frame to the body-fixed frame.

Referring to the updated roll and pitch angles references (4.18), their corresponding components of the torque are designed by a PID controller(Further detail see Figure.B.1)

$$\begin{bmatrix}\dot{\theta}_e\\ \dot{\phi}_e\end{bmatrix} = \begin{bmatrix}\theta_r\\ \phi_r\end{bmatrix} - \begin{bmatrix}\theta\\ \phi\end{bmatrix} \tag{4.21a}$$

$$\begin{bmatrix}\tau_2\\ \tau_1\end{bmatrix} = P_{\tau 12}\left(\begin{bmatrix}\theta_r\\ \phi_r\end{bmatrix} - \begin{bmatrix}\theta\\ \phi\end{bmatrix}\right) + I_\tau\begin{bmatrix}\theta_e\\ \phi_e\end{bmatrix} - D_{\tau 12}\begin{bmatrix}\omega_2\\ \omega_1\end{bmatrix}, \tag{4.21b}$$

where $P_{\tau 12} = \begin{bmatrix}0.013 & 0\\ 0 & 0.01\end{bmatrix}$, $I_{\tau 12} = 0.01$ and $D_{\tau 12} = \begin{bmatrix}0.002 & 0\\ 0 & 0.003\end{bmatrix}$.

Besides the horizontal position controller above, the remainder controllers for the altitude and yaw angle corresponding to force scale f and third component of the torque τ can be written as

$$f + f_r = P_z(z_r - z) - D_z v_3 \tag{4.22a}$$

$$\tau_3 = -P_{\tau 3}\psi - D_{\tau 3}\omega_3, \tag{4.22b}$$

where $f_r = mg$; $P_z = 0.8$ and $D_z = 0.5$; $P_{\tau 3} = 0.004$ and $D_{\tau 3} = 0.00012$.

To be convenient, substitute (4.18) into (4.21), where the high order term of $R(\psi)$ can be ignored under hover condition, a full default controller is derived

$$f + mg = 0.8(z_r - z) - 0.5v_3 \tag{4.23a}$$
$$\phi_r = 0.24\text{Sat}[-\psi(x_r - x) + (y_r - y)] - 0.1v_2 \tag{4.23b}$$
$$\theta_r = -0.24\text{Sat}[(x_r - x) + \psi(y_r - y)] + 0.1v_1 \tag{4.23c}$$
$$\begin{bmatrix}\dot{\phi}_e \\ \dot{\theta}_e\end{bmatrix} = \begin{bmatrix}\phi_r \\ \theta_r\end{bmatrix} - \begin{bmatrix}\phi \\ \theta\end{bmatrix} \tag{4.23d}$$
$$\tau_1 = 0.01(\phi_r - \phi) + 0.01\phi_e - 0.003\omega_1 \tag{4.23e}$$
$$\tau_2 = 0.013(\theta_r - \theta) + 0.01\theta_e - 0.002\omega_2 \tag{4.23f}$$
$$\tau_3 = -0.004\psi - 0.00012\omega_3, \tag{4.23g}$$

4.3 Concluding remarks

This chapter relates to a brief introduction of setup and the derivations of default estimator and controller in Parrot Simulink package. As a sophisticated estimator, it determines and filters required states for the later controller. Even though such an unmodeled system limits the default controller at the hover point, it can be a fundamental component to perform better for the controller by [26] in the next chapter [39].

Chapter 5

Simulation and Implementation

In this chapter, the controller theoretically introduced in Chapter 3 is numerically simulated and experimentally validated. Due to the complex nonlinear behavior, as mentioned in Chapter 4, the default PID controller is an ideal fundamental subject to mimic. Its linearization and approximation for gains tuning of [26] can be found in section 5.1. Next, a comparison between simulations and experiments is delivered and analyzed in section 5.2. Section 5.3 states the optical flow theory for improving the performance, estimates the horizontal velocity from optical flow with a better model and presents the new estimator validation and flight results.

5.1 Approximation of input-output linearization for feedback tuning

Even though the constraints of feedback gains based on Lyapunov theory [26] are proposed to ensure a stable flight processing, its experiments still take a high risk with undesired instability. In reality, breeze, imbalanced rotors, or any other turbulence would limit the ideal stability region. However, for such a nonlinear system with six feedback gains, determining the theoretical robust constraints is challenging. For many sophisticated control strategies, the lowest level is designed with PID control, which can be viewed as the "bread and butter" of control engineering [3]. Therefore, as mentioned in section 4.3, the default PID controller is an appropriate entry point for the controller in [26] to resemble.

Because a linear control system is much simpler than its nonlinear form [23], this section starts by linearizing the default PID controller and the paper's controller [26] separately around the equilibrium hover point. Then given by the same inputs and outputs, the two linearized controllers are comparable, which allows the paper's controller to find equivalent gains. Finally, via simulations, a set of well-performed gains are settled.

For the controller, we define

- Input space: $u := [x, y, z, v_1, v_2, v_3, \phi, \theta, \psi, \omega_1, \omega_2, \omega_3]^T$
- State space: $x := [\theta_e, \phi_e]^T$
- Output space: $y := [f, \tau_1, \tau_2, \tau_3]^T$
- Parameter: m: vehicle mass; J: inertial tensor; x_r, y_r, z_r: constant hover position reference; $R_r = I$, $\omega_r = [0, 0, 0]^T$: attitude references; $f_r = mg$: constant thrust reference equal to gravity; $[v_1, v_2, v_3] = I$: the direction vector.

Linearizing a controller around hover

$$u^T = [x_r, y_r, z_r, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T \tag{5.1a}$$
$$x^T = [0, 0]^T \tag{5.1b}$$
$$y^T = [mg, 0, 0, 0]^T, \tag{5.1c}$$

requires us to consider the variables

$$\bar{u} := u - u^T = [x - x_r, y - y_r, z - z_r, v_1, v_2, v_3, \phi, \theta, \psi, \omega_1, \omega_2, \omega_3]^T \tag{5.2a}$$
$$\bar{x} := x - x^T = [\theta_e, \phi_e]^T \tag{5.2b}$$
$$\bar{y} := y - y^T = [f - mg, 0, 0, 0]^T, \tag{5.2c}$$

Specifically, we denote $(*)$ as an original state, $(*)_L$ as the linearized form of $(*)$, $(*)_{,i}$ as the i^{th} element of a vector $(*)$, $(*)_{i,j}$ as the element on the i^{th} row and the j^{th} column of a matrix $(*)$. Furthermore, ρ is a vector of $[x, y, z]^T$, ν is a vector of $[v_1, v_2, v_3]^T$, ω is a vector of $[\omega_1, \omega_2, \omega_3]^T$ and R is the rotation matrix from body-fixed to the inertial frame from Euler angles, which is

$$R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{5.3}$$

5.1.1 Linearized full controller [26] without observer

Before going to linearize the system [26], since the outputs from the default estimator are filtered (see section 4.2.1), the default PID does not contain any observer inside. For comparability, the full controller [26] should also be simplified without observer around hover point as

$$\begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} = \begin{bmatrix} R_r^T(\rho_r - \rho) \\ \nu_r - R_r^T R\nu \end{bmatrix} = \begin{bmatrix} -\rho \\ -R\nu \end{bmatrix} \tag{5.4a}$$
$$u_{vir} = -(k_\rho \rho_e + k_\nu \nu_e) \tag{5.4b}$$
$$f = \|m u_{vir} + f_r e_3\| \tag{5.4c}$$
$$f_d := \frac{f_r e_3 + m u_{vir}}{\|f_r e_3 + m u_{vir}\|} \tag{5.4d}$$
$$R_d := \begin{bmatrix} 1 - \frac{f_{d1}^2}{(1+f_{d3})} & -\frac{f_{d1} f_{d2}}{(1+f_{d3})} & f_{d1} \\ -\frac{f_{d1} f_{d2}}{(1+f_{d3})} & 1 - \frac{f_{d2}^2}{(1+f_{d3})} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix} \tag{5.4e}$$
$$\omega_d := \begin{bmatrix} -\dot{f}_{d2} + \frac{f_{d2}\dot{f}_{d3}}{1+f_{d3}} \\ \dot{f}_{d1} - \frac{f_{d1}\dot{f}_{d3}}{1+f_{d3}} \\ \frac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix} \tag{5.4f}$$
$$R_{rd} = R_r R_d = R_d \tag{5.4g}$$
$$\omega_{rd} = R_d^T \omega_r + \omega_d = \omega_d \tag{5.4h}$$
$$\tau_{rd} = -S(J\omega_{rd})\omega_{rd} + J(R_d S(\omega_d))^T \omega_r + J R_d^T \dot{\omega}_r + J\dot{\omega}_d = -S(J\omega_d)\omega_d + J\dot{\omega}_d \tag{5.4i}$$
$$\tau = \tau_{rd} + S(J\omega_e)\omega_{rd} + K_\omega \omega_e + \textstyle\sum_{i=1}^3 k_i S(R_{rd}^T \nu_i) R^T \nu_i = \tau_{rd} + S(J(\omega_d - \omega))\omega_d + K_\omega(\omega_d - \omega) + \sum_{i=1}^3 k_i S(R_d^T \nu_i) R^T \nu_i, \tag{5.4j}$$

where u_{vir} is the virtual input without saturation function, because its small variance under the hover condition settles in the linear range of saturation function.

Following the general procedure [23] [9], differentiating the position control (5.4a)-(5.4c) with respect to input u gives

$$\rho_{eL} = \rho_e(0) + \frac{\partial \rho_e}{\partial u}\bar{u} = -\begin{bmatrix}\bar{u}_1 & \bar{u}_2 & \bar{u}_3\end{bmatrix}^T \tag{5.5a}$$

$$\nu_{eL} = \nu_e(0) + \frac{\partial \nu_e}{\partial u}\bar{u} = -\begin{bmatrix}\bar{u}_4 & \bar{u}_5 & \bar{u}_6\end{bmatrix}^T \tag{5.5b}$$

$$u_{virL} = -(k_\rho \rho_{eL} + k_\nu \nu_{eL}) \tag{5.5c}$$

$$f_L = f(0) + \frac{\partial f}{\partial u}\bar{u} = mg + \frac{(f_r e_3)^T}{\|f_r e_3\|} m u_{virL} = mg - [0,0,m](k_\rho \rho_{eL} + k_\nu \nu_{eL}) \tag{5.5d}$$

$$\bar{f}_L = f_L - f_r = -[0,0,m](k_\rho \rho_{eL} + k_\nu \nu_{eL}). \tag{5.5e}$$

Subsequently, the designed parameters (5.4d)-(5.4f) can be linearized as

$$f_{dL} = f_d(0) + \frac{\partial f_d}{\partial u}\bar{u} = \begin{bmatrix}0\\0\\1\end{bmatrix} - \frac{1}{g}\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}(k_\rho \rho_{eL} + k_\nu \nu_{eL}) \tag{5.6a}$$

$$R_{dL} = R_d(0) + \frac{\partial R_d}{\partial u}\bar{u} = \begin{bmatrix} 1 & 0 & -\dfrac{k_\rho \rho_{eL,1} + k_\nu \nu_{eL,1}}{g} \\ 0 & 1 & -\dfrac{k_\rho \rho_{eL,2} + k_\nu \nu_{eL,2}}{g} \\ \dfrac{k_\rho \rho_{eL,1} + k_\nu \nu_{eL,1}}{g} & \dfrac{k_\rho \rho_{eL,2} + k_\nu \nu_{eL,2}}{g} & 1 \end{bmatrix} \tag{5.6b}$$

$$\omega_{dL} = \omega_d(0) + \frac{\partial \omega_d}{\partial u}\bar{u} = -\frac{1}{g}\begin{bmatrix}-k_\rho k_\nu \rho_{eL,2} + (k_\rho - k_\nu^2)\nu_{eL,2}\\ k_\rho k_\nu \rho_{eL,1} - (k_\rho - k_\nu^2)\nu_{eL,1}\\ 0\end{bmatrix}. \tag{5.6c}$$

which leads to

$$\tau_{rdL} = J\frac{\partial \dot{\omega}_d}{\partial u}\bar{u} = -\frac{J}{g}\begin{bmatrix}-k_\rho k_\nu \dot{\rho}_{eL,2} + (k_\rho - k_\nu^2)\dot{\nu}_{eL,2}\\ k_\rho k_\nu \dot{\rho}_{eL,1} - (k_\rho - k_\nu^2)\dot{\nu}_{eL,1}\\ 0\end{bmatrix} \tag{5.7a}$$

$$\begin{aligned}
\tau_L &= \tau_L(0) + \tau_{rdL} + K_\omega(\omega_{dL} - \omega_L) + \frac{\partial}{\partial u}(\sum_{i=1}^{3} k_i S(R_d^T \nu_i) R^T \nu_i \\
&= \tau_{rdL} + K_\omega(\omega_{dL} - \omega) - \sum_{i=1}^{3} k_i S(v_i) S(v_i) \begin{bmatrix}R_{dL,2,3} + R_{L,2,3}\\ R_{dL,3,1} + R_{L,3,1}\\ R_{L,1,2}\end{bmatrix} \\
&= \tau_{rdL} + K_\omega(\omega_{dL} - \omega) - \begin{bmatrix}(k_3 + k_2) & 0 & 0\\ 0 & (k_3 + k_1) & 0\\ 0 & 0 & (k_2 + k_1)\end{bmatrix}\begin{bmatrix}R_{dL,2,3} + R_{L,2,3}\\ R_{dL,3,1} + R_{L,3,1}\\ R_{L,1,2}\end{bmatrix} \\
&= -\frac{J}{g}\begin{bmatrix}-k_\rho k_\nu \dot{\rho}_{eL,2} + (k_\rho - k_\nu^2)\dot{\nu}_{eL,2}\\ k_\rho k_\nu \dot{\rho}_{eL,1} - (k_\rho - k_\nu^2)\dot{\nu}_{eL,1}\\ 0\end{bmatrix} - \frac{K_\omega}{g}\begin{bmatrix}-k_\rho k_\nu \rho_{eL,2} + (k_\rho - k_\nu^2)\nu_{eL,2}\\ k_\rho k_\nu \rho_{eL,1} - (k_\rho - k_\nu^2)\nu_{eL,1}\\ 0\end{bmatrix} - K_p\begin{bmatrix}R_{dL,2,3} + \phi\\ R_{dL,3,1} + \theta\\ \psi\end{bmatrix} - K_\omega \omega,
\end{aligned} \tag{5.7b}$$

where R_L is linearized from (5.3) around $\phi = \theta = \psi = 0$ as

$$\begin{aligned}
R_L &= R_0 + \frac{\partial R}{\partial u}\bar{u} \\
&= I + \begin{bmatrix}0 & -\psi & 0\\ \psi & 0 & 0\\ 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & \theta\\ 0 & 0 & 0\\ -\theta & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -\phi\\ 0 & \phi & 0\end{bmatrix} + h.o.t = \begin{bmatrix}1 & -\psi & \theta\\ \psi & 1 & -\phi\\ -\theta & \phi & 1\end{bmatrix} + h.o.t.
\end{aligned} \tag{5.8}$$

Consequently, substituting (5.2) into above derivation of linearized full controller, its linear outputs are achieved as

$$\bar{f} = mk_\rho \bar{u}_3 + mk_\nu \bar{u}_6 \tag{5.9a}$$

$$\bar{\tau} = \frac{k_\rho[(k_\rho - k_\nu^2)J - K_\omega k_\nu - gK_p]}{g}\begin{bmatrix} -\bar{u}_2 \\ \bar{u}_1 \\ 0 \end{bmatrix} + \frac{k_\nu(2k_\rho - k_\nu^2)J - K_\omega(k_\rho - k_\nu^2) - gK_p k_\nu}{g}\begin{bmatrix} -\bar{u}_5 \\ \bar{u}_4 \\ 0 \end{bmatrix} - K_p \begin{bmatrix} \bar{u}_7 \\ \bar{u}_8 \\ \bar{u}_9 \end{bmatrix} - K_\omega \begin{bmatrix} \bar{u}_{10} \\ \bar{u}_{11} \\ \bar{u}_{12} \end{bmatrix}. \tag{5.9b}$$

5.1.2 Linearized default controller

Following the same linearization procedure as above, we can differentiate the default altitude controller(4.23a) as

$$\bar{f} = 0.8\bar{u}_3 + 0.5\bar{u}_6. \tag{5.10}$$

Given by the hover horizontal reference around initial point $[x_r, y_r] = [0, 0]$ and the steady indoor environment, (4.23b)-(4.23c) would not trigger the saturation function, i.e.

$$\phi_r = 0.24[-\psi(x_r - x) + (y_r - y)] - 0.1v_2 \tag{5.11a}$$

$$\theta_r = -0.24[(x_r - x) + \psi(y_r - y)] + 0.1v_1, \tag{5.11b}$$

which can be linearized as

$$\theta_{rL} = -0.24xy - 0.1v_2 \tag{5.12a}$$

$$\phi_{rL} = 0.24x + 0.1v_1. \tag{5.12b}$$

Therefore, the linearized default attitude controller (4.23d)-(4.23g) yields

$$\dot{\bar{x}} = \begin{bmatrix} \dot{\phi}_{eL} \\ \dot{\theta}_{eL} \end{bmatrix} = \begin{bmatrix} -0.24\bar{u}_2 - 0.1\bar{u}_5 - \bar{u}_7 \\ 0.24\bar{u}_1 + 0.1\bar{u}_4 - \bar{u}_8 \end{bmatrix} \tag{5.13a}$$

$$\bar{\tau} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \\ 0 & 0 \end{bmatrix}\bar{x} + \begin{bmatrix} -0.01 & 0 & 0 \\ 0 & -0.013 & 0 \\ 0 & 0 & -0.004 \end{bmatrix}\begin{bmatrix} \bar{u}_7 \\ \bar{u}_8 \\ \bar{u}_9 \end{bmatrix} + \begin{bmatrix} -0.003 & 0 & 0 \\ 0 & -0.002 & 0 \\ 0 & 0 & -0.00012 \end{bmatrix}\begin{bmatrix} \bar{u}_{10} \\ \bar{u}_{11} \\ \bar{u}_{12} \end{bmatrix}. \tag{5.13b}$$

5.1.3 Comparison and approximation

For an appropriate choice of feedback gains in the controller by [26], section 5.1.1 and 5.1.2 show two relatively similar linear control laws. In each case, there is the same state error stabilizing system, where the altitude feedback gains can be derived from the comparison of (5.9a) and (5.10) as

$$k_\rho = \frac{0.8}{m} \qquad\qquad k_\nu = \frac{0.5}{m}. \tag{5.14}$$

However, because the default controller neglects the flight dynamics but an error state-space model (5.13a) for integral instead, it is clear that (5.9b) and (5.13b) cannot match exactly. Due to the well-corresponding force control laws of (5.9a) and (5.10), both torque controls are supposed to ignore the overlapped part about transitional states. We obtain

$$K_p = \begin{bmatrix} (k_3 + k_2) & 0 & 0 \\ 0 & (k_3 + k_1) & 0 \\ 0 & 0 & (k_2 + k_1) \end{bmatrix} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.013 & 0 \\ 0 & 0 & 0.004 \end{bmatrix} \tag{5.15a}$$

$$K_\omega = \begin{bmatrix} 0.003 & 0 & 0 \\ 0 & 0.002 & 0 \\ 0 & 0 & 0.00012 \end{bmatrix} \tag{5.15b}$$

As expected, the approximated feedback gains still satisfy the stability constraints from Lyapunov theory posed in section 3.2 as

$$k_\rho = \frac{0.8}{m} > 0, \tag{5.16a}$$

$$k_\nu = \frac{0.5}{m} > 0 \tag{5.16b}$$

$$K_\omega = \begin{bmatrix} 0.003 & 0 & 0 \\ 0 & 0.002 & 0 \\ 0 & 0 & 0.00012 \end{bmatrix} = K_\omega^T > 0 \tag{5.16c}$$

$$k_1 = 0.0035 > 0 \tag{5.16d}$$

$$k_2 = 0.0005 > 0 \tag{5.16e}$$

$$k_3 = 0.0095 > 0. \tag{5.16f}$$

5.2 Undesired performance with hover reference and causes analysis

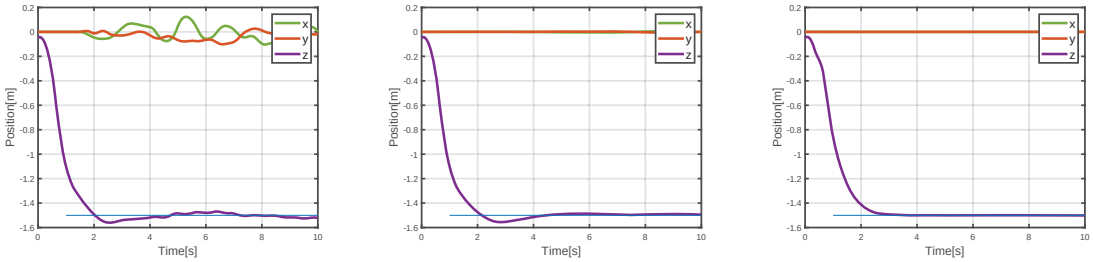
The PID-based gain tuning for the paper's controller, which has been discussed in the previous section, should sufficiently prove stability when operating on load. But from the non-observer simulations with default estimator, the analogical K_ω still triggers oscillation until introducing the vehicle initial tensor J(see Figure.5.1a5.1b). Finally, within a simple design of experiment(DOE) for none steady-state error, smallest overshoot and settle time with Minitab and Simulink(see Figure.5.1c), we obtain a set of well-performed feedback and observer gains as

$$k_\rho = \frac{0.8}{m} \qquad k_\nu = \frac{0.5}{m} \qquad K_\omega = 30J \tag{5.17a}$$

$$k_1 = 0.0035 \qquad k_2 = 0.0005 \qquad k_3 = 0.0095 \tag{5.17b}$$

$$L_1 = \frac{1}{m} \qquad L_1 = \frac{10}{m} \qquad L_1 = \frac{2L_2}{L_1} + 1 \tag{5.17c}$$

$$c_R = 0.0001 \qquad c_\omega = 0.001 \qquad C_R = 30J \tag{5.17d}$$



(a) **(b)** **(c)**

Figure 5.1: The x, y, z position of hover simulations. (a)The results with analogical gains; (b) The results with re-tuned gains introducing the vehicle tensor flow J; (c) The results with with final settled gains (including feedback and observer gains)

Unfortunately, the hover experimental results of these gains are undesired: once the drone reaches a certain height, a dangerous circling around the origin starts, even flipping over. During the experiments, all hand-tuning tries failed to correct the aggressive thrust, which easily reaches the

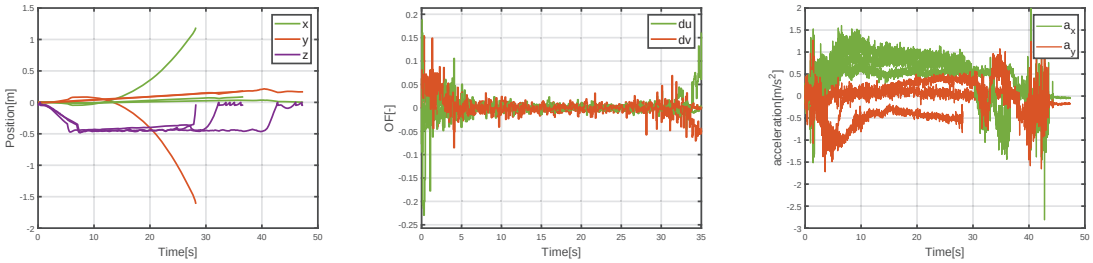
500 Pulse-Width Modulation signals limitation per motor or cannot lift to the required height and shut down. To figure out the possible causes of instability, it is necessary to have insights into the horizontal estimator and controller eigenvalue.

5.2.1 Default horizontal estimator checking

As mentioned in section 4.2.1, the default horizontal estimator applies an inaccurate conversion model and fuses data from the optical flow sensor and accelerometer. According to Parrot [12], the data from the low-cost accelerometer is too noisy to integrate twice, which leads the doubt about the necessity of data fusing. Moreover, considering the remaining drift problem in Brekelmans's report [7], the unmodeled optical flow estimation should be taken into account.

Accelerometer drift

To evaluate the interaction of the accelerometer, the drone is manually held around 0.4m high above the origin, which is repeated three times. The position $[x, y, z]^T$ is integrated from horizontal optical flow and detected by ultrasonic sensor shown as Figure 5.2a. Movement x and y of this load diagram clearly illustrate the increasing horizontal drift, but the movement z offers a reliable detection keeping around 0.4m high. The optical flow converges around zero as depicted as Figure 5.2b.



(a) **(b)** **(c)**

Figure 5.2: Handheld hover experiments (a)The position of handheld experiments.From 10 second the drone reaches desired height; (b) The raw $x-$ and $y-$axis optical flow $[du, dv]^T$ from optical flow sensor; (c) The raw $x-$ and $y-$axis data $[a_x, a_y]^T$ from accelerometer

Based on the refined conversion algorithm (4.14), such stable and reliable altitude and optical flow should promise the horizontal position around zero close to the ground truth. In other words, the drift only can be introduced by the constantly drifted acceleration in the Kalman filter as depicted as Figure 5.2c. Due to the variant drift of each experiment, it cannot be removed by set-off. Therefore, we recommend removing the Kalman filter with acceleration (further experimental proof see section 5.3.2).

Unmodeled optical flow estimator

Referring to the default estimator in section 4.2.1 and the thesis of G.H. Brekelmans [7], both methods only consider the depth variation but ignore the rigid orientation, which are unmodeled as

$$\dot{x} = -az * du \tag{5.18a}$$

$$\dot{y} = -az * dv, \tag{5.18b}$$

where in [7] $a = 1.05$ and in the default estimator $a = 1.15$.

However, as the flow field always reflects the change from the transition and rotation of on-board camera itself, it is inappropriate to leave the angular rate along. Hence, in section 5.3, we would further detail a new corrected estimator fusing the data from gyroscope and optical flow sensor.

5.2.2 Eigenvalue checking

Given the drifted horizontal estimation, it is natural to investigate the robust stability of the paper's controller within the inaccurate estimated inputs and unknown environmental variances [34]. From the none-observer state-space representation

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{5.19a}$$
$$\dot{\nu}_e = -S(\omega_r)\nu_e + u \tag{5.19b}$$
$$u = -(k_\rho \rho_e + k_\nu \nu_e) \tag{5.19c}$$
$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{5.19d}$$
$$J\dot{\omega} = S(J\omega)\omega + \tau \tag{5.19e}$$
$$\tau = \tau_{rd} + S(J\omega_e)\omega_{rd} + K_\omega \omega_e + \sum_{i=1}^{3} k_i S(q_{rd}^* \odot \nu_i) q^* \odot \nu_i, \tag{5.19f}$$

we can follow the similar steps of section 5.1.1 to linearize this model around hover point as

$$\begin{bmatrix} \dot{\rho}_e \\ \dot{\nu}_e \end{bmatrix} = \begin{bmatrix} \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ I \end{bmatrix} u \tag{5.20a}$$
$$u = \begin{bmatrix} -k_\rho & -k_\nu \end{bmatrix} \begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} \tag{5.20b}$$
$$\dot{E} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \omega \tag{5.20c}$$
$$\dot{\omega} = \tau_L \tag{5.20d}$$
$$\tau_L = \underbrace{\frac{k_\rho[(k_\rho - k_\nu^2)J - K_\omega k_\nu - gK_p]}{g}}_{K_{\tau 1}} \begin{bmatrix} y_e \\ -x_e \\ 0 \end{bmatrix} + \underbrace{\frac{k_\nu(2k_\rho - k_\nu^2)J - K_\omega(k_\rho - k_\nu^2) - gK_p k_\nu}{g}}_{K_{\tau 2}} \begin{bmatrix} v_{ye} \\ -v_{xe} \\ 0 \end{bmatrix} - K_p E - K_\omega \omega, \tag{5.20e}$$

with

$$K_p = \begin{bmatrix} (k_3 + k_2) & 0 & 0 \\ 0 & (k_3 + k_1) & 0 \\ 0 & 0 & (k_2 + k_1) \end{bmatrix}. \tag{5.21}$$

Such set of equations can be equivalently rearranged into

$$\begin{bmatrix} \dot{\rho}_e \\ \dot{\nu}_e \\ \dot{E} \\ \dot{\omega} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{A} \begin{bmatrix} \rho_e \\ \nu_e \\ E \\ \omega \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}}_{B} \begin{bmatrix} u_L \\ \tau_L \end{bmatrix} \tag{5.22a}$$

$$\begin{bmatrix} u_L \\ \tau_L \end{bmatrix} = \underbrace{\begin{bmatrix} -k_\rho I & -k_\nu I & \mathbf{0} & \mathbf{0} \\ K_1 & K_2 & -K_p & -K_\omega \end{bmatrix}}_{K} \begin{bmatrix} \rho_e \\ \nu_e \\ E \\ \omega \end{bmatrix} \tag{5.22b}$$
$$K_1 = \begin{bmatrix} -K_{\tau 1}(:,2) & K_{\tau 1}(:,1) & \mathbf{0} \end{bmatrix} \tag{5.22c}$$
$$K_2 = \begin{bmatrix} -K_{\tau 2}(:,2) & K_{\tau 2}(:,1) & \mathbf{0} \end{bmatrix} \tag{5.22d}$$

From the pole-placement theorem, we know that stabilizing the closed loop

$$\dot{e} = (A - BK)e = Me \tag{5.23}$$

requires the eigenvalues of M in the open left-half plane. Substituting (5.22) into M, its eigenvalues are calculated as

$$\text{eig}(M) = \begin{bmatrix} -K_{\omega 1}/2 - \sqrt{(K_{\omega 1}^2 - 4k_2 - 4k_3)}/2 \\ \sqrt{(K_{\omega 1}^2 - 4k_2 - 4k_3)}/2 - K_{\omega 1}/2 \\ -K_{\omega 2}/2 - \sqrt{(K_{\omega 2}^2 - 4k_1 - 4k_3)}/2 \\ \sqrt{(K_{\omega 2}^2 - 4k_1 - 4k_3)}/2 - K_{\omega 2}/2 \\ -K_{\omega 3}/2 - \sqrt{(K_{\omega 3}^2 - 4k_1 - 4k_2)}/2 \\ \sqrt{(K_{\omega 3}^2 - 4k_1 - 4k_2)}/2 - K_{\omega 3}/2 \\ -k_v/2 - \sqrt{(k_v^2 - 4k_\rho)}/2 \\ -k_v/2 - \sqrt{(k_v^2 - 4k_\rho)}/2 \\ -k_v/2 - \sqrt{(k_v^2 - 4k_\rho)}/2 \\ \sqrt{(k_v^2 - 4k_\rho)}/2 - k_v/2 \\ \sqrt{(k_v^2 - 4k_\rho)}/2 - k_v/2 \\ \sqrt{(k_v^2 - 4k_\rho)}/2 - k_v/2 \end{bmatrix} \tag{5.24}$$

Substitution of the value of the well-p simulation gains (5.17) results in the following eigenvalue

$$\text{eig}(M) = \begin{bmatrix} -0.0012 + 0.1140i \\ -0.0012 - 0.1140i \\ -0.0010 + 0.1000i \\ -0.0010 - 0.1000i \\ -7.1865 + 0.0000i \\ -7.1865 + 0.0000i \\ -7.1865 + 0.0000i \\ -1.6371 + 0.0000i \\ -1.6371 + 0.0000i \\ -1.6371 + 0.0000i \\ -0.0022 + 0.0632i \\ -0.0022 - 0.0632i \end{bmatrix}. \tag{5.25}$$

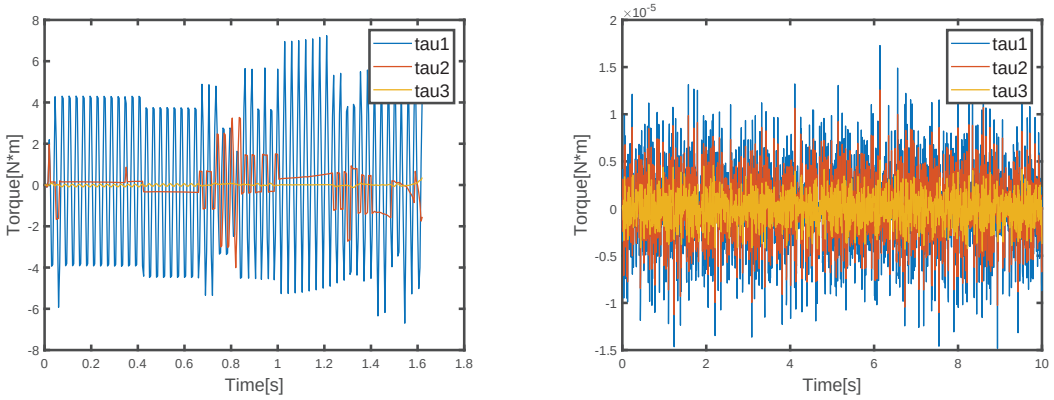
The first four and last two entries given above are too close to zero, which can be one possible cause for the oscillating and circling during the experiments. To align the magnitude of all entries, the eigenvalue of closed loop can set a boundary of $\lambda < -1$ to obtain underlying constraints of

attitude gains, which leads to

$$K_\omega = \begin{bmatrix} K_{\omega 1} & 0 & 0 \\ 0 & K_{\omega 2} & 0 \\ 0 & 0 & K_{\omega 3} \end{bmatrix} > 2I \tag{5.26a}$$

$$\begin{bmatrix} (k_2 + k_3) & 0 & 0 \\ 0 & (k_1 + k_3) & 0 \\ 0 & 0 & (k_2 + k_1) \end{bmatrix} > I. \tag{5.26b}$$

However, such aggressive gains are infeasible for a mini-drone like Parrot Mambo. In general, during hover experiments with default PID controller, the gyroscope provides the range of angular rate in $[-0.2, 0.2]$rad/s. Assuming an ideal situation that position error and angle error are around 4 decimal places (trailing zeros suppressed). If we set a conservative gain as $K_\omega = 2I$, the torque (5.20e) is given by at least 1 decimal places(trailing zeros suppressed). Combined with the anti-gravity thrust f, the motor will immediately reach the limitation and flip over (see a torque simulation comparison between infeasible paper's controller and stable default PID controller in Figure 5.3a and 5.3b).



(a) **(b)**

Figure 5.3: Torque comparison: (a)The torque range of tuned attitude controller gains; (b) The torque range of default PID controller

Therefore, the robust gains are impossible to apply to our set-up. But the improvement of optical flow estimation might compensate the sensitive controller and stop sending continuous drifted horizontal position error that causes vehicle tilting.

5.3 Improvement of horizontal estimation process

There follows a description of a new horizontal estimation based on modeled optical flow(OF) processing. We provide a corresponding relationship between the data of OF sensor and the rigid motion. Then the new estimator is validated by the handheld hover experiments and simple harmonic pendulum experiments. In particular, since a small constant acceleration drift leads to a greater increasing velocity error after integration shown in section 5.2.1, it is appropriate to exclude the Kalman filter with acceleration data in the new estimator.

5.3.1 Theory of optical flow in motion field

After estimating the optical flow$(\dot{u}, \dot{v})$ through the firmware, in order to explain the actual linear velocity, an accurate model should not only perform the scaling change as (4.14), but also the

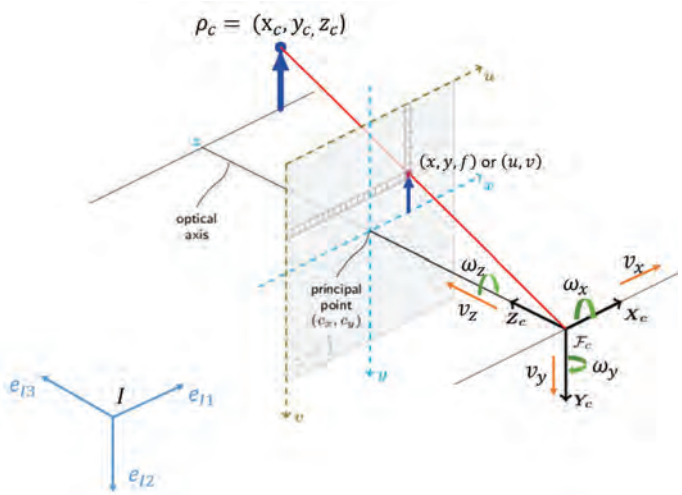


Figure 5.4: Pinhole camera model (Bradski 2000)

rotation interaction [12]. Theoretically, we assume the optical flow is generated by rigid motions in a pinhole model [2] [12].

In an ideal pinhole model, let the optical axis be the z-axis of the camera frame (see Figure 5.4), and let $\rho = [x_c, y_c, z_c]^T$ be the coordinate of a point in the camera frame. And projected coordination $p = [x, y, f]^T$ under the image frame is given by

$$\frac{p}{f} = \frac{\rho_c}{z_c}. \tag{5.27}$$

A close look at the image plane in Figure.5.5, the pixel frame normalizes the physical image frame with the pixel size $P \times P$. Its origin sets at the upper-left corner rather than the image center. Any point on the image plane can be transformed as

$$(u, v) = \frac{(0,0) - (x_0, y_0) + (x, y)}{P} = \frac{(x - x_0, y - y_0)}{P} \tag{5.28a}$$

$$(x, y) = ((u, v) - (0,0) - (u_0, v_0))P = (u - u_0, v - v_0)P. \tag{5.28b}$$

During the navigation, the instantaneous velocity of the camera can be decomposed to a translation $T = [v_x, v_y, v_z]^T$ and a rotation $\omega = [\omega_x, \omega_y, \omega_z]^T$ [2]. Considering relative movement, the point $\rho = [x_c, y_c, z_c]^T$ can be viewed as moving in an opposite way [18]

$$\begin{aligned}
\dot{\rho}_c &= -\omega \times \rho_c - T \\
&= -\begin{bmatrix} v_x - y_c\omega_z + z_c\omega_y \\ v_y + x_c\omega_z - z_c\omega_x \\ v_z - x_c\omega_y + y_c\omega_x \end{bmatrix} \\
&= -\begin{bmatrix} v_x - \dfrac{yz_c}{f}\omega_z + z_c\omega_y \\ v_y + \dfrac{xz_c}{f}\omega_z - z_c\omega_x \\ v_z - \dfrac{xz_c}{f}\omega_y + \dfrac{yz_c}{f}\omega_x \end{bmatrix}
\end{aligned} \tag{5.29}$$

By derivation of (5.27) with respected to time, the optical flow can be expressed as

$$\dot{p} = f\frac{\dot{\rho}_c z_c - \rho_c \dot{z}_c}{z_c^2}. \tag{5.30}$$

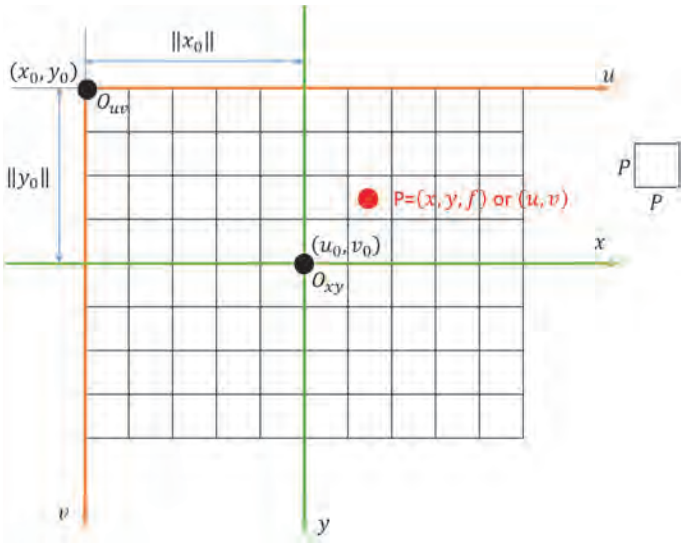


Figure 5.5: Image plane(each pixel is a squad)

Substituting (5.29) into the above equation, the projected point moving in the image frame can be expressed as

$$\dot{x} = \frac{v_z x - v_x f}{z_c} - \omega_y f + \omega_z y + \frac{\omega_x xy - \omega_y x^2}{f} \tag{5.31a}$$

$$\dot{y} = \frac{v_z y - v_y f}{z_c} + \omega_x f - \omega_z x + \frac{\omega_x y^2 - \omega_y xy}{f}, \tag{5.31b}$$

where $\dot{x}, \dot{y}$ can be sampled and scaled into the discrete optical flow $[\dot{u}, \dot{v}]$ as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\dot{x}}{P} \\ \frac{\dot{y}}{P} \end{bmatrix}. \tag{5.32}$$

Comparing with the magnitude of the altitude $z_c \in (0, 6]$m and the focal length $f = 23$mm, the image plane of the charge-coupled device(CCD) only comes in 1/2.4-inch (Diagonal 7.487mm). Therefore, in (5.31) those terms, which contain the image coordinate $[x, y]$ and are divided by altitude z_c or the focal length f, are smaller in more than one order of magnitude than others [18]. And as mentioned in section 4.2.1, for an evenly integrated detection algorithm on the whole image, transition v_z or rotation ω_z of the $z-$axis has no contribution to optical flow. A simplified conversion model is

$$\dot{x} = \frac{-v_x f}{z_c} - \omega_y f \tag{5.33a}$$

$$\dot{y} = \frac{-v_y f}{z_c} + \omega_x f. \tag{5.33b}$$

As the last step, the optical flow $[\dot{u}, \dot{v}]$ detected in pixel frame needs to be reversed to horizontal velocity $[v_x, v_y]$ in body frame, which is

$$v_x = \frac{-z_c}{f}\dot{x} - \omega_y z_c = (\frac{-P}{f}\dot{u} - \omega_y) z_c \tag{5.34a}$$

$$v_y = \frac{-z_c}{f}\dot{y} + \omega_x z_c = (\frac{-P}{f}\dot{v} + \omega_x) z_c. \tag{5.34b}$$

5.3.2 Experimental validation

Even though the mathematical equation (5.34) promises the horizontal velocity theoretically, since the output (du, dv) of the OF sensor is still unclear about Parrot interpretation of their unit and the internal correction of the inaccessible firmware, (5.34) requires the introduction of two parameters (a, b) into reversion:

$$v_x(a,b) = (-a * du - b * \omega_y) z_c \tag{5.35a}$$
$$v_y(a,b) = (-a * dv + b * \omega_x) z_c. \tag{5.35b}$$

For factor estimation and validation, the experiments, such as handheld hover and pendulum experiments, with calculable trajectory as ground truth are desired for statistical analysis.

Handheld hover validation

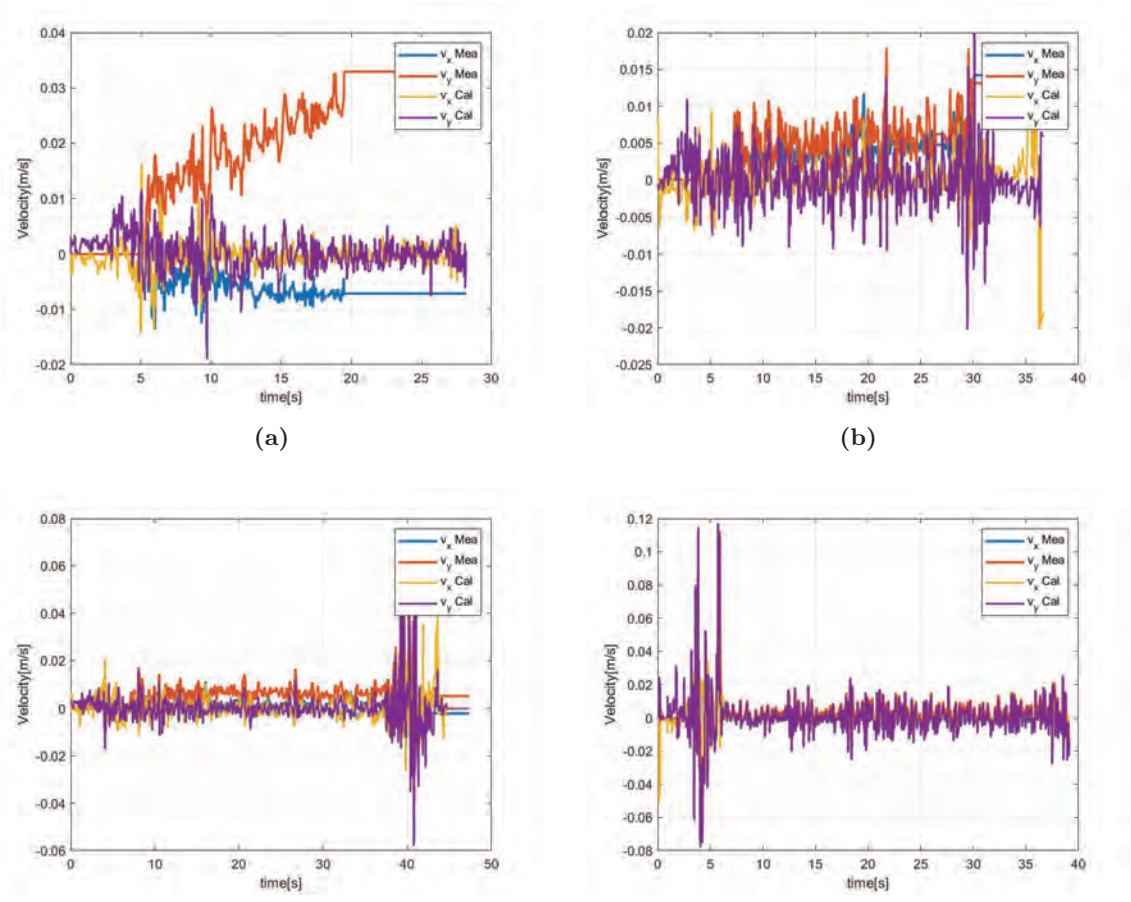


Figure 5.6: The handheld hover validations. $*Mea$ denotes the default estimated velocity and $*Cal$ denotes the calculated velocity with same default model but without Kalman filter.

A particular advantage of hover experiments is the zero reference horizontal velocity, i.e. $[v_x, v_y]^T = [0, 0]^T$m/s, which is an intuitive starting point for troubleshooting. Measurements were performed in a hand-lifting drone and focused on the static period. During this period, a desired horizontal velocity estimation should converge to the ground truth, zero.

As mentioned earlier, we suspect that it is the Kalman filter of the default controller triggering horizontal velocity drifts constantly. For validation, the default estimated velocity compares with its pure conversion model (4.14) in Figure 5.6. In the first three sets of data (5.6a-5.6c), the default estimated velocity drifts clearly, where the unfiltered velocity calculated from the same model(4.14) converges around zero. Even though the default estimator performs more reliably in Figure(5.6d), in reality besides the rotating reference, resisting wind requires to tilt the vehicle. Due to such situations, gravity acceleration tensor G_b always interferes with the horizontal acceleration measurements with an item $R^T G_b$ (see (4.1a)). There is hesitation to trust the noisy gyroscope-based rotation matrix R can ensure an accurate G_b transformation and the final correct calibration. Therefore, it is necessary to remove the Kalman filter to avoid velocity drift caused by acceleration through hand-held hover verification.

Pendulum validation

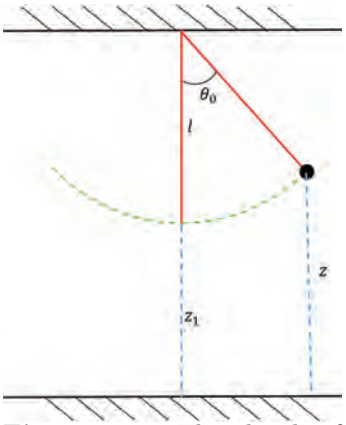


Figure 5.7: The sketch of pendulum model

Figure 5.8: The pendulum set-up

As the most common simple harmonic model, a simple pendulum can be easily assembled. The classic model(see Figure.5.7) consists of a mass point m and an infinitely light rod with length l initiated at a θ_0 angle displacement from vertical position on and rotates around a frictionless pivot [4]. Considering Newton's second law

$$F = ma, \tag{5.36}$$

in the pendulum model, the gravity force provides the force F and angular acceleration delivers the linear acceleration $a = l\frac{d^2}{dt^2}\theta$, which is

$$-mg\sin\theta = ml\frac{d^2}{dt^2}\theta. \tag{5.37}$$

To simplify the state space model of (5.37), we assume the pendulum oscillates in a small angle($\theta < 8°$) that

$$(1 - \cos 8°)/1 \approx 0.0097 < 1\%. \tag{5.38}$$

Therefore, with the small angle approximation, $\sin\theta \approx \theta$, we have

$$\dot{x}_1 = x_2 \tag{5.39a}$$

$$\dot{x}_2 = -\frac{g}{l}x_1, \tag{5.39b}$$

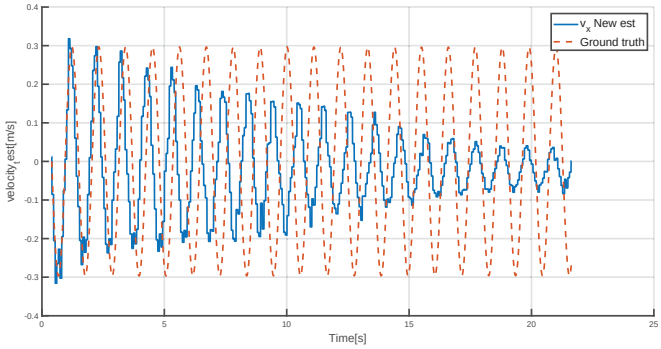


Figure 5.9: The fitted velocity and theoretical value

	OF factor	New			Improvement (cov_OF-cov_New)/cov_OF
	cov_OF	cov_New	a	b	
(a)	0.405	0.3292	1	0.02	0.187160494
(b)	0.5181	0.3455	1	0.04	0.33314032
(c)	0.564	0.5397	1	0	0.043085106
(d)	0.3026	0.2562	1	0.02	0.15333774
(e)	0.3739	0.3497	1	0	0.064723188
(f)	0.3177	0.3015	1	0	0.050991501

Table 5.1: Corresponding covariance of 6 simple pendulum experiments

where x_1 is the angle θ and x_2 is the angular velocity $\dot{\theta}$; g is gravity acceleration; l is pendulum length. The solution of the angle position x_1 or θ can be written as

$$\theta = \theta_0 \sin\left(\sqrt{\frac{g}{l}}t\right) \tag{5.40}$$

Given by (5.40), the theoretical value of period T and linear velocity v can be derived

$$T = 2\pi\sqrt{\frac{l}{g}} \tag{5.41a}$$

$$v = \dot{\theta}l = \theta_0\sqrt{gl}\cos\left(\sqrt{\frac{g}{l}}t\right) \tag{5.41b}$$

In the experiments, we attach the drone with a light stick in a joint as shown in Figure 5.8 and collect 6 sets of data with small-angle oscillation. However, since the friction attenuates the oscillation significantly and enlarges the period gradually as Figure 5.9, only the first period of six experiments is available to evaluate the covariance of the estimation.

As illustrated in Figure.5.10, each first period of 6 simple pendulum data sets was separated and fitted with the theoretical value/ground truth (5.41b). Considering this theoretical value as the expected value, we can evaluate the covariance from the two estimation models: (5.18b) and (5.35). Specifically, the optical flow factor (OF factor) method(5.18b) sets the parameter as 1.05 from [7], and the new estimator (5.35) dynamically calculates its optimal parameters $a \in [1, 2), b \in [0, 1)$ based on a minimum covariance loop. From Table 5.1, in general, this new estimator can match all the ground truth more accurately, even though 3 sets of data only improve $4.3 - 5.1\%$. Observing all pairs of calculated parameters (a, b) for this new estimator, we can find that different from the dynamic b, a is statical to 1; and half of the datasets show the introduction of angular compensation

parameter b significantly improving the accuracy more than 15%. To benefit from the angular compensation and avoid more noise from the gyroscope, we recommend using a conservative value $b = 0.02$. However, since the experiments significantly lose energy from friction or air resistance, in the future, including the friction factor into the model can sufficiently utilize more periods of data and improve the accuracy of parameters (a, b).

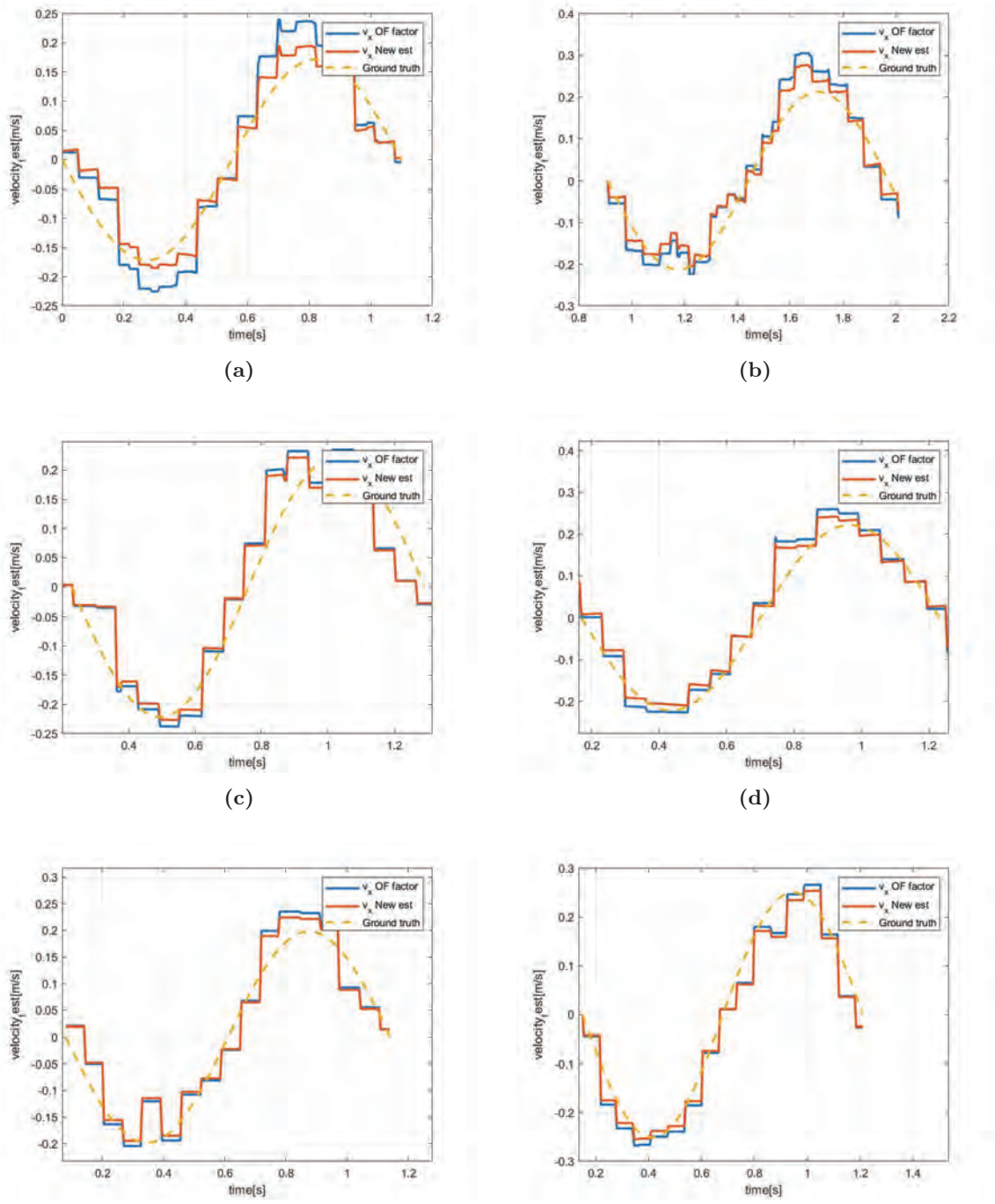


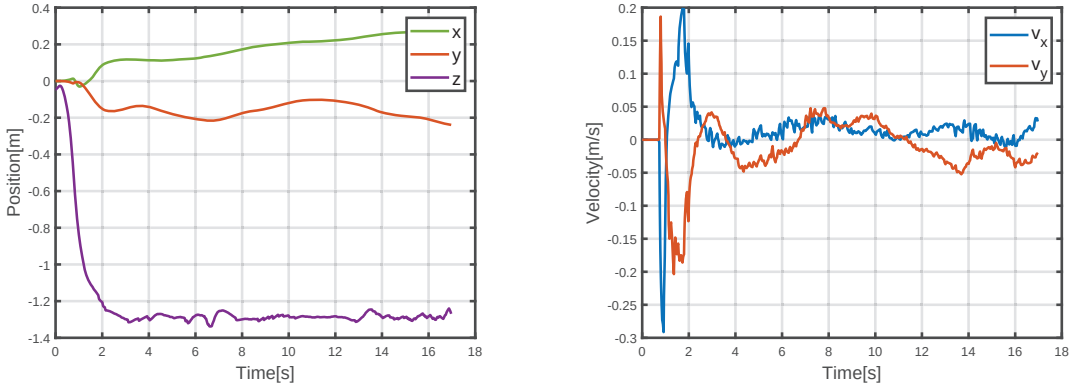
Figure 5.10: The pendulum validations. $*OFfactor$ denotes the default estimated velocity without Kalman filter(i.e. Brekelmans's method [7]) and $*Newest$ denotes the calculated velocity based on (5.35) with smallest covariance.

5.3.3 Hover test

Given the validations in section 5.3.2, we determine a new optical flow estimator excluding the Kalman filter and applying the new estimation model

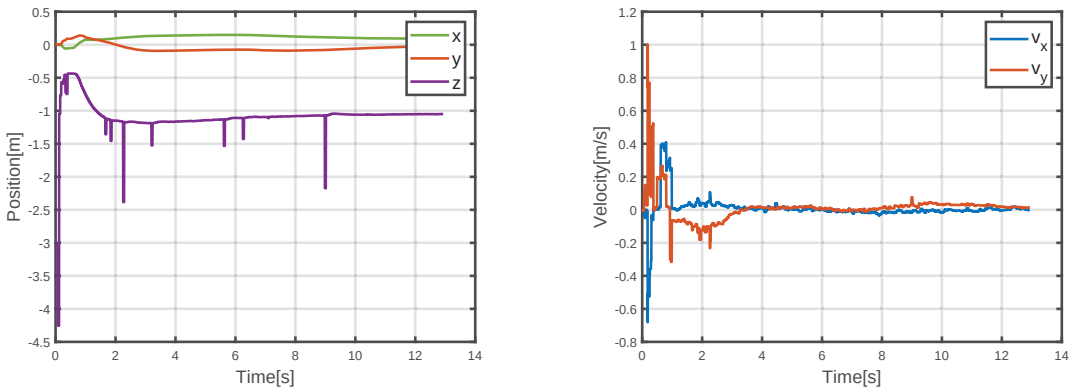
$$v_x = (-du - 0.02\omega_y)z_c \tag{5.42a}$$
$$v_y = (-dv + 0.02\omega_x)z_c. \tag{5.42b}$$

As a robust controller, the default PID controller can be the first subject to validate this new estimator in hover experiments without flipping safety concerns. There are two sets of hover flight position data. In Figure 5.11, even with preprocessing filters, the original default horizontal estimator still provides turbulent horizontal velocity, which renders its integrated horizontal position increasingly drifting away from the origin. Figure 5.12 shows that the new estimator delivers a stable set of velocity regressing to zero, even though all filters have been removed. Subsequently, its position also converges to the origin. However, even though this new estimator has promised the default PID controller to drift less during hover flight, it still cannot avoid the oscillations and achieve stable flight of the paper's controller.



(a) **(b)**

Figure 5.11: The hover performance with default estimator: (a) x, y, z position (b) Horizontal velocity v_x, v_y



(a) **(b)**

Figure 5.12: The hover performance with new estimator: (a) x, y, z position (b) Horizontal velocity v_x, v_y

5.4 Concluding remarks

By the support of the default PID controller, a set of resembling gains are derived from the linearized comparison between the default controller and the paper's controller. Within these knowledge and Simulink tests, a set of gains finally settles in perfect simulation results; however, they perform undesirably in reality. Its causes analysis illustrates that the drifted horizontal estimator and sensitive controller force lead to the circling flight. Unfortunately, such a mini drone cannot satisfy such high gains for robustness. Hence, we focused on the improvement of the estimator. Via the hover experiments with the default controller, our estimator has been proved for better reference tracking.

Chapter 6

Conclusions and recommendations

The purpose of this thesis is to implement the controller in [26] on a Parrot Mambo drone. According to the contents of the earlier chapters, this chapter draws conclusions and recommendations for future research. In Section 6.1, we summarize the work of this project and conclude it. Based on these conclusions, suggestions for future research are discussed in Section 6.2.

6.1 Summary and conclusion

This project acts as a stepping stone for future controller design and implementation on the Parrot Mambo drone. Beginning with a theoretically promising controller of [26], we discussed the orientation representation forms, two crucial time derivative expressions, and a general flight dynamics model as preliminaries in Chapter 2. Without loss of generality, the versor is chosen as the final form in code for faster calculation, whereas a rotation matrix representation is commonly exhibited in the whole thesis for intuitive expression. For simplifying mathematical equations in the remainder, two time-differentiation forms are explained in section 2.2. In the end, as the foundation of control design, a nonlinear flight dynamics model of the quadcopter is introduced.

Based on the general flight dynamics, Chapter 3 first derives a flat-output-based reference. It ensures a sufficiently smooth trajectory for future controller validation. The controller from the study in [26] is discussed in the second section. We explain the construction of position and attitude controllers and detail the virtual-input-based bridge for both controllers. Finally, a full state controller with versor form is delivered. Specifically, since there is a designed observer as a low-pass filter, it is unnecessary to facilitate any extra preprocessing filter to avoid the delay.

Before starting to implement the controller of Chapter 3, Chapter 4 delivers a pivotal insight into the experimental set-up: a Parrot Mambo drone and the Parrot Simulink Package. Given Parrot's website and Brekelmans' work [7], the two crucial physical characters: mass and inertial tensor, have been settled. And we are also aware of the limited accuracy and accessibility of each on-board sensor. After that, a control law consisting of an estimator and a controller should take over the measured data, estimate the required states, and control the rotor. These algorithms can be simulated and then embedded through the Parrot Mambo package to the drone. Considering the default control law given by the Parrot Mambo package is robust in hover experiments, this chapter detailedly analysis the schemas and mathematical formulas of this control law. Hereto, the next chapter can utilize this information for the implementation of the paper's controller.

Given the theoretical control law of Chapter 3 and the experimental set-up pre-knowledge of Chapter 4, the final implementation is discussed in Chapter 5. For such a complex nonlinear controller illustrated in Chapter 3, its broad theoretical constraints provide few hints for stability tuning during the implementation. Therefore, linearizing and resembling a robust controller, like

the default PID controller in Chapter 4, promises an intuitive tuning range. Supported by this range, we determine a set of well-performed gains with a perfect performance in hover simulations. However, its experimental validations oscillate or even circle dangerously. Through the eigenvalue and state estimation checking, the sensitive attitude controller and the constantly drifted horizontal estimation should be the causes of the instability. For the former one, some eigenvalue entries of the optical gains are too close to zero, only 2 decimal places(trailing zeros suppressed). Following the pole-placement theorem, we can set a relatively robust constraint for eigenvalue, i.e. $\lambda < -1$, and derive its corresponding constraints for gains. Due to the motor limitation, the derived robust but high gains are infeasible for the Parrot Mambo mini-drone. For the latter cause, as mentioned in Brekelmans' thesis [7], the default optical flow estimator drifts continuously for horizontal velocity. From the handheld hover experiments, the noisy accelerometer is proved as the cause for most constant drift, where it is wise to remove the Kalman filter. Additionally, for better estimation, we developed a theoretical model from the theory of optical flow in the motion field, which takes not only transition (as the default estimator did) but also rotation into account. Validated by the simple pendulum experiments, this new estimator can improve at least $5\% - 33\%$ accuracy.

6.2 Recommendation for further research

Even though a set of well-performed gains has been settled among well-performed hover simulations, its hover experiments show dangerous oscillation. As mentioned in section 6.1, the eigenvalue checking has proved the infeasibility of the robust gains for such a mini-drone. Therefore, we focus on improving the horizontal motion estimator to limit the continuous drifted signals, which would ask the attitude controller tilting vehicle and finally lead to oscillation even circling. Although our new horizontal estimator improves $5 - 33\%$ accuracy, it depends on the altitude data. Given the observers designed in Chapter 3, we removed all filters of estimation processing. During the experiments, the altitude occasionally reflects the nonnegligible peak noise. For future research, we recommend to re-consider adding some filters before the estimation. To avoid over filtering, it is wise to design a set of control experiments to statistically evaluate the delay and the accuracy improvement of the combination of ultra-sonar filter and observer. Moreover, the camera photo would provide the position information, too. Via setting two known landmarks, the scale between the absolute size/distance and the shown size/distance on the image can easily calculate the height. Note that the onboard camera provides a small field of view (FoV). Therefore, this auxiliary height calculation algorithm might only suit the hover experiment to catch all landmarks. Or paving equally spaced landmarks is a more general choice in the future.

For the continuous oscillation of the paper's controller [26] during its experiments, we recommend taking the inertial tensor of the vehicle and the sign of direction vectors v_i into account. In the default PID controller, the inertia tensor is absent, which causes the drone to maintain balance even if the horizontal position has biased more than 0.5m. Additionally, in the paper the well-performed simulation bases on a drone supposed as 0.1 kg but 1000 times larger inertial tensor than our set-up's. For the direction vectors v_i, our implementation uses positive unit vectors from the axes of a Cartesian coordinate system; however, the paper uses two negative vectors for the attitude controller. Therefore, future research should reconsider these points to stabilize the vehicle attitude.

Appendix A

The derivation of desired trajectory

As mentioned in Chapter 3, this Appendix would detail the derivation of the desired attitude set for attitude, including rotation matrix R_d, unit quaternions q_d, and angular rate ω_d. Moreover, their time differential formulas would also be delivered to update the attitude reference dynamics (3.29a-3.29c). From (3.25) it delivers a clear message that though $R_d = R_r^T R$ or $q_d = q_r^* q$ the vector e_3 can be rotated to the desired thrust direction f_d. For intuition, the derivation (3.26,3.36) focuses on versor form [22].

As was known for a unit quaternions q_d, it can be constructed by an Euler axis $\vec{n}$ and rotation angle θ around $\vec{n}$. That is

$$q_d = \begin{bmatrix} \cos\frac{\theta}{2} \\ \vec{n}\sin\frac{\theta}{2} \end{bmatrix}. \tag{A.1}$$

Once the f_d direction

$$f_d = \begin{bmatrix} f_{d1} \\ f_{d2} \\ f_{d3} \end{bmatrix} \tag{A.2}$$

is derived by (3.25), the Euler axis and rotation angle can be achieved by inner product and cross product between e_3 and f_d

$$e_3 \cdot f_d = f_{d3} = \|e_3\|\|f_d\|\cos\theta \tag{A.3a}$$

$$e_3 \times f_d = \begin{bmatrix} -f_{d2} \\ f_{d1} \\ 0 \end{bmatrix} = \|e_3\|\|f_d\|\sin\theta\vec{n} \tag{A.3b}$$

.

It is obvious that the Euler axis can be extracted from [22] (A.3) as

$$\vec{n} = \frac{e_3 \times f_d}{\sin\theta} = \frac{e_3 \times f_d}{\sqrt{1-(e_3 \cdot f_d)^2}} = \frac{1}{\sqrt{1-f_{d3}^2}} \begin{bmatrix} -f_{d2} \\ f_{d1} \\ 0 \end{bmatrix}, \tag{A.4}$$

and the cosine and sine function of half rotation angle $\frac{\theta}{2}$ are given as

$$\cos\frac{\theta}{2} = \sqrt{(1+\cos\theta)/2} = \sqrt{(1+e_3 \cdot f_d)/2} = \sqrt{(1+f_{d3})/2} \tag{A.5a}$$

$$\sin\frac{\theta}{2} = \sqrt{(1-\cos\theta)/2} = \sqrt{(1-e_3 \cdot f_d)/2} = \sqrt{(1-f_{d3})/2}. \tag{A.5b}$$

Substitution of (A.4-A.5) into (A.1) gives

$$q_d = \begin{bmatrix} \sqrt{\frac{(1+f_{d3})}{2}} \\ -\frac{f_{d2}}{\sqrt{2(1+f_{d3})}} \\ \frac{f_{d1}}{\sqrt{2(1+f_{d3})}} \\ 0 \end{bmatrix}. \tag{A.6}$$

Equivalently, its rotation matrix is

$$R_d = \begin{bmatrix} 1-\frac{f_{d1}^2}{(1+f_{d3})} & -\frac{f_{d1}f_{d2}}{(1+f_{d3})} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{(1+f_{d3})} & 1-\frac{f_{d2}^2}{(1+f_{d3})} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix}. \tag{A.7}$$

Differentiating (3.36) with time gives

$$\dot{q}_d = \begin{bmatrix} \frac{1}{4}(\frac{1+f_{d3}}{2})^{-\frac{1}{2}} \\ -(\frac{\dot{f}_{d2}}{\sqrt{2(1+f_{d3})}} - \frac{f_{d2}\dot{f}_{d3}}{(2(1+f_{d3}))^{\frac{3}{2}}}) \\ \frac{\dot{f}_{d1}}{\sqrt{2(1+f_{d3})}} - \frac{f_{d1}\dot{f}_{d3}}{(2(1+f_{d3}))^{\frac{3}{2}}} \\ 0 \end{bmatrix}. \tag{A.8}$$

Following (3.32c), the desired angular is derived as

$$\omega_d = 2q_d^* \otimes \dot{q}_d = \begin{bmatrix} -\dot{f}_{d2} + \frac{f_{d2}\dot{f}_{d3}}{1+f_{d3}} \\ \dot{f}_{d1} - \frac{f_{d1}f_{d3}}{1+f_{d3}} \\ \frac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix}. \tag{A.9}$$

The desired attitude parameters have been derived in above equations. It is possible to update the attitude reference dynamics:

$$\begin{aligned} \dot{R}_{rd} &= \frac{d}{dt}(R_r R_d) = \dot{R}_r R_d + R_r \dot{R}_d \\ &= R_r S(\omega_r) R_d + R_r R_d S(\omega_d) \\ &= R_r R_d S(R_d^T \omega_r) + R_r R_d S(\omega_d) \\ &= R_{rd} S(\underbrace{R_d^T \omega_r + \omega_d}_{\omega_{rd}}) \\ &= R_{rd} S(\omega_{rd}) \end{aligned} \tag{A.10a}$$

$$\begin{aligned} J\dot{\omega}_{rd} &= J\frac{d}{dt}(R_d^T \omega_r + \omega_d) \\ &= J\dot{R}_d^T \omega_r + JR_d^T \dot{\omega}_r + J\dot{\omega}_d \\ &= J(R_d S(\omega_d))^T \omega_r + JR_d^T \dot{\omega}_r + J\dot{\omega}_d \\ &= S(J\omega_{rd})\omega_{rd} + \underbrace{(-S(J\omega_{rd})\omega_{rd} + J(R_d S(\omega_d))^T \omega_r + JR_d^T \dot{\omega}_r + J\dot{\omega}_d)}_{\tau_{rd}} \\ &= S(J\omega_{rd})\omega_{rd} + \tau_{rd} \end{aligned} \tag{A.10b}$$

or equivalently,

$$\dot{q}_{rd} = \frac{1}{2} q_{rd} \otimes \begin{bmatrix} 0 \\ \omega_{rd} \end{bmatrix} \tag{A.11a}$$

$$J\dot{\omega}_{rd} = S(J\omega_{rd})\omega_{rd} + \tau_{rd} \tag{A.11b}$$

$$\tau_{rd} = -S(J\omega_{rd})\omega_{rd} + J\dot{q}_d^* \odot \omega_r + Jq_d^* \odot \dot{\omega}_r + J\dot{\omega}_d. \tag{A.11c}$$

Note that in (A.12) the particular shape of τ_{rd} is associated with several time derivatives of desired parameters

$$\dot{q}_d^* = \frac{1}{2}(q_d \otimes \begin{bmatrix} 0 \\ \omega_d \end{bmatrix})^* = \frac{1}{2}\begin{bmatrix} 0 \\ \omega_d \end{bmatrix}^* \otimes q_d^* = -\frac{1}{2}\begin{bmatrix} 0 \\ \omega_d \end{bmatrix} \otimes q_d^* \tag{A.12a}$$

$$\dot{\omega}_d = \begin{bmatrix} -\ddot{f}_{d2} + \frac{(\dot{f}_{d2}\dot{f}_{d3} + f_{d2}\ddot{f}_{d3})(1+f_{d3}) - f_{d2}\dot{f}_{d3}^2}{(1+f_{d3})^2} \\ \ddot{f}_{d1} + \frac{(\dot{f}_{d1}\dot{f}_{d3} + f_{d1}\ddot{f}_{d3})(1+f_{d3}) - f_{d1}\dot{f}_{d3}^2}{(1+f_{d3})^2} \\ \frac{(f_{d2}\ddot{f}_{d1} - f_{d1}\ddot{f}_{d2})(1+f_{d3}) - (f_{d1}f_{d2} - f_{d1}\dot{f}_{d2})\dot{f}_{d3}}{(1+f_{d3})^2} \end{bmatrix} \tag{A.12b}$$

However it is clear that the 1st and 2nd order derivatives of desired thrust direction f_d (3.25) are needed, which are given by:

$$\begin{aligned} \frac{d}{dt}f_d &= -\frac{\dot{f}}{f^2}(f_r e_3 + mu) + \frac{1}{f}\dot{u} \\ &= -\frac{\dot{f}}{f}f_d + \frac{1}{f}m\dot{u} \end{aligned} \tag{A.13a}$$

$$\begin{aligned} \frac{d^2}{dt^2}f_d &= \frac{d}{dt}(-\frac{\dot{f}}{f}f_d + \frac{1}{f}m\dot{u}) \\ &= \frac{1}{f}(-\ddot{f}f_d - 2\dot{f}\dot{f}_d + m\ddot{u}). \end{aligned} \tag{A.13b}$$

Consequently, it is also necessary to derive the 1st and 2nd order derivatives of thrust magnitude $f = \|f_r e_3 + mu\|$. That is

$$\begin{aligned} \frac{d}{dt}f &= \frac{d}{dt}\|f_r e_3 + mu\| \\ &= \frac{(f_r e_3 + mu)^T}{\|f_r e_3 + mu\|}m\dot{u} \\ &= \frac{(f_r e_3 + mu)^T}{f}m\dot{u} \end{aligned} \tag{A.14a}$$

$$\begin{aligned} \frac{d^2}{dt^2}f &= \frac{d}{dt}\frac{(f_r e_3 + mu)^T}{f}m\dot{u} \\ &= \frac{1}{f}(-\dot{f}^2 + m^2\dot{u}^T\dot{u} + (f_r e_3 + mu)^T m\ddot{u}) \end{aligned} \tag{A.14b}$$

.

Appendix B

Schema of the default PID controller in Parrot simulink package

See next page

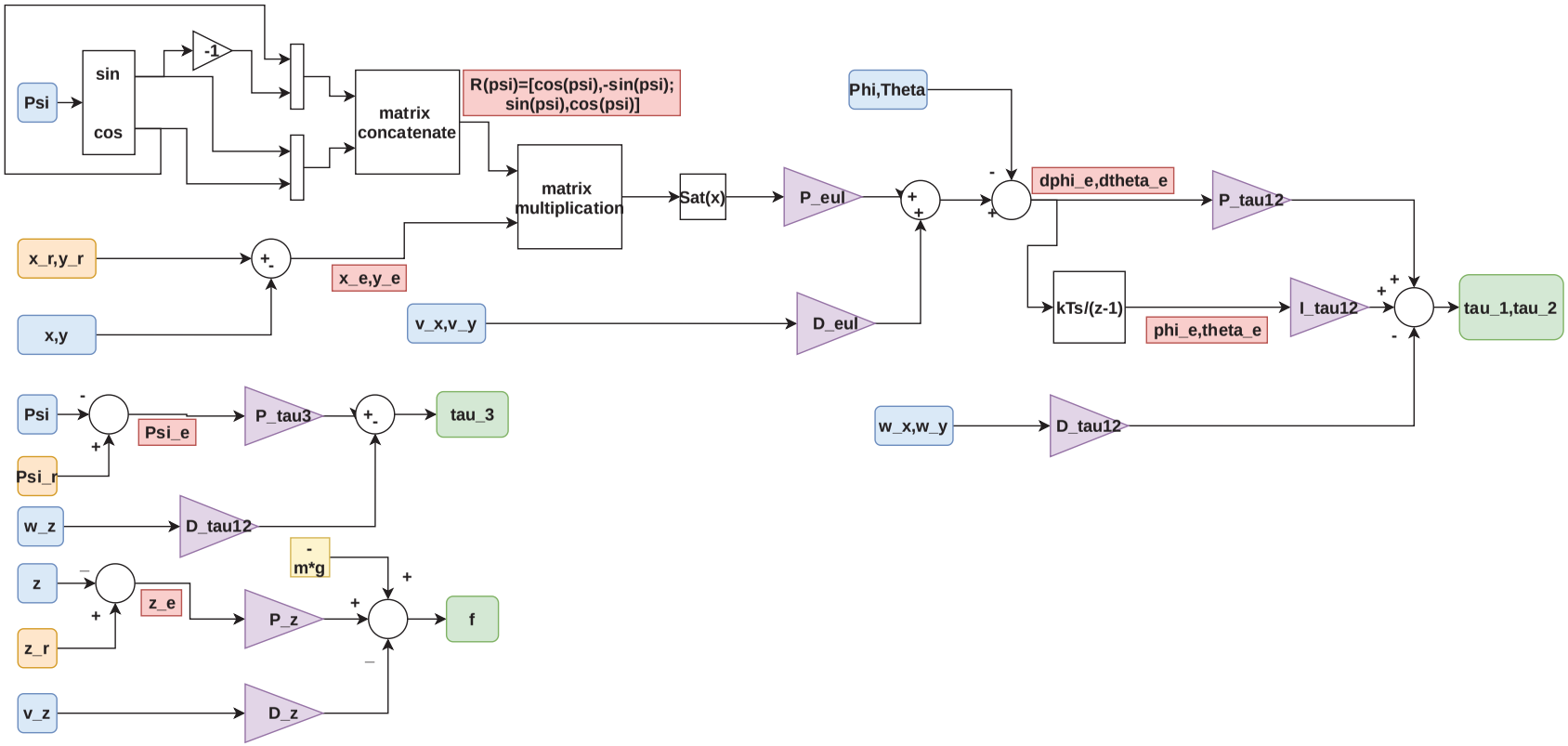


Figure B.1: Schema of the default PID controller in Parrot simulink package; where blue boxes are estimated states, orange boxes are references, red boxed are processing parameters, purple boxes are Proportional, Integral and Derivative gains w.r.t P_*, I_*, D_* and green boxes are outputs

Bibliography

[1] E. Adelson, C. Anderson, J. Bergen, P. Burt, and J. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[2] G. Adiv. Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):477–489, 1989.

[3] K. Åström. Control system design lecture notes for ME 155a. *Department of Mechanical and Environmental Engineering University of California Santa Barbara*, 333, 2002.

[4] G. Baker and J. Blackburn. *The pendulum: a case study in physics*. Oxford University Press, 2005.

[5] M. Bekar and Y. Yaylı. Involutions of complexified quaternions and split quaternions. *Advances in Applied Clifford Algebras*, 23(2):283–299, 2013.

[6] S. Bouabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2247–2252. IEEE, 2005.

[7] G. Brekelmans. Extended quadrotor dynamics: from simulations to experiments. Msc thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2019. DC2019. 090.

[8] P. J. Bristeau, F. Callou, D. Vissiere, and N. Petit. The navigation and control technology inside the AR. drone micro UAV. *IFAC Proceedings Volumes*, 44(1):1477–1484, 2011.

[9] C. Byrnes and A. Isidori. Local stabilization of minimum-phase nonlinear systems. *Systems & Control Letters*, 11(1):9–17, 1988.

[10] P. Castillo, R. Lozano, and A. Dzul. Stabilization of a mini-rotorcraft having four rotors. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2693–2698. IEEE, 2004.

[11] T. Curtright, D. B. Fairlie, and C. K. Zachos. A compact formula for rotations as spin matrix polynomials. *SIGMA. Symmetry, Integrability and Geometry: Methods and Applications*, 10:084, 2014.

[12] T. Derbanne. Method of evaluating the horizontal speed of a drone, in particular a drone capable of performing hovering flight under autopilot, July 30 2013. US Patent 8,498,447.

[13] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58, 01 2006.

[14] D. Eberly. Rotation representations and performance issues. *Magic Software: Chapel Hill, NC, USA*, 2002.

[15] W. L. Gaddy. Digital processing method and system for determination of optical flow, Apr. 29 2014. US Patent 8,712,095.

[16] M. Greiff. Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation, 2017. ISSN 0280-5316. Student Paper.

[17] D. Hoag. Apollo guidance and navigation: Considerations of apollo imu gimbal lock. *Canbridge: MIT Instrumentation Laboratory*, pages 1–64, 1963.

[18] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *2013 IEEE International Conference on Robotics and Automation*, pages 1736–1741. IEEE, 2013.

[19] B. Horn and B. Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.

[20] X. Huo, M. Huo, and H. R. Karimi. Attitude stabilization control of a quadrotor uav by using backstepping approach. *Mathematical Problems in Engineering*, 2014, 2014.

[21] N. Jeurgens. Identification and control implementation of an AR.Drone 2.0. Msc thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 01 2017. DC 2017.013.

[22] A. Kehlenbeck. *Quaternion-Based Control for Aggressive Trajectory Tracking with a Micro-Quadrotor UAV*. PhD thesis, University of Maryland, 01 2014.

[23] A. Krener and A. Isidori. Linearization by output injection and nonlinear observers. *Systems & Control Letters*, 3(1):47–52, 1983.

[24] B. Y. Lee, H. I. Lee, and M. J. Tahk. Analysis of adaptive control using on-line neural networks for a quadrotor UAV. In *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pages 1840–1844. IEEE, 2013.

[25] T. Lee, M. Leok, and N. McClamroch. *Global formulations of Lagrangian and Hamiltonian dynamics on manifolds*. Springer, 2017.

[26] E. Lefeber, M. Greiff, and A. Robertsson. *Filtered Output Feedback Tracking Control of a Quadrotor UAV*. Number DC 2020.053 in DC Reports. Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, May 2020.

[27] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*. Vancouver, British Columbia, 1981. https://www.ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_2/lucas_bruce_d_1981_2.pdf.

[28] J. Lugo and A. Zell. Framework for autonomous on-board navigation with the AR. Drone. *Journal of Intelligent & Robotic Systems*, 73(1-4):401–412, 2014.

[29] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19(3):20–32, 2012.

[30] Matlab. Parrot minidrones support from simulink, 2020. https://nl.mathworks.com/hardware-support/parrot-minidrones.html, Last accessed on 2020-6-15.

[31] Matlab. Aerospace toolbox, 2020. https://nl.mathworks.com/products/aerospace-toolbox.html, Last accessed on 2020-6-15.

[32] J. Mebius. Derivation of the Euler-Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations. *arXiv preprint math/0701759*, 2007.

[33] H. Parwana and M. Kothari. Quaternions and attitude representation. *arXiv preprint arXiv:1708.08680*, 2017.

[34] S. Poljak and J. Rohn. Checking robust nonsingularity is NP-hard. *Mathematics of Control, Signals and Systems*, 6(1):1–9, 1993.

[35] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[36] M. Rischmuller and F. D'haeyer. Method of piloting a multiple rotor rotary-wing drone to follow a curvilinear turn, June 25 2013. US Patent 8,473,125.

[37] H. Seydoux, F. Callou, and M. Babel. Altitude estimator for a rotary-wing drone with multiple rotors, Mar. 24 2015. US Patent 8,989,924.

[38] S. van den Eijnden. Cascade based tracking control of quadrotors. Msc thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2017. DC 2017.012.

[39] A. Visioli. *Practical PID control*. Springer Science & Business Media, 2006.

[40] S. Widnall. Lecture l3-vectors, matrices and coordinate transformations. https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec03.pdf, 2009. Lecture note.

[41] A. Zulu and S. John. A review of control algorithms for autonomous quadrotors. *arXiv preprint arXiv:1602. 02622*, 2016.