Image-Based Visual Servoing of a Quadrotor Using Model Predictive Control

by

Huaiyuan Sheng

B.Eng., University of Victoria, 2017

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

Image-Based Visual Servoing of a Quadrotor Using Model Predictive Control

by

Huaiyuan Sheng

B.Eng., University of Victoria, 2017

Supervisory Committee

Dr. Yang Shi, Supervisor

(Department of Mechanical Engineering)

Dr. Ben Nadler, Departmental Member

(Department of Mechanical Engineering)

Supervisory Committee

Dr. Yang Shi, Supervisor

(Department of Mechanical Engineering)

Dr. Ben Nadler, Departmental Member

(Department of Mechanical Engineering)

ABSTRACT

With numerous distinct advantages, quadrotors have found a wide range of applications, such as structural inspection, traffic control, search and rescue, agricultural surveillance, etc. To better serve applications in cluttered environment, quadrotors are further equipped with vision sensors to enhance their state sensing and environment perception capabilities. Moreover, visual information can also be used to guide the motion control of the quadrotor. This is referred to as visual servoing of quadrotor. In this thesis, we identify the challenging problems arising in the area of visual servoing of the quadrotor and propose effective control strategies to address these issues.

The control objective considered in this thesis is to regulate the relative pose of the quadrotor to a ground target using a limited number of sensors, e.g., a monocular camera and an inertia measurement unit. The camera is attached underneath the center of the quadrotor and facing down. The ground target is a planar object consisting of multiple points. The image features are selected as image moments defined

in a "virtual image plane". These image features offer an image kinematics that is independent of the tilt motion of the quadrotor. This independence enables the separation of the high level visual servoing controller design from the low level attitude tracking control.

A high-gain observer-based model predictive control (MPC) scheme is proposed in this thesis to address the image-based visual servoing of the quadrotor. The high-gain observer is designed to estimate the linear velocity of the quadrotor which is part of the system states. Due to a limited number of sensors on board, the linear velocity information is not directly measurable. The high-gain observer provides the estimates of the linear velocity and delivers them to the model predictive controller. On the other hand, the model predictive controller generates the desired thrust force and yaw rate to regulate the pose of the quadrotor relative to the ground target. By using the MPC controller, the tilt motion of the quadrotor can be effectively bounded so that the scene of the ground target is well maintained in the field of view of the camera. This requirement is referred to as visibility constraint. The satisfaction of visibility constraint is a prerequisite of visual servoing of the quadrotor.

Simulation and experimental studies are performed to verify the effectiveness of the proposed control strategies. Moreover, image processing algorithms are developed to extract the image features from the captured images, as required by the experimental implementation.

Table of Contents

List of Tables

List of Figures

Acronyms

UAV Unmanned Aerial Vehicle

IBVS Image-Based Visual Servoing

PBVS Position-Based Visual Servoing

MPC Model Predictive Control

NMPC Nonlinear Model Predictive Control

EMPC Explicit Model Predictive Control

PID Proportional-Integral-Derivative

GPS Global Positioning System

IMU Inertia Measurement Unit

FOV Field of View

COG Center of Gravity

SF Scaling Factor

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my supervisor, Professor Yang Shi for his guidance and patience throughout my study in the past two years. From him, I learned not only what constitutes of a capable researcher, but also how to become a responsible man. I am always inspired by his diligence and attitude towards work, and amazed by his strict attention to details. He is always there to encourage me with great patience whenever I feel frustrated; he is always there to support whenever I seek for help to carry on my project; he is always there to offer valuable suggestions to help me improve in all aspects. Moreover, I appreciate the opportunity that I was assigned to be a teaching assistant to work closely with him, from which experience I learned how to work promptly and professionally.

I would also like to thank the committee members, Professor Ben Nadler and Professor Panajotis Agathoklis, for their insightful comments.

I would like to extend my sincere gratitude to the group members at University of Victoria. Special thanks go to Qi Sun, who offered substantial help throughout the preparation and the realization of the experiments. I feel grateful for Kunwu Zhang, Jicheng Chen and Kenan Yong, who helped me revise my paper manuscript. I was motivated by their attitudes towards their works and benefited a lot from their constructive suggestions. I thank Zhang Zhang, Henglai Wei, Xinxin Shang, Chonghan Ma for their constructive suggestions and comments on my project. I also thank Zhuo Li and Tianxiang Lu, who spent their precious time helping me record research videos. I owe my thanks to Changxin Liu, Yuan Yang and Qian Zhang, who are always willing to answer fundamental questions from me regarding control theories, as well as Tianyu Tan, who is generous to offer rides to grab foods. Thank Chen Ma for her amazing pastry and secret recipes. Thank Xiang Sheng for sharing his knowledge of mechanical design and teaching me how to play foosball. Thank Dr.

Bingxian Mu for sharing his insights on job interview and industrial work.

Lastly, but most importantly, I would like to thank my parents. I could not have completed this thesis without their selfless love and support. Many thanks to my wife, Jinzhong Yang, who is always accompanying me making sure I am healthy and happy.

To my parents.

Chapter 1

Introduction

1.1 Literature Review

In this section, literature review is conducted to summarize the recent development in the field of visual servoing. First, the concept of visual servoing is introduced, with an emphasis on its classifications. Then, we discuss the visual servoing of fully-actuated systems, which lays the foundation for the following discussion of the under-actuated systems, e.g., quadrotors. Some challenging problems arising in the visual servoing of the quadrotor are pinpointed and the prevalent control methods are also introduced.

1.1.1 Visual Servoing

Visual servoing is referred to as using computer vision data as feedback in the motion control of robotics applications. Visual servoing is classified into *eye-to-hand configuration* and *eye-in-hand configuration* based on where the camera is attached. For *eye-to-hand configuration*, the camera is fixed in the world frame, and the motion of the robot and the target object is observed by the camera. For *eye-in-hand configuration*, the camera is attached to the robot end effector and only the motion of the

target is observed. For example, the motion capture systems commonly installed in the lab environment are typically *eye-to-hand configuration*, while a wheeled robot equipped with an on-board camera belongs to *eye-in-hand configuration*.

Visual servoing can be also classified into image-based visual servoing (IBVS) and position-based visual servoing (PBVS) based on types of features used in the feedback loop. IBVS directly uses the 2-D visual information in the image plane, while PBVS exploits the 3-D camera pose that is reconstructed from the image sequence [1].

These two methods have their own strength and weakness, respectively. PBVS minimizes the feature errors defined in the 3-D space, and accordingly gives the camera trajectory that is optimal in the 3-D space. Nevertheless, the resulting trajectory in the image space given by PBVS can be unsatisfactory; that is, the object scene could potentially leave the field of view (FOV) of the camera (see Figure 1.2(b)), which causes the failure of control input generation. Furthermore, the reconstruction of the camera pose requires the knowledge of camera's intrinsic parameters and the 3-D model of the target object [1, 2]. This reconstruction process is typically sensitive to the calibration errors and imaging noise, and thus PBVS is less robust than IBVS with the presence of calibration errors and imaging noise.

On the other hand, since IBVS minimizes the feature errors defined in the image plane, it favors the object scene staying in the FOV. Moreover, as mentioned before, IBVS is robust against camera calibration errors and imaging noise. Yet, the main drawback of IBVS, particularly when a monocular camera is employed, is the loss of the depth information caused by perspective projection. As shown in Figure 1.1, the same object scene can be produced either by a small object in close proximity or a large one in the distance, and hence it is impossible to determine the depth of the observed object. As a result, the depth of the target object needs to be estimated, as required by the control law [1]. Even though it is feasible to approximate the

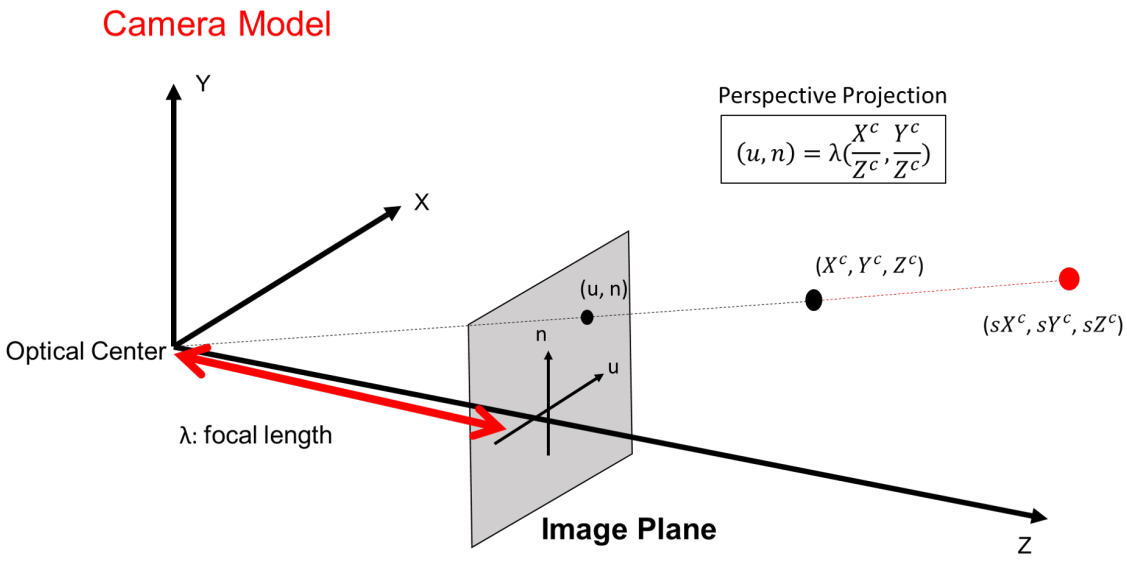
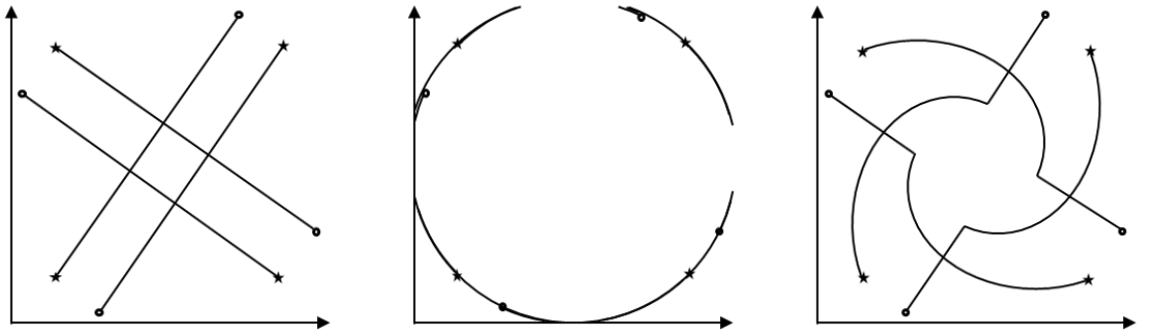


Figure 1.1: Perspective projection.

actual depth by the known desired value, this imperfect estimation may lead to the compromise of control performance [1].

In addition, when confronting large displacement, especially large rotation, IBVS may not provide good behavior of the 3-D space trajectory [1, 3]. As shown in Figure 1.2, where dots represent the initial point coordinates, and stars represent the desired ones. The initial camera pose deviates from the desired pose by 160° about the optical axis. In contrast to PBVS which results in a pure rotation motion to minimize error of the camera pose (see Figure 1.2(b)), IBVS causes the camera to retreat along the optical axis so that the point features are following straight lines to their desired configuration (see Figure 1.2(a)). The phenomenon of camera retreat should be alleviated to avoid reaching the joint/workspace limits of the robotic platforms.

Various hybrid approaches have been proposed to combine the strength of the two methods while compensating their shortcomings. In [4], "2 1/2-D visual servoing" is proposed to use 2-D image features and several 3-D parameters as feedback.



(a) IBVS. (b) PBVS. (c) Switched visual servoing.

Figure 1.2: Point coordinate trajectories in the field of view [3].

More specifically, the 3-D parameters are the rotation matrix and the direction of the translation between the current and the desired camera pose. These information are estimated by the homography method, which necessitates feature matching between the initial scene and the target scene. Moreover, the rotational control loop is decoupled from the translational one, and global asymptotic stability (GAS) can be attained by using this control scheme. A switching scheme is proposed in [3] to address the camera retreat induced by IBVS, and the feature loss induced by PBVS. The proposed switched mechanism is a state dependent switch, which triggers IBVS when the image trajectory approaches boundaries of the FOV, and triggers PBVS when the camera reaches boundaries of the robot workspace (see Figure 1.2(c)). As a result, the switching scheme allows the visibility constraint and the joint/workspace limit constraints to be simultaneously satisfied.

Moreover, advanced 2-D features are exploited to decouple the system dynamics and simplify the controller design. In [5] and [6], the authors propose six image moments that are invariant to a variety of 2-D transformation, such as translation, rotation, and scale. These invariance properties give rise to fully partitioned visual servoing such that each image moment corresponds to one degree of freedom of the robot. The resulting controller provides satisfactory behavior of both image trajectory

and 3-D trajectory and an enlarged domain of convergence as compared to using simple 2-D features, i.e., point coordinates.

1.1.2 Image-Based Visual Servoing of Fully-Actuated Systems

For IBVS of fully-actuated systems, e.g., manipulators, the "look-and-move" control structure is widely adopted [7]. As shown in Figure 1.3, there exist two control loops: the *outer-loop* and the *inner-loop*. In the *outer-loop*, an image-based visual servoing controller receives the vision feedback and outputs the desired joint trajectories, whereas in the *inner-loop*, a joint controller receives the desired joint trajectories and tracks these signals by generating proper actuator level commands, e.g., joint torques. Since the two controllers can be designed separately, this structure simplifies the controller design and accounts for the different sampling rates of the vision sensor and the joint sensor [7]. Furthermore, since many robots have the interface for receiving joint position or joint velocity commands, the look-and-move structure demonstrates good applicability.

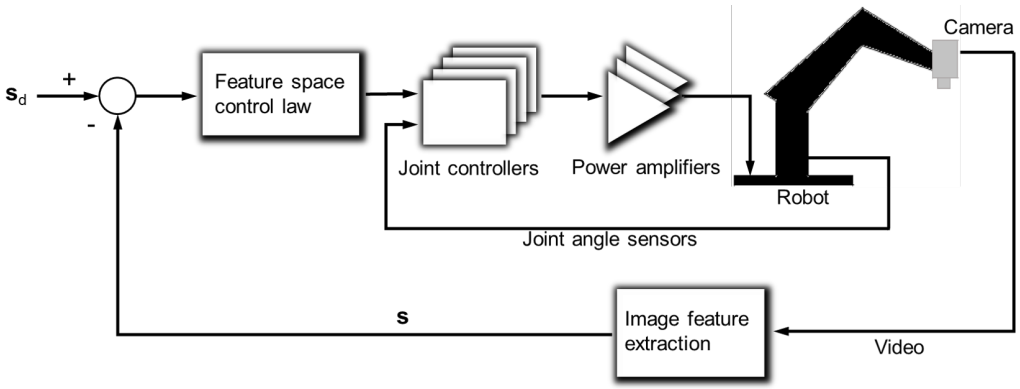


Figure 1.3: Image-based look-and-move structure [7].

In addition, the typical image features used in the IBVS of fully-actuated systems are point coordinates [8, 9, 10, 11]. Moreover, the visual servoing controller solely

considers the kinematic model of the robot and generates the desired Cartesian/joint velocities, while a dynamics inversion-based controller generates the actuator level commands, e.g., joint torques, to track the desired velocities.

1.1.3 Image-Based Visual Servoing of a Quadrotor

In Section 1.1.2, we discussed the application of visual servoing in the field of fully-actuated robotic platforms, where the dynamics of the platforms can be neglected, and only the kinematic model is considered in the design of visual servoing control law. The underlying reason for it is that the desired velocity commands generated by the visual servoing controller can be well tracked by the inner-loop dynamics under a high-gain control law. However, this assumption may not hold when confronting high speed task and under-actuated platforms [12].

Quadrotor is a typical type of under-actuated systems for which the number of actuators is less than degrees-of-freedom (DOF) of the system. With four motors actuating six DOF, the quadrotor is not capable of tracking every six dimensional reference velocity. As a result, the approaches used for fully-actuated systems [8, 9, 10, 11] cannot be directly applied to quadrotors.

However, since the quadrotor dynamics possesses a cascade triangular structure [13], the dual-loop control structure can be adopted by incorporating the translational dynamics into the outer-loop and considering the thrust magnitude and attitude as the control inputs of the outer-loop dynamics. Meanwhile, the rotational dynamics is handled by an inner-loop attitude tracking controller, which evaluates the required torques to track the desired attitude signals. The dual-loop scheme for the motion control of a quadrotor is illustrated in Figure 1.4. The outer-loop controller and the inner-loop controller can be designed separately, and yet the stability of the entire system is proven by exploiting the theories of cascade systems [14].

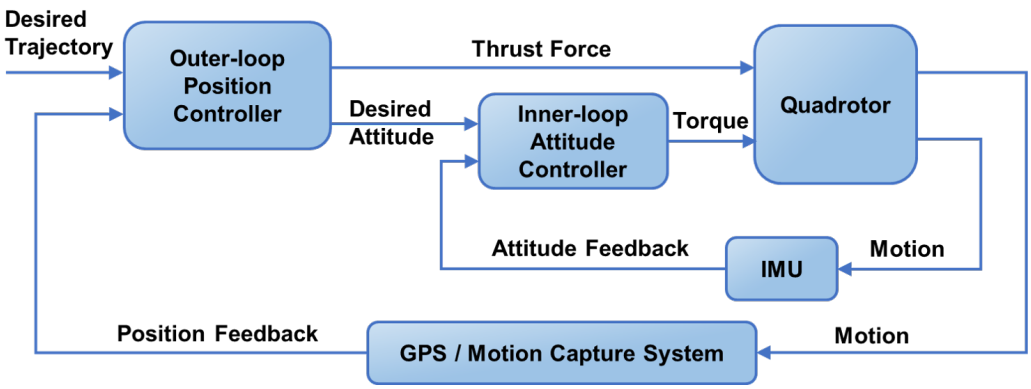


Figure 1.4: Dual-loop scheme for the motion control of a quadrotor.

Nevertheless, the dual-loop structure cannot be directly applied in the visual servoing of the quadrotor for the reason that the cascade triangular form is destroyed by the perspective projection [13]. To address this issue, researchers have put efforts in finding proper image features that can recover the cascade triangular form of the system dynamics.

Image Feature Selection

The two most popular methods proposed in the literature are spherical projection approach [13, 15] and virtual camera approach [16, 17, 18]. Spherical projection approach is firstly proposed by Hamel *et al.* [13] in 2002. Centroids defined in a spherical camera space are proposed to preserve the passivity-like property of the original quadrotor dynamics and recover the cascade triangular structure. The passivity-like property refers to that the norms of the image features are only functions of position such that their derivatives are not affected by the angular motion of the quadrotor [19]. Despite that global asymptotic stability can be attained by using this method, the resulting interaction matrix is ill-conditioned and causes slow convergence in the direction of focal axis [19]. To address this issue, the authors of [20] propose to properly scale the image features so that convergence rate among degrees of freedom can

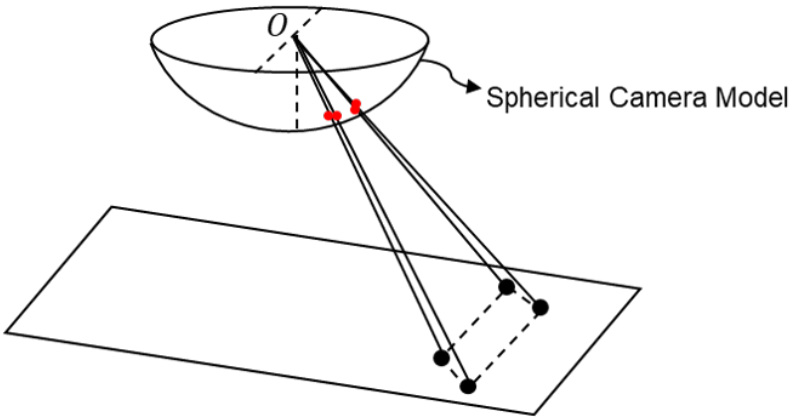


Figure 1.5: Spherical projection.

be balanced.

On the other hand, virtual camera approach defines a "virtual image plane" that is always parallel to the level ground. The object scene obtained through the perspective projection is reprojected onto this plane to remove the effects of the tilt of the quadrotor. As a result, image features defined in the virtual image plane give rise to the image kinematics that is independent of the roll and pitch motion of the quadrotor. This independence enables the recovery of the cascade structure of the overall dynamics. The virtual camera approach is firstly proposed by Jabbari *et al.* [21] in 2013. By re-defining the image moments [5, 6] in the virtual image plane, a simple and decoupled system dynamics can be obtained. Backstepping is employed in the controller design, and the control inputs are derived up to the angular velocity level to avoid unnecessary complexity, and then a low level high-gain controller is designed to track the angular velocity. This work is further extended in [22] to consider a moving target object. In [23], Xie *et al.* also explicitly consider system uncertainties and measurement bias. Furthermore, asymptotic stability of the dual-loop control structure in the application of IBVS of the quadrotor is developed in [24].

Explicit Consideration of Visibility Constraint

As discussed before, fruitful results have been witnessed in the IBVS of the quadrotor. However, the fulfillment of "visibility constraint" during the IBVS of the quadrotor remains as an open problem. In the course of visual servoing, the object of interest is required to stay in the field of view (FOV) of the camera, otherwise it will cause failure of the control input generation. This requirement is referred to as "visibility constraint". It is worth noting that the visibility constraint can be explicitly addressed by model predictive control when the robot is fully-actuated [11, 25]. Since point coordinates are used as the image features, the visibility constraint can be formulated as an output constraint.

However, for the IBVS of under-actuated robots, the dynamics is required to be taken into account, making the fulfillment of the visibility constraint even more challenging. Moreover, since advanced image features such as centroids and moments are used, it is nontrivial to formulate the visibility constraint as state or output constraints [18]. Given these facts, input saturation control laws [16, 17] are proposed in the literature to address this issue. In [16], an input saturated law is proposed to sufficiently bound the reference roll and pitch to alleviate the feature loss. It adopts a dual-loop control structure and focuses on the regulation of the translational dynamics in the outer-loop. To ensure the boundedness of the rotational dynamics, the authors of [17] propose the combination of an outer-loop bounded-input controller and an inner-loop saturated controller. The overall system is proved to be asymptotically stable by using the theory of cascade systems developed in [14]. On the other hand, path planning of image trajectories proves to be effective for fulfilling the visibility constraint [18]. The optimal paths of image moments are generated by minimizing the fourth order time derivative of the paths which are characterized by polynomials. When the object scene is about to leave the FOV, the planning time will be extended

to generate more intermediate waypoints, thus resulting in mild roll/pitch motion. In short, saturating roll/pitch angles is an effective technique to improve the visibility, even though it does not guarantee that the target object stays in the FOV for all scenarios.

Estimation of Linear Velocity

Early works [13, 21] assume that the linear velocity measurements of the quadrotor are available via either a global positioning system (GPS) or a motion capture system. However, GPS signals are not available in indoor or cluttered urban area, and the bandwidth is not high enough for the stabilization of the quadrotor [26]. Motion capture systems, e.g., OptiTrack [27], require an expensive multi-camera system and severely limit the workspace of the robot.

Alternatively, an inertia measurement unit (IMU) and a monocular camera form a compact sensor suite that is adequate for full state estimation and automatic flight [28]. The inertia measurement unit is a combination of accelerometers, gyrometers and magnetometers. The data from different sensors are fused to provide reliable attitude and angular velocity information of the platform. A vision system serves as a good complement to the IMU to estimate the position and linear velocity [26]. Using this sensor suite to estimate the full states of the system is referred to as visual-inertia odometry [28] in the literature. The visual-inertia odometry reconstructs the camera pose from the sensor data, and therefore it is similar to the aforementioned position-based visual servoing. Even though this technique appears to be promising, the performance of state estimator highly depends on the initialization and camera calibration [28].

On the other hand, inspired by vision of insects, optical flow can be used to estimate the linear velocity [26, 29]. However, the computation of optical flow involves

the numerical differentiation of the point coordinates, and thus the results are noisy.

Other approaches incorporate the estimation of the linear velocity into the controller design. Since these methods exploit the features defined in the image place, they inherit from IBVS the robustness to the imaging noise and calibration errors [30]. These methods can be further classified into observer-based control and output feedback control, based on whether the explicit estimate of the linear velocity has been used in the controller design [15].

Observer designs are proposed in [30, 31] to estimate the linear velocity by exploiting the image kinematics of the spherical centroids. Nevertheless, these observers require the knowledge of the depth information, which is typically available from a stereo vision system. This requirement is relaxed in [32], where the image moments defined in the virtual image plane are employed, and only the desired depth value is required. One the other hand, output feedback control constructs an "implicit observer" that does not provide an explicit estimate of the state, but contains auxiliary states to facilitate the design of controller [15, 33]. In [15], the requirement of accelerometers and magnetometers is removed, and in [33], system uncertainties and measurement bias are explicitly considered.

Prevalent Control Methods

In this section, we introduce two prevalent control methods that are widely applied in the field of visual servoing of robots. The first method is **backstepping**, which is a recursive design procedure derived from the constructed Lyapunov function. Compared with feedback linearization, backstepping avoids the cancellation of useful nonlinearities and allows for additional nonlinear terms to improve the transient performance [33]. The application of backstepping typically requires the system to be in a strict

feedback form [34], which possesses a cascade triangular structure as follows:

$$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\zeta}_1) \tag{1.1}$$

$$\dot{\boldsymbol{\zeta}}_1 = \boldsymbol{\zeta}_2 \tag{1.2}$$

$$...$$

$$\dot{\boldsymbol{\zeta}}_m = \boldsymbol{u} \tag{1.3}$$

where ($\boldsymbol{z}$, $\boldsymbol{\zeta}_1$, ... , $\boldsymbol{\zeta}_m$) are a series of vectors denoting state variables, $\boldsymbol{\mathcal{G}}$ is a function of state variables, and $\boldsymbol{u}$ is the control input.

First, the state $\boldsymbol{\zeta}_1$ is considered as "virtual control input" of (1.1), and a state feedback control law $\boldsymbol{\zeta}_1 = \boldsymbol{\chi}_1(\boldsymbol{z})$ is designed such that $\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\chi}_1(\boldsymbol{z}))$ is asymptotically stable with respect to the origin $\boldsymbol{z} = \boldsymbol{0}$. Next, an error $\boldsymbol{e}_1$ is defined as $\boldsymbol{e}_1 = \boldsymbol{\zeta}_1 - \boldsymbol{\chi}_1(\boldsymbol{z})$, and (1.1) and (1.2) can be rewritten as:

$$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\chi}_1(\boldsymbol{z}) + \boldsymbol{e}_1) \tag{1.4}$$

$$\dot{\boldsymbol{e}}_1 = \boldsymbol{\zeta}_2 - \dot{\boldsymbol{\chi}}_1(\boldsymbol{z}) \tag{1.5}$$

Note that $\dot{\boldsymbol{\chi}}_1(\boldsymbol{z})$ can be explicitly evaluated as:

$$\dot{\boldsymbol{\chi}}_1(\boldsymbol{z}) = \frac{\partial \boldsymbol{\chi}_1(\boldsymbol{z})}{\partial \boldsymbol{z}} \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\zeta}_1) \tag{1.6}$$

We can define that $\boldsymbol{\nu} = \boldsymbol{\zeta}_2 - \dot{\boldsymbol{\chi}}_1(\boldsymbol{z})$, and (1.4) and (1.5) can be further reduced to:

$$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\chi}_1(\boldsymbol{z}) + \boldsymbol{e}_1) \tag{1.7}$$

$$\dot{\boldsymbol{e}}_1 = \boldsymbol{\nu} \tag{1.8}$$

where $\boldsymbol{\nu}$ is considered as the control inputs of the above cascade system. A state

feedback control law $\boldsymbol{\nu} = \boldsymbol{\chi}_2(\boldsymbol{z}, \boldsymbol{e}_1)$ can be designed such that (1.7) and (1.8) are asymptotically stable with respect to the origin ($\boldsymbol{z} = \mathbf{0}, \boldsymbol{e}_1 = \mathbf{0}$). Accordingly, the second virtual control input $\boldsymbol{\zeta}_2$ is determined as:

$$\begin{aligned}\boldsymbol{\zeta}_2 &= \boldsymbol{\nu} + \dot{\boldsymbol{\chi}}_1(\boldsymbol{z}) \\ &= \boldsymbol{\chi}_2(\boldsymbol{z}, \boldsymbol{e}_1) + \frac{\partial \boldsymbol{\chi}_1(\boldsymbol{z})}{\partial \boldsymbol{z}} \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\zeta}_1) \\ &= \boldsymbol{\chi}_2(\boldsymbol{z}, \boldsymbol{\zeta}_1 - \boldsymbol{\chi}_1(\boldsymbol{z})) + \frac{\partial \boldsymbol{\chi}_1(\boldsymbol{z})}{\partial \boldsymbol{z}} \boldsymbol{\mathcal{G}}(\boldsymbol{z}, \boldsymbol{\zeta}_1)\end{aligned} \tag{1.9}$$

The above procedure is implemented recursively until the expression of $\boldsymbol{u}$ is derived. Every time a lower level dynamics is taken into account, the Lyapunov function needs to be augmented to incorporate the corresponding error variables, and the design of the following virtual input needs to guarantee that the derivative of the Lyapunov function is negative definite along the closed-loop system trajectories.

As discussed before, spherical projection approach and virtual camera approach are proposed in the literature to decouple the roll/pitch motion from the image kinematics, so that the cascade triangular structural property is recovered and backstepping can be applied to tackle the regulation of such system. By using backstepping, the stability of the closed-lop system is guaranteed along the derivation of the control law. Backstepping is a popular method applied in the visual servoing of the quadrotor, and there exist a number of works explicitly considering the visibility constraint [16, 17, 18] and the linear velocity estimation [15, 30, 31, 32, 33].

On the other hand, **model predictive control** (MPC) is also adopted in many works to address the system constraints appearing in the visual servoing of robots. As suggested by its name, model predictive control relies on the dynamic model of the plant, and provides control actions that take account of future behavior of the plant. As shown in Figure 1.6, the future outputs of the plant are predicted based on the plant model and the current output, and then the controller evaluates a sequence

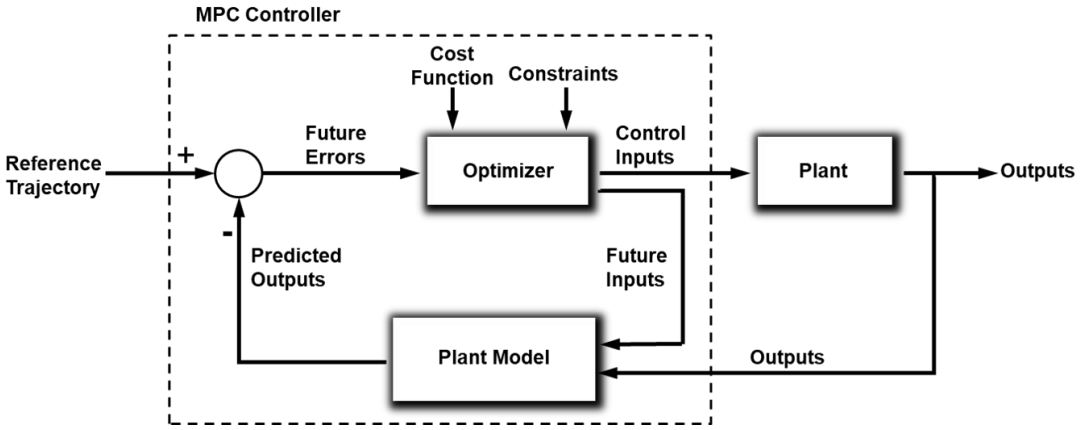


Figure 1.6: Basic structure of model predictive control.

of control inputs that minimizes the cost function while conforming to the specified constraints. The cost function is typically a weighted sum of squares of output tracking errors and control efforts, while constraints can be either imposed on the control inputs or on the predicted future states. The control inputs are implemented until the next sampling instant, when a new sequence of control inputs is obtained based on the updated output measurement.

Since MPC has inherent advantages of accounting for system constraints, many works [8, 9, 10, 11, 25] employ MPC to address joint limits, actuator saturation and visibility constraint. In [25], the visibility constraint is explicitly addressed but only the kinematic model of the camera is considered. This work is extended to fully-actuated systems, such as manipulators [8, 9], ground robots [10], and underwater vehicles [11]. However, very few works apply MPC in the area of visual servoing of under-actuated systems, e.g., quadrotors, to address the visibility constraint.

On the other hand, model predictive control requires the full state information to initialize the prediction. However, the linear velocity of the quadrotor, as part of the states, is not directly available due to the use of a minimal sensor suite. Thus, an observer is needed to provide the estimation of the linear velocity, as required by the MPC controller. It would be convenient if we can design the observer and

the controller separately. Despite that there is no general "separation principle" for nonlinear systems, Atassi and Khalil present and prove a separation principle for a class of nonlinear system using the high-gain observer [35]. Furthermore, the authors in [36] state that the high-gain observer can be used with a variety of NMPC control schemes to achieve semi-global practical stability. That is, for initial conditions in any compact set contained in the region of attraction of the NMPC controller, the system states can enter any small set containing the origin, if the high-gain observer is fast enough and the sampling frequency is sufficiently high [36]. Moreover, this work is extended in [37], which states that, to allow the separate design of the NMPC controller and the observer, the observer error needs be made sufficiently small in a sufficiently fast manner. Besides high-gain observers, other observers satisfying this requirement are moving horizon estimators, sliding mode observers, etc. Particularly, compared to the moving horizon estimator [38], which requires solving an optimization problem in real-time [37], the high-gain observer offers relatively reduced computational load.

1.2 Contributions

As discussed before, IBVS of the quadrotor remains as an active area, and further improvements can be expected in fulfilling the visibility constraint and linear velocity estimation. In this thesis, we propose control strategies that improve the visibility performance and require a minimum number of sensors. The contributions of the thesis can be summarized as follows.

- Image features are deliberately selected to decouple the system dynamics and enable the simplification of the controller design.
- A dual-loop control structure is adopted such that the translational visual servoing control law can be designed independent of the lower level attitude tracking

control.

- A high-gain observer is proposed to address the unavailability of the linear velocity when a minimal sensor suite is utilized.
- A nonlinear model predictive controller is proposed to improve the visibility during visual servoing by exploiting its inherent advantages in fulfilling state/input constraints.
- A roll/pitch constraints adjustment law is proposed to automate the adjustment of the constraints based on the point coordinates feedback, thus reducing the conservativeness induced by constant constraints.
- An alternative explicit MPC controller is proposed to alleviate the online computation load while improving the real-time performance.
- Experiments are conducted to verify the effectiveness of the proposed control schemes, which extends the application of MPC to IBVS of the quadrotor.

1.3 Thesis Organization

The remainder of the thesis is organized as follows:

Chapter 2 illustrates the system modelling, which includes the reference coordinate frames, the kinematics and dynamics of the quadrotor, as well as the kinematics of a number of image features, ranging from point coordinates to image moments.

Chapter 3 presents the main work of this thesis: A high-gain observer-based NMPC controller along with a roll/pitch constraints adjustment law, and

an explicit MPC controller that offers faster online implementation. Moreover, a proportional-derivative controller is elaborated to demonstrate a feasible solution for the inner-loop attitude tracking control. Simulation results are provided to show the validity of the proposed control schemes.

Chapter 4 starts with introducing the overall set-up of the experiments. Then, the image feature extraction process is elaborated. Lastly, the experimental results are presented to evaluate the performance of the proposed control strategies.

Chapter 5 concludes the thesis and highlights potential future work.

Chapter 2

System Modelling

2.1 Overview

In this chapter, the equations governing the dynamics of the image-based visual servoing of a quadrotor are presented. We first introduce the necessary coordinate frames, with respect to which all the equations are defined. Then, we present a nonlinear model that describes the kinematics and dynamics of the quadrotor in the 3-D space. Next, the image kinematics of a number of image features is derived, ranging from point coordinates to image moments. Lastly, the model for the image-based visual servoing of the quadrotor is shown to be comprised of the aforementioned image kinematics and part of the quadrotor dynamics in the 3-D space.

2.2 Reference Coordinate Frames

Four reference coordinate frames are defined in this section to describe the IBVS model [16]. Inertia frame $\mathbf{N} = \{O_n, n_1, n_2, n_3\}$ is fixed on earth with its basis n_1, n_2, n_3 orienting north, east, and down, respectively. This frame is used to describe the location of the ground target and the flight trajectory of the quadrotor. The second

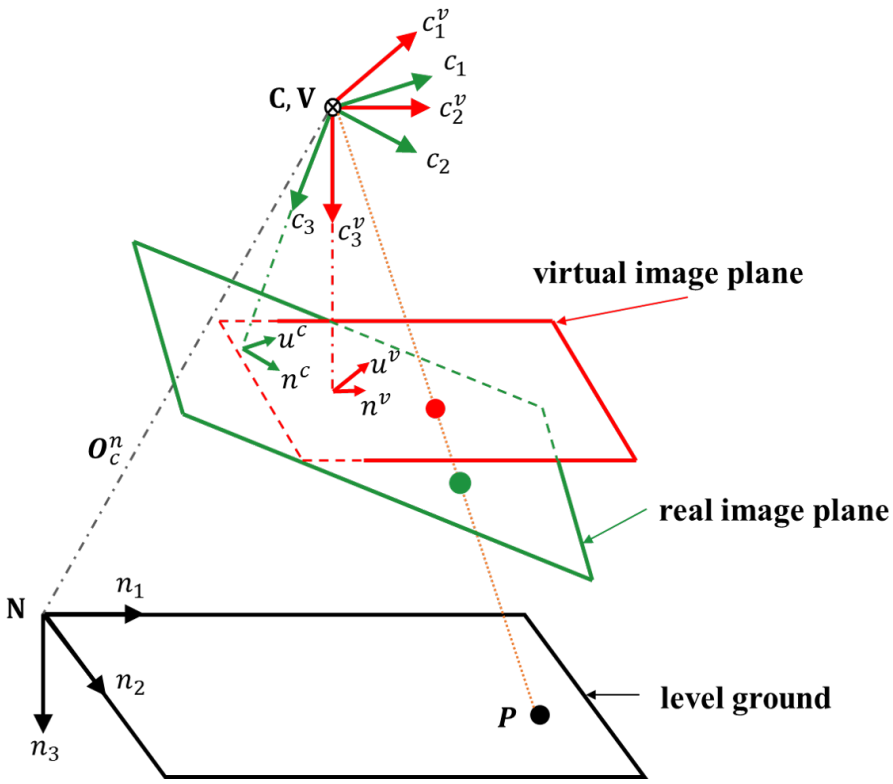


Figure 2.1: Reference coordinate frames.

coordinate frame is the body fixed frame $\mathbf{B} = \{O_b, b_1, b_2, b_3\}$, where the origin O_b is located at the center of gravity (COG) of the quadrotor, and b_1, b_2 and b_3 point forward, right and down with respect to the quadrotor. The third frame is the camera frame $\mathbf{C} = \{O_c, c_1, c_2, c_3\}$, which is rigidly attached to the camera. O_c is located at the optical center of the camera, and c_3 is aligned with camera's optical axis and perpendicular to the lens. In case that the camera is attached underneath the center of the quadrotor, it can be assumed that $\mathbf{B} = \mathbf{C}$ by neglecting the displacement between O_b and O_c. This assumption enables the simplification of the model. To avoid confusion, only $\mathbf{C}$ is employed in the following analysis. Lastly, the fourth coordinate frame is the virtual camera frame $\mathbf{V} = \{O_v, c_1^v, c_2^v, c_3^v\}$ whose origin O_v coincides with O_c. c_3^v is considered as the optical axis of the virtual camera, and it is parallel to n_3. The inertia frame $\mathbf{N}$, camera frame $\mathbf{C}$ and virtual camera frame $\mathbf{V}$

are illustrated in Figure 2.1.

2.2.1 Euler Angles and Rotation Matrices

The orientation relationship between two coordinate frames can be characterized by a rotation matrix. For example, the orientation of **C** with respect to **N** is represented by the rotation matrix $\boldsymbol{R}_C^N$. $\boldsymbol{R}_C^N$ is parametrized by Euler angles $\{\psi, \theta, \phi\}$ which are referred to as the yaw, pitch and roll of the quadrotor, respectively. The Euler angles are illustrated in Figure 2.2, in which $X'Y'Z'$ and $X''Y''Z''$ are intermediate frames obtained after the first and second elementary rotation, respectively. The orientation of **C** is obtained by a rotation of **N** about its Z axis (n_3) by ψ, followed by two rotations about the Y' and X'' axes of the intermediate frames by θ and ϕ, respectively.

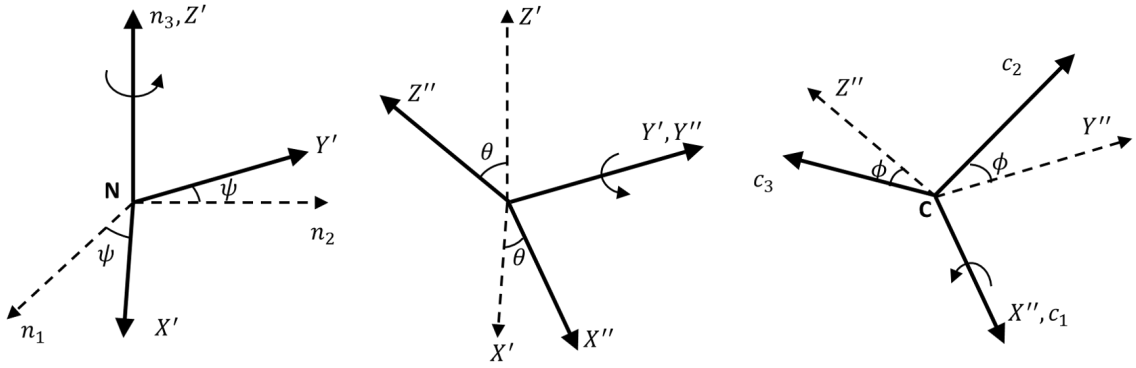


Figure 2.2: Definition of Euler angles.

$\boldsymbol{R}_C^N$ is the rotation matrix from **C** to **N** and calculated as [39]:

$$\begin{aligned}
\boldsymbol{R}_C^N &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \\
&= \begin{bmatrix} \mathrm{c}_\psi & -\mathrm{s}_\psi & 0 \\ \mathrm{s}_\psi & \mathrm{c}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}_\theta & 0 & \mathrm{s}_\theta \\ 0 & 1 & 0 \\ -\mathrm{s}_\theta & 0 & \mathrm{c}_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \mathrm{s}_\phi & \mathrm{c}_\phi \end{bmatrix} \\
&= \begin{bmatrix} \mathrm{c}_\theta\mathrm{c}_\psi & \mathrm{s}_\theta\mathrm{s}_\phi\mathrm{c}_\psi - \mathrm{s}_\psi\mathrm{c}_\phi & \mathrm{s}_\theta\mathrm{c}_\phi\mathrm{c}_\psi + \mathrm{s}_\psi\mathrm{s}_\phi \\ \mathrm{c}_\theta\mathrm{s}_\psi & \mathrm{s}_\theta\mathrm{s}_\phi\mathrm{s}_\psi + \mathrm{c}_\psi\mathrm{c}_\phi & \mathrm{s}_\theta\mathrm{c}_\phi\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{c}_\theta\mathrm{s}_\phi & \mathrm{c}_\theta\mathrm{c}_\phi \end{bmatrix}
\end{aligned} \tag{2.1}$$

where $\boldsymbol{R}_z$, $\boldsymbol{R}_y$, $\boldsymbol{R}_x$ denote the rotation matrices for the three successive elementary rotations, and the notations $\mathrm{s}_{(\cdot)}$, $\mathrm{c}_{(\cdot)}$ are shorthand forms of $\sin(\cdot)$ and $\cos(\cdot)$, respectively.

Since frame **V** has no roll and pitch motion but the same yaw as **C**, the rotation matrix from **V** to **N** is expressed as

$$\boldsymbol{R}_V^N = \boldsymbol{R}_z(\psi) \tag{2.2}$$

Based on (2.1) and (2.2), the rotation matrix from **C** to **V** is derived as:

$$\boldsymbol{R}_C^V = (\boldsymbol{R}_V^N)^{\mathrm{T}}\boldsymbol{R}_C^N = \boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \tag{2.3}$$

2.3 Equations of Motion of the Quadrotor

Equations governing the motion of the quadrotor are given by [40]:

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{v}^n \tag{2.4}$$

$$\dot{\boldsymbol{v}}^n = \boldsymbol{F}^n/m + g\boldsymbol{E}_3 \tag{2.5}$$

$$\dot{\boldsymbol{R}}_C^N = \boldsymbol{R}_C^N \left[\boldsymbol{\Omega}^c\right]_\times \tag{2.6}$$

$$\dot{\boldsymbol{\Omega}}^c = -\boldsymbol{I}^{-1} \left[\boldsymbol{\Omega}^c\right]_\times \boldsymbol{I}\boldsymbol{\Omega}^c + \boldsymbol{I}^{-1}\boldsymbol{\tau}^c \tag{2.7}$$

where $\boldsymbol{\zeta} = [x, y, z]^{\mathrm{T}}$ is the displacement of the COG of the quadrotor expressed in $\mathbf{N}$; $\boldsymbol{v}^n$ is the linear velocity of the quadrotor expressed in $\mathbf{N}$; $\boldsymbol{F}^n$ is the total thrust force generated by the four propellers expressed in $\mathbf{N}$; m is the mass of the quadrotor; g is the gravitational constant; $\boldsymbol{E}_3 = [0, 0, 1]^{\mathrm{T}}$; $\boldsymbol{\Omega}^c = [\Omega_1^c, \Omega_2^c, \Omega_3^c]^{\mathrm{T}}$ denotes the angular velocity of frame $\mathbf{C}$ relative to $\mathbf{N}$ expressed in $\mathbf{C}$;

$$\left[\boldsymbol{\Omega}^c\right]_\times = \begin{bmatrix} 0 & -\Omega_3^c & \Omega_2^c \\ \Omega_3^c & 0 & -\Omega_1^c \\ -\Omega_2^c & \Omega_1^c & 0 \end{bmatrix},$$

where $\times$ denotes cross product operation, and $[a]_\times b = a \times b$; $\boldsymbol{I}$ is the moment of inertia of the quadrotor expressed in $\mathbf{C}$; $\boldsymbol{I}^{-1}$ is the inverse of $\boldsymbol{I}$; $\boldsymbol{\tau}^c$ is the resultant torque generated by the propellers expressed in $\mathbf{C}$.

Equations (2.4) and (2.5) are the linear kinematics and translational dynamics of the quadrotor, respectively. It is straightforward to derive them from Newton's second law. Equation (2.6) is the rotational kinematics and the detailed derivation is given in [41]. Since $\boldsymbol{R}_C^N$ is a matrix parameterized by the Euler angles, (2.6) describes the time evolution of Euler angles driven by the angular velocities. An alternative

expression of (2.6) is given by [42]:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{W}(\boldsymbol{\eta})\boldsymbol{\Omega}^c \tag{2.8}$$

where $\boldsymbol{\eta} = [\phi, \theta, \psi]^{\mathrm{T}}$, and

$$\boldsymbol{W}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & \mathrm{s}_\phi \mathrm{t}_\theta & \mathrm{c}_\phi \mathrm{t}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \mathrm{s}_\phi / \mathrm{c}_\theta & \mathrm{c}_\phi / \mathrm{c}_\theta \end{bmatrix}$$

where $\mathrm{t}_{(\cdot)}$ denotes $\tan(\cdot)$.

The derivation of (2.8) is given below. According to Figure 2.2, we have:

$$\boldsymbol{\Omega}^n = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \boldsymbol{R}_z(\psi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta) \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{2.9}$$

where $\boldsymbol{\Omega}^n$ is the angular velocity of frame $\mathbf{C}$ relative to $\mathbf{N}$ expresseed in $\mathbf{N}$, and therefore

$$\boldsymbol{\Omega}^n = \boldsymbol{R}_C^N \boldsymbol{\Omega}^c = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi)\boldsymbol{\Omega}^c$$

Substituting the above equation into (2.9) and multiplying both sides by $\boldsymbol{R}_x^{-1}(\phi)\boldsymbol{R}_y^{-1}(\theta)\boldsymbol{R}_z^{-1}(\psi)$,

we have:

$$\begin{aligned}\boldsymbol{\Omega}^c &= \boldsymbol{R}_x^{-1}(\phi)\boldsymbol{R}_y^{-1}(\theta)\boldsymbol{R}_z^{-1}(\psi)\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} + \boldsymbol{R}_x^{-1}(\phi)\boldsymbol{R}_y^{-1}(\theta)\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix} + \boldsymbol{R}_x^{-1}(\phi)\begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix} \\ &= \begin{bmatrix}1 & 0 & -\mathrm{s}_\theta\\ 0 & \mathrm{c}_\phi & \mathrm{c}_\theta\mathrm{s}_\phi\\ 0 & -\mathrm{s}_\phi & \mathrm{c}_\theta\mathrm{c}_\phi\end{bmatrix}\begin{bmatrix}\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix} \\ &= \boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}\end{aligned} \tag{2.10}$$

With reference to (2.8), $\boldsymbol{W}(\boldsymbol{\eta})$ is the inverse of $\boldsymbol{M}(\boldsymbol{\eta})$, i.e., $\boldsymbol{W}(\boldsymbol{\eta}) = \boldsymbol{M}^{-1}(\boldsymbol{\eta})$.

Equation (2.7) is the rotational dynamics of the quadrotor expressed in frame **C**, which can be derived from principle of angular momentum [43]. Writing the equation in frame **C** enables the moment of inertia $\boldsymbol{I}$ to be a constant matrix.

2.3.1 Force and Torque Calculation

As shown in Figure 2.3, the resultant force generated by the four propellers are along the $-c_3$ direction, and thus

$$\begin{aligned}\boldsymbol{F}^n &= -\boldsymbol{R}_C^N\boldsymbol{E}_3 T \\ &= -\boldsymbol{R}_C^N\boldsymbol{E}_3(f_1 + f_2 + f_3 + f_4)\end{aligned} \tag{2.11}$$

where T is the magnitude of $\boldsymbol{F}^n$, and f_1, f_2, f_3, f_4 are the magnitude of the thrust generated by each propeller. The torques generated by the propellers are expressed

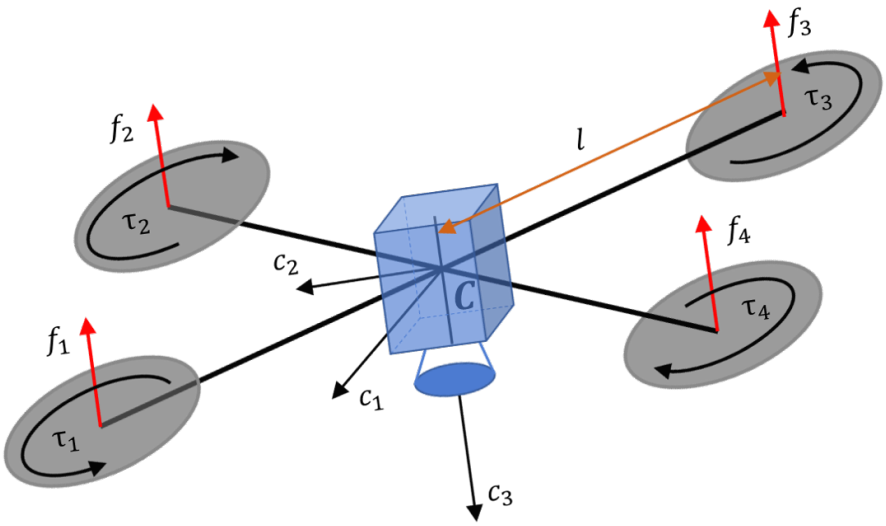


Figure 2.3: Thrusts and reaction torques applied on the quadrotor.

as:

$$\boldsymbol{\tau}^c = \begin{bmatrix} \tau_1^c \\ \tau_2^c \\ \tau_3^c \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} l(f_3 + f_4 - f_1 - f_2) \\ \frac{\sqrt{2}}{2} l(f_1 + f_4 - f_2 - f_3) \\ \tau_2 + \tau_4 - \tau_1 - \tau_3 \end{bmatrix} \tag{2.12}$$

where l is the arm length of the quadrotor, and τ_1, τ_2, τ_3, τ_4 are the reaction torques generated at each propeller. The thrust and reaction torque generated by one propeller are related to the its rotational speed by:

$$f_i = K_T w_i^2 \tag{2.13a}$$

$$\tau_i = K_\tau w_i^2 \tag{2.13b}$$

where K_T and K_τ are the aerodynamics constants dominated by the geometric properties of the propellers, and w_i denotes the propeller rotational speed for $1 \leq i \leq 4$.

According to (2.11)–(2.13), the following one-to-one mapping can be established.

$$\begin{bmatrix} T \\ \tau_1^c \\ \tau_2^c \\ \tau_3^c \end{bmatrix} = \begin{bmatrix} K_T & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} l K_T & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} l K_T & 0 \\ 0 & 0 & 0 & K_\tau \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}}_{\boldsymbol{\mathcal{M}}} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{2.14}$$

It is straightforward to verify that the matrix $\boldsymbol{\mathcal{M}}$ is nonsingular. That is to say, whenever a set of $(T, \tau_1^c, \tau_2^c, \tau_3^c)$ is given, the propeller speed $(w_1^2, w_2^2, w_3^2, w_4^2)$ can be correspondingly determined, provided that the aerodynamic constants have been accurately identified. Thanks to the existence of such one-to-one mapping, it is common to consider $(T, \tau_1^c, \tau_2^c, \tau_3^c)$ instead of $(w_1^2, w_2^2, w_3^2, w_4^2)$ as the control inputs, which simplifies the controller design.

2.4 Image Feature and Kinematics

The ground target considered in this thesis is a planar object consisting of multiple points. The coordinates of these points in the real image plane are first extracted from the captured images, and then re-projected onto the virtual image plane that is always parallel to the flat ground. In this section, we first derive the image kinematics of a single point in the real image plane. Then, we show its counterpart in the virtual image plane. Next, we give the definitions of image moments which are algebraic equations of multiple point coordinates in the virtual image plane. Finally, the kinematics of the image moments is derived based on the preceding derivations.

2.4.1 Kinematics of a Point in the Real Image Plane

Consider a static point P whose coordinates in $\mathbf{N}$ are denoted by $\boldsymbol{P}^n$. Then, its coordinates in frame $\mathbf{C}$ can be expressed as:

$$\boldsymbol{P}^c = \boldsymbol{R}_N^C[\boldsymbol{P}^n - \boldsymbol{O}_c^n] \tag{2.15}$$

where $\boldsymbol{P}^c = [p_x^c, p_y^c, p_z^c]^{\mathrm{T}}$, and $\boldsymbol{O}_c^n$ denotes the displacement of the origin of $\mathbf{C}$ expressed in $\mathbf{N}$. Based on the perspective projection equation [40], the projection of P on the real image plane can be obtained as

$$\begin{bmatrix} u^c \\ n^c \end{bmatrix} = \frac{\lambda}{p_z^c} \begin{bmatrix} p_x^c \\ p_y^c \end{bmatrix} \tag{2.16}$$

where λ is the focal length of the camera.

Taking the time derivative of (2.15), we have

$$\begin{aligned} \dot{\boldsymbol{P}}^c &= \dot{\boldsymbol{R}}_N^C[\boldsymbol{P}^n - \boldsymbol{O}_c^n] + \boldsymbol{R}_N^C[\dot{\boldsymbol{P}}^n - \dot{\boldsymbol{O}}_c^n] \\ &= -[\boldsymbol{\Omega}^c]_\times \boldsymbol{R}_N^C[\boldsymbol{P}^n - \boldsymbol{O}_c^n] + \boldsymbol{R}_N^C[\mathbf{0}_3 - \boldsymbol{v}^n] \\ &= -[\boldsymbol{\Omega}^c]_\times \boldsymbol{P}^c - \boldsymbol{v}^c \end{aligned} \tag{2.17}$$

where $\mathbf{0}_3 = [0, 0, 0]^{\mathrm{T}}$, and $\boldsymbol{v}^c = [v_1^c, v_2^c, v_3^c]^{\mathrm{T}}$ is the linear velocity of the quadrotor expressed in frame $\mathbf{C}$.

Taking the time derivative of (2.16), we have

$$\begin{cases} \dot{u}^c = \lambda \dfrac{\dot{p}_x^c p_z^c - p_x^c \dot{p}_z^c}{(p_z^c)^2} = \lambda \dfrac{\dot{p}_x^c}{p_z^c} - u^c \dfrac{\dot{p}_z^c}{p_z^c} \\ \dot{n}^c = \lambda \dfrac{\dot{p}_y^c p_z^c - p_y^c \dot{p}_z^c}{(p_z^c)^2} = \lambda \dfrac{\dot{p}_y^c}{p_z^c} - n^c \dfrac{\dot{p}_z^c}{p_z^c} \end{cases} \tag{2.18}$$

Substituting the expressions of $\dot{p}_x^c$, $\dot{p}_y^c$ and $\dot{p}_y^c$ given in (2.17) into the right-hand

side of (2.18), then point kinematics in the real image plane is obtained as:

$$\begin{bmatrix} \dot{u}^c \\ \dot{n}^c \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{\lambda}{p_z^c} & 0 & \frac{u^c}{p_z^c} & \frac{u^c n^c}{\lambda} & -\frac{\lambda^2+(u^c)^2}{\lambda} & n^c \\ 0 & -\frac{\lambda}{p_z^c} & \frac{n^c}{p_z^c} & \frac{\lambda^2+(n^c)^2}{\lambda} & -\frac{u^c n^c}{\lambda} & -u^c \end{bmatrix}}_{\boldsymbol{\mathcal{L}}_c} \begin{bmatrix} v_1^c \\ v_2^c \\ v_3^c \\ \Omega_1^c \\ \Omega_2^c \\ \Omega_3^c \end{bmatrix} \tag{2.19}$$

where the matrix $\boldsymbol{\mathcal{L}}_c$ is called the interaction matrix or image Jacobian matrix. It relates the velocity of the image projection of the point to the velocity of the camera. Notice that this interaction matrix is affected by the current image projection of the point, u^c and n^c, as well as the depth of the point with respect to the camera frame, p_z^c.

2.4.2 Kinematics of a Point in the Virtual Image Plane

Since the selected image features are defined in the virtual image plane instead of the real camera plane, the kinematics of a point in the virtual image plane needs to be derived. Then, based on the point kinematics, the kinematics of image moments is derived thereafter.

The point coordinates in the real image plane (u^c, n^c) are re-projected onto the virtual image plane by [40]:

$$\begin{bmatrix} u^v \\ n^v \\ \lambda \end{bmatrix} = \frac{\lambda}{\lambda_0} \boldsymbol{R}_C^V \begin{bmatrix} u^c \\ n^c \\ \lambda \end{bmatrix} \tag{2.20}$$

where λ_0 is defined as:

$$\lambda_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_C^V \begin{bmatrix} u^c \\ n^c \\ \lambda \end{bmatrix} \tag{2.21}$$

Notice that the defined projection enables the normal distance from the optical center O_v to the virtual image plane being equal to λ.

To derive the point kinematics in the virtual image plane, the same approach is taken as in (2.15)–(2.19). The coordinates of point P in frame $\mathbf{V}$ is expressed as:

$$\boldsymbol{P}^v = \boldsymbol{R}_N^V[\boldsymbol{P}^n - \boldsymbol{O}_c^n] \tag{2.22}$$

where $\boldsymbol{P}^v = [p_x^v, p_y^v, p_z^v]^{\mathrm{T}}$. The projection of P on the virtual image plane is expressed as

$$\begin{bmatrix} u^v \\ n^v \end{bmatrix} = \frac{\lambda}{p_z^v} \begin{bmatrix} p_x^v \\ p_y^v \end{bmatrix} \tag{2.23}$$

Taking the time derivative of (2.22), we have

$$\begin{aligned} \dot{\boldsymbol{P}}^v &= \dot{\boldsymbol{R}}_N^V[\boldsymbol{P}^n - \boldsymbol{O}_c^n] + \boldsymbol{R}_N^V[\dot{\boldsymbol{P}}^n - \dot{\boldsymbol{O}}_c^n] \\ &= -[\boldsymbol{\Omega}^v]_\times \boldsymbol{R}_N^V[\boldsymbol{P}^n - \boldsymbol{O}_c^n] + \boldsymbol{R}_N^V[\boldsymbol{0}_3 - \boldsymbol{v}^n] \\ &= -[\boldsymbol{\Omega}^v]_\times \boldsymbol{P}^v - \boldsymbol{v}^v \\ &= -\left[\dot{\psi}\boldsymbol{E}_3\right]_\times \boldsymbol{P}^v - \boldsymbol{v}^v \end{aligned} \tag{2.24}$$

where $\boldsymbol{\Omega}^v$ is the angular velocity of frame $\mathbf{V}$ with respect to $\mathbf{N}$ expressed in $\mathbf{V}$, and $\boldsymbol{v}^v = [v_1^v, v_2^v, v_3^v]^{\mathrm{T}}$ is the linear velocity of the quadrotor expressed in frame $\mathbf{V}$. $\boldsymbol{\Omega}^v = \dot{\psi}\boldsymbol{E}_3$ can be derived from (2.10) given that $\phi = 0$ and $\theta = 0$.

Taking the time derivative of (2.23), we have

$$\begin{cases} \dot{u}^v = \lambda \dfrac{\dot{p}_x^v p_z^v - p_x^v \dot{p}_z^v}{(p_z^v)^2} = \lambda \dfrac{\dot{p}_x^v}{p_z^v} - u^v \dfrac{\dot{p}_z^v}{p_z^v} \\ \dot{n}^v = \lambda \dfrac{\dot{p}_y^v p_z^v - p_y^v \dot{p}_z^v}{(p_z^v)^2} = \lambda \dfrac{\dot{p}_y^v}{p_z^v} - n^v \dfrac{\dot{p}_z^v}{p_z^v} \end{cases} \tag{2.25}$$

Substituting the expressions of $\dot{p}_x^v$, $\dot{p}_y^v$ and $\dot{p}_y^v$ given in (2.24) into the right-hand side of (2.25), then point kinematics in the virtual image plane is obtained as:

$$\begin{bmatrix} \dot{u}^v \\ \dot{n}^v \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{\lambda}{p_z^v} & 0 & \frac{u^v}{p_z^v} & n^v \\ 0 & -\frac{\lambda}{p_z^v} & \frac{n^v}{p_z^v} & -u^v \end{bmatrix}}_{\boldsymbol{\mathcal{L}}_v} \begin{bmatrix} v_1^v \\ v_2^v \\ v_3^v \\ \dot{\psi} \end{bmatrix} \tag{2.26}$$

where $\boldsymbol{\mathcal{L}}_v$ is the interaction matrix when the point coordinates are defined in the virtual image plane. Compared to (2.19), (2.26) is independent of the roll and pitch, as u^v and n^v are defined in the virtual image plane which does not have roll or pitch motion.

2.4.3 Image Moments

The selected image features are image moments defined in the virtual image plane, and they are functions of u^v and n^v. The image moments $\boldsymbol{s} = [s_1, s_2, s_3, s_4]^{\mathrm{T}}$ are

defined as [40]:

$$s_1 = s_3 \frac{u_g^v}{\lambda} \tag{2.27a}$$

$$s_2 = s_3 \frac{n_g^v}{\lambda} \tag{2.27b}$$

$$s_3 = \sqrt{\frac{a^*}{a}} \tag{2.27c}$$

$$s_4 = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{2.27d}$$

where $u_g^v = \frac{1}{K}\sum_{k=1}^{K} u_k^v$, $n_g^v = \frac{1}{K}\sum_{k=1}^{K} n_k^v$, u_k^v and n_k^v are the coordinates of the kth point in the virtual image plane, K is the number of the points contained in the target object, $\mu_{ij} = \sum_{k=1}^{K}(u_k^v - u_g^v)^i(n_k^v - n_g^v)^j$, $a = \mu_{20} + \mu_{02}$, a^* is the value of a measured at the pre-defined desired height, Z^*.

2.4.4 Kinematics of Image Moments

The kinematics of the image moments is given by [40]:

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \end{bmatrix} = \begin{bmatrix} -\frac{1}{Z^*} & 0 & 0 \\ 0 & -\frac{1}{Z^*} & 0 \\ 0 & 0 & -\frac{1}{Z^*} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^v \\ v_2^v \\ v_3^v \end{bmatrix} + \begin{bmatrix} s_2 \\ -s_1 \\ 0 \\ -1 \end{bmatrix} \dot{\psi} \tag{2.28}$$

where Z^* is the prescribed desired height.

Since s_1 and s_2 are functions of s_3, we start from the derivation of $\dot{s}_3$. We first

show that $s_3 = \sqrt{\frac{a^*}{a}} = \frac{Z}{Z^*}$.

$$\begin{aligned} a &= \mu_{20} + \mu_{02} \\ &= \sum_{k=1}^{K} (u_k^v - u_g^v)^2 + \sum_{k=1}^{K} (n_k^v - n_g^v)^2 \\ &= \sum_{k=1}^{K} (u_k^v - u_g^v)^2 + (n_k^v - n_g^v)^2 \\ &= \sum_{k=1}^{K} \left(\frac{\lambda(p_{x,k}^v - \bar{p}_x^v)}{p_{z,k}^v} \right)^2 + \left(\frac{\lambda(p_{y,k}^v - \bar{p}_y^v)}{p_{z,k}^v} \right)^2 \\ &= \frac{\lambda^2}{Z^2} \sum_{k=1}^{K} (p_{x,k}^v - \bar{p}_x^v)^2 + (p_{y,k}^v - \bar{p}_y^v)^2 \\ &= \frac{\lambda^2}{Z^2} \sum_{k=1}^{K} ||[p_{x,k}^v, p_{y,k}^v]^{\mathrm{T}} - [\bar{p}_x^v, \bar{p}_y^v]^{\mathrm{T}}||^2 \end{aligned} \tag{2.29}$$

where $\bar{p}_x^v = \frac{1}{K} \sum_{k=1}^{K} p_{x,k}^v$, $\bar{p}_y^v = \frac{1}{K} \sum_{k=1}^{K} p_{y,k}^v$, $p_{z,k}^v$ are the same for $1 \leq k \leq K$ and denoted by Z. $||\cdot||$ denotes the Euclidean norm of a vector. Note that the term $\sum_{k=1}^{K} ||[p_{x,k}^v, p_{y,k}^v]^{\mathrm{T}} - [\bar{p}_x^v, \bar{p}_y^v]^{\mathrm{T}}||^2$ is a constant as it measures the squared sum of the absolute distance from each individual point to the barycenter of the object. Since $Z^2 a$ is equal to a constant, we have:

$$s_3 = \sqrt{\frac{a^*}{a}} = \frac{Z}{Z^*} \tag{2.30}$$

It is straightforward to derive that

$$\dot{s}_3 = \frac{\dot{Z}}{Z^*} = -\frac{v_3^v}{Z^*} \tag{2.31}$$

Notice that the "−" sign in (2.31) results from the fact that the axis c_3^v orients downward.

To obtain the expressions of $\dot{s}_1$ and $\dot{s}_2$, we first derive the expressions of $\dot{u}_g^v$ and

$\dot{n}_g^v$ by substituting (2.26) into $u_g^v = \frac{1}{K}\sum_{k=1}^{K} u_k^v$ and $n_g^v = \frac{1}{K}\sum_{k=1}^{K} n_k^v$, respectively.

$$\begin{aligned}
\dot{u}_g^v &= \frac{1}{K}\sum_{k=1}^{K} \dot{u}_k^v \\
&= \frac{1}{K}\sum_{k=1}^{K} \left(\begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u_k^v}{Z} \end{bmatrix} \boldsymbol{v}^v + n_k^v \dot{\psi} \right) \\
&= \begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{1}{K}\sum_{k=1}^{K} \frac{u_k^v}{Z} \end{bmatrix} \boldsymbol{v}^v + \frac{1}{K}\sum_{k=1}^{K} n_k^v \dot{\psi} \\
&= \begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u_g^v}{Z} \end{bmatrix} \boldsymbol{v}^v + n_g^v \dot{\psi}
\end{aligned} \tag{2.32}$$

where $\boldsymbol{v}^v = [v_1^v, v_2^v, v_3^v]^{\mathrm{T}}$.

Similarly, we have

$$\dot{n}_g^v = \begin{bmatrix} 0 & -\frac{\lambda}{Z} & \frac{n_g^v}{Z} \end{bmatrix} \boldsymbol{v}^v - u_g^v \dot{\psi} \tag{2.33}$$

Using (2.27a), (2.30), (2.31), (2.32), we have

$$\begin{aligned}
\dot{s}_1 &= \frac{1}{\lambda}(\dot{s}_3 u_g^v + s_3 \dot{u}_g^v) \\
&= \frac{1}{\lambda}\left(-\frac{v_3^v}{Z^*} u_g^v + \frac{Z}{Z^*} \left(\begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u_g^v}{Z} \end{bmatrix} \boldsymbol{v}^v + n_g^v \dot{\psi} \right) \right) \\
&= -\frac{1}{Z^*} v_1^v + s_2 \dot{\psi}
\end{aligned} \tag{2.34}$$

Similarly, the expression of $\dot{s}_2$ in (2.28) can be obtained using (2.27b), (2.30), (2.31), (2.33).

s_4 can be simplified as:

$$\begin{aligned} s_4 &= \frac{1}{2}\arctan\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right) \\ &= \frac{1}{2}\arctan\left(\frac{2\sum_{k=1}^{K}(u_k^v-u_g^v)(n_k^v-n_g^v)}{\sum_{k=1}^{K}(u_k^v-u_g^v)^2-\sum_{k=1}^{K}(n_k^v-n_g^v)^2}\right) \\ &= \frac{1}{2}\arctan\left(\frac{2\frac{\lambda^2}{Z^2}\sum_{k=1}^{K}(p_{x,k}^v-\bar{p}_x^v)(p_{y,k}^v-\bar{p}_y^v)}{\frac{\lambda^2}{Z^2}\sum_{k=1}^{K}(p_{x,k}^v-\bar{p}_x^v)^2-\frac{\lambda^2}{Z^2}\sum_{k=1}^{K}(p_{y,k}^v-\bar{p}_y^v)^2}\right) \\ &= \frac{1}{2}\arctan\left(\frac{2\sum_{k=1}^{K}(p_{x,k}^v-\bar{p}_x^v)(p_{y,k}^v-\bar{p}_y^v)}{\sum_{k=1}^{K}(p_{x,k}^v-\bar{p}_x^v)^2-\sum_{k=1}^{K}(p_{y,k}^v-\bar{p}_y^v)^2}\right) \\ &= \alpha = -\psi+\beta \end{aligned} \tag{2.35}$$

where α is the angle from the axis c_1^v to the principal axis of the ground object [5]; ψ is the yaw angle; β is the angle from the axis n_1 to the principal axis (see Figure 2.4). Since the object is static with respect to frame $\mathbf{N}$, β is a constant.

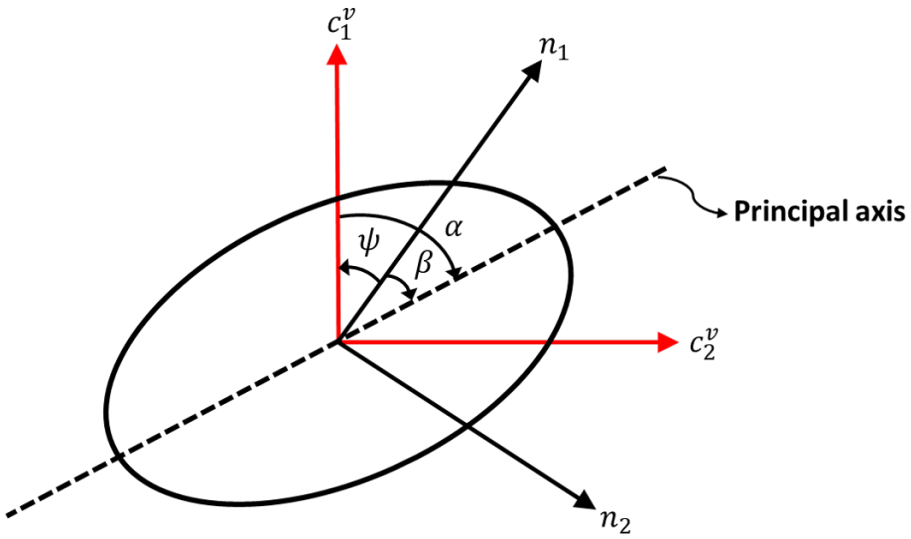


Figure 2.4: Geometrical explanation of image moment s_4.

Thus, it is straightforward to derive that

$$\dot{s}_4 = -\dot{\psi} \tag{2.36}$$

2.5 Equations of Image-Based Visual Servoing of a Quadrotor

In terms of IBVS of a quadrotor, the position information of the quadrotor is not directly measurable due to the absence of a global positioning system or a motion capture system. Therefore, the original linear kinematics of the quadrotor (2.4) is replaced by the kinematics of image features (2.28).

The model for the IBVS of the quadrotor is comprised of (2.28), (2.5), (2.6), and (2.7), as summarized below:

$$\dot{\boldsymbol{s}}_l = -\frac{1}{Z^*}\boldsymbol{v}^v + \boldsymbol{S}\dot{\psi} \tag{2.37a}$$

$$\dot{s}_4 = -\dot{\psi} \tag{2.37b}$$

$$\dot{\boldsymbol{v}}^v = -[\dot{\psi}\boldsymbol{E_3}]_\times \boldsymbol{v}^v - \boldsymbol{R}_C^V \boldsymbol{E}_3 T/m + g\boldsymbol{E}_3 \tag{2.37c}$$

$$\dot{\boldsymbol{R}}_C^N = \boldsymbol{R}_C^N \left[\boldsymbol{\Omega}^c\right]_\times \tag{2.37d}$$

$$\dot{\boldsymbol{\Omega}}^c = -\boldsymbol{I}^{-1}\left[\boldsymbol{\Omega}^c\right]_\times \boldsymbol{I}\boldsymbol{\Omega}^c + \boldsymbol{I}^{-1}\boldsymbol{\tau}^c \tag{2.37e}$$

where $\boldsymbol{s}_l = [s_1, s_2, s_3]^{\mathrm{T}}$; $\boldsymbol{S} = [s_2, -s_1, 0]^{\mathrm{T}}$.

Notice that (2.37c) is an equivalent expression of (2.5), but it is expressed in $\mathbf{V}$. The derivation of (2.37c) is shown below:

$$\begin{aligned}
&\dot{\boldsymbol{v}}^n = \boldsymbol{F}^n/m + g\boldsymbol{E}_3 \\
&\frac{d}{dt}(\boldsymbol{R}_V^N \boldsymbol{v}^v) = \boldsymbol{R}_V^N \boldsymbol{F}^v/m + g\boldsymbol{E}_3 \\
&\dot{\boldsymbol{R}}_V^N \boldsymbol{v}^v + \boldsymbol{R}_V^N \dot{\boldsymbol{v}}^v = \boldsymbol{R}_V^N \boldsymbol{F}^v/m + g\boldsymbol{E}_3 \\
&\boldsymbol{R}_V^N [\boldsymbol{\Omega}^v]_\times \boldsymbol{v}^v + \boldsymbol{R}_V^N \dot{\boldsymbol{v}}^v = \boldsymbol{R}_V^N \boldsymbol{F}^v/m + g\boldsymbol{E}_3
\end{aligned}$$

Multiplying both sides by $\boldsymbol{R}_N^V$, we have:

$$[\boldsymbol{\Omega}^v]_\times \boldsymbol{v}^v + \dot{\boldsymbol{v}}^v = \boldsymbol{F}^v/m + \boldsymbol{R}_N^V g \boldsymbol{E}_3$$

Then, rearrange the above equation and we have:

$$\begin{aligned}\dot{\boldsymbol{v}}^v &= -[\boldsymbol{\Omega}^v]_\times \boldsymbol{v}^v + \boldsymbol{F}^v/m + \boldsymbol{R}_N^V g \boldsymbol{E}_3 \\ &= -[\dot{\psi}\boldsymbol{E}_3]_\times \boldsymbol{v}^v - \boldsymbol{R}_C^V \boldsymbol{E}_3 T/m + g\boldsymbol{E}_3\end{aligned}$$

where $\boldsymbol{F}^v$ is the total thrust force generated by the four propellers expressed in frame $\mathbf{V}$. As mentioned before, $\boldsymbol{\Omega}^v = \dot{\psi}\boldsymbol{E}_3$ is derived from (2.10) given that $\phi = 0$ and $\theta = 0$. In addition, $\boldsymbol{R}_N^V$ is omitted in the final expression since $\boldsymbol{R}_N^V \boldsymbol{E}_3 = \boldsymbol{E}_3$.

2.6 Conclusion

In this chapter, the modelling of the IBVS of the quadrotor is illustrated. The virtual camera approach is adopted and the image features are selected as four image moments defined in the virtual camera plane. This deliberate selection of image features leads to an image kinematics that is independent of the tilt motion of the quadrotor, which contributes to a nicely decoupled dynamic model for the IBVS of the quadrotor.

Chapter 3

High-Gain Observer-Based Model Predictive Control

3.1 Overview

In this chapter, we first illustrate the dual-loop control structure which consists of an outer-loop controller and an inner-loop controller. Then, a high-gain observer-based nonlinear MPC controller is employed in the outer-loop to regulate the relative position and yaw of the quadrotor to the ground target while addressing:

1. the fulfillment of the visibility constraint during the visual servoing.
2. the unavailability of the linear velocity information due to the use of a minimal sensor suite.

Furthermore, an adjustment law is developed to automatically adjust the constraints on the roll/pitch angles to keep the target object within the FOV. Moreover, to reduce the online computational load and improve the real-time performance, an explicit MPC controller is designed such that the optimization problem can be solved

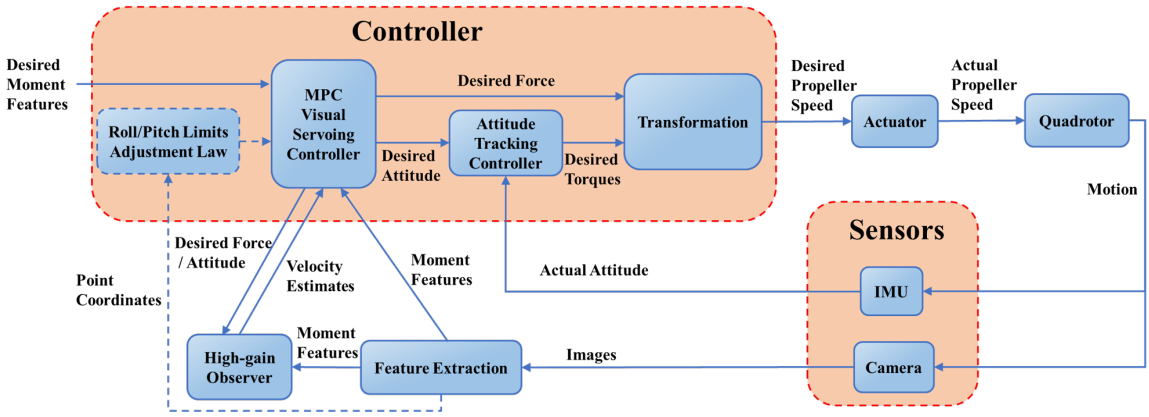


Figure 3.1: Dual-loop control structure.

offline. On the other hand, since the inner-loop rotational dynamics can be feedback linearized, a dynamics inversion-based proportional-derivative (PD) controller is accordingly designed to track the desired attitude signals received from the outer-loop controller. Lastly, simulation results are presented to show the validity of the proposed control algorithms.

3.2 Dual-Loop Control Structure

Instead of using vision feedback to directly generate actuator level commands, we adopt the dual-loop control structure (see Figure 3.1). The outer-loop visual servoing controller receives the extracted image features and outputs the desired thrust and attitude signals to the inner-loop, while the inner-loop attitude tracking controller tracks the desired attitude signals by outputting the desired torque to the quadrotor. The desired thrust and torque signals are then transformed into the propeller speed commands to actuate the quadrotor. The advantages of the dual-loop control structure lie in: First, it enables each loop effectively fully actuated even though the overall dynamics is under-actuated; Second, it naturally fits the scenario where the sampling rate of the outer-loop sensor (camera) is much slower than that of the inner-

loop sensor (IMU). Since the dual-loop structure allows for the separate design of the outer-loop controller from the inner-loop controller, we thus focus on the design of the visual servoing controller that regulates the outer-loop dynamics. A benchmark attitude tracking controller developed in [44] is used as the inner-loop controller.

3.3 High-Gain Observer

A high-gain observer is developed to estimate the linear velocity of the quadrotor, which is required by the design of the MPC controller.

3.3.1 Design Procedure

The observer design employs the outer-loop dynamics described by (2.37a) – (2.37c). First, an input transformation is performed to simplify the observer design. We define a variable $\mathbf{f}$ as

$$\mathbf{f} = g\boldsymbol{E}_3 - \boldsymbol{R}_C^V \boldsymbol{E}_3 T/m \tag{3.1}$$

where $\mathbf{f} = [\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3]^{\mathrm{T}}$. Then, (2.37c) can be expanded as

$$\begin{bmatrix} \dot{v}_1^v \\ \dot{v}_2^v \\ \dot{v}_3^v \end{bmatrix} = \begin{bmatrix} 0 & \dot{\psi} & 0 \\ -\dot{\psi} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^v \\ v_2^v \\ v_3^v \end{bmatrix} + \begin{bmatrix} \mathrm{f}_1 \\ \mathrm{f}_2 \\ \mathrm{f}_3 \end{bmatrix} \tag{3.2}$$

Equations (2.28) and (3.2) together form a nonlinear system that belongs to a class of nonlinear systems discussed in [45], and accordingly the high-gain observer design technique can be applied. To better show the design procedure, (2.28) and (3.2) are

written into the form $\dot{\boldsymbol{x}} = \boldsymbol{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{u})$ as follows:

$$\begin{aligned}
\dot{x}_1 &= -\frac{1}{Z^*}x_5 + x_2u_1 & \dot{x}_2 &= -\frac{1}{Z^*}x_6 + x_1u_1 \\
\dot{x}_3 &= -\frac{1}{Z^*}x_7 & \dot{x}_4 &= -u_1 \\
\dot{x}_5 &= x_6u_1 + u_2 & \dot{x}_6 &= -x_5u_1 + u_3 \\
\dot{x}_7 &= u_4
\end{aligned} \tag{3.3}$$

where $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]^{\mathrm{T}} = [s_1, s_2, s_3, s_4, v_1^v, v_2^v, v_3^v]^{\mathrm{T}}$, $\boldsymbol{u} = [u_1, u_2, u_3, u_4]^{\mathrm{T}} = [\dot{\psi}, \mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3]^{\mathrm{T}}$.

State equations (3.3) can be further transformed into three chains of double integrators by a mapping $\boldsymbol{\xi} = \boldsymbol{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{u})$:

$$\begin{aligned}
\xi_1 &= x_1 & \xi_2 &= -\frac{1}{Z^*}x_5 + x_2u_1 \\
\xi_3 &= x_2 & \xi_4 &= -\frac{1}{Z^*}x_6 + x_1u_1 \\
\xi_5 &= x_3 & \xi_6 &= -\frac{1}{Z^*}x_7
\end{aligned} \tag{3.4}$$

It is straightforward to verify that $\dot{\xi}_1 = \xi_2$, $\dot{\xi}_3 = \xi_4$, and $\dot{\xi}_5 = \xi_6$. This is an observable form that is sufficient for the application of high-gain observer [45]. Note that the equation in (3.3) related to $\dot{x}_4$ is omitted , as it does not contribute to the estimation of the linear velocity.

By differentiating (3.4) and employing (3.3), the transformed state equations are obtained as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\Delta}(\boldsymbol{\xi}, \boldsymbol{u}) \tag{3.5}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{\xi} \tag{3.6}$$

where $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6]^{\mathrm{T}}$; $\boldsymbol{y} = [s_1, s_2, s_3]^{\mathrm{T}}$; $\boldsymbol{A} = \mathrm{blockdiag}(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3)$, $\boldsymbol{B} =$

blockdiag($\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3$), $\boldsymbol{C}$ = blockdiag($\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3$), where blockdiag($\cdot$) denotes forming a matrix by block diagonalizing the matrices within the parentheses,

$$\boldsymbol{A}_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad \boldsymbol{B}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \boldsymbol{C}_i = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

for $1 \leq i \leq 3$; $\boldsymbol{\Delta}(\boldsymbol{\xi}, \boldsymbol{u}) = [\Delta_1, \Delta_2, \Delta_3]^{\mathrm{T}}$, where

$$\begin{aligned} \Delta_1 &= \frac{\partial \xi_2}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{1}{Z^*}u_2 + 2\xi_4 u_1 + \xi_1 u_1^2 \\ \Delta_2 &= \frac{\partial \xi_4}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{1}{Z^*}u_3 - 2u_1\xi_2 + \xi_3 u_1^2 \\ \Delta_3 &= \frac{\partial \xi_6}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{1}{Z^*}u_4 \end{aligned} \tag{3.7}$$

Accordingly, the high-gain observer is designed as:

$$\dot{\hat{\boldsymbol{\xi}}} = \boldsymbol{A}\hat{\boldsymbol{\xi}} + \boldsymbol{B}\boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}}, \boldsymbol{u}) + \boldsymbol{H}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{\xi}}) \tag{3.8}$$

where $\hat{\boldsymbol{\xi}}$ is the estimate of $\boldsymbol{\xi}$; $\boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}}, \boldsymbol{u})$ is the estimate of the nonlinear term $\boldsymbol{\Delta}(\boldsymbol{\xi}, \boldsymbol{u})$; $\boldsymbol{H}$ = blockdiag($\boldsymbol{H}_1, \boldsymbol{H}_2, \boldsymbol{H}_3$), where

$$\boldsymbol{H}_i = \begin{bmatrix} \alpha_1^i/\epsilon \\ \alpha_2^i/\epsilon^2 \end{bmatrix} \tag{3.9}$$

for $1 \leq i \leq 3$. $\boldsymbol{H}$ is the observer gain matrix, and ϵ is the high gain parameter that is required to be sufficiently small. Positive constants α_j^i need to be selected such that the roots of the quadratic polynomials

$$n^2 + \alpha_1^i n + \alpha_2^i = 0 \tag{3.10}$$

are all in the left-half plane, for $i = 1, 2, 3$.

Given $\hat{\boldsymbol{\xi}}$ is obtained by (3.8), the estimate of state $\hat{\boldsymbol{x}}$ is then determined by the inverse of map (3.4), i.e., $\hat{\boldsymbol{x}} = \boldsymbol{\mathcal{H}}^{-1}(\hat{\boldsymbol{\xi}}, \boldsymbol{u})$. $\boldsymbol{\mathcal{H}}^{-1}(\hat{\boldsymbol{\xi}}, \boldsymbol{u})$ is given as:

$$\begin{aligned} \hat{x}_1 &= \hat{\xi}_1 \qquad & \hat{x}_5 &= -Z^*(\hat{\xi}_2 - \hat{\xi}_3 u_1) \\ \hat{x}_2 &= \hat{\xi}_3 \qquad & \hat{x}_6 &= -Z^*(\hat{\xi}_4 + \hat{\xi}_1 u_1) \\ \hat{x}_3 &= \hat{\xi}_5 \qquad & \hat{x}_7 &= -Z^* \hat{\xi}_6 \end{aligned} \tag{3.11}$$

where $[\hat{x}_5, \hat{x}_6, \hat{x}_7]^{\mathrm{T}}$ is the estimate of the linear velocity of the quadrotor.

3.3.2 Theoretical Analysis

We define the scaled estimation error as:

$$\boldsymbol{\eta} = \left[\frac{\xi_1 - \hat{\xi}_1}{\epsilon^1}, \frac{\xi_2 - \hat{\xi}_2}{\epsilon^0}, \frac{\xi_3 - \hat{\xi}_3}{\epsilon^1}, \frac{\xi_4 - \hat{\xi}_4}{\epsilon^0}, \frac{\xi_5 - \hat{\xi}_5}{\epsilon^1}, \frac{\xi_6 - \hat{\xi}_6}{\epsilon^0} \right]^{\mathrm{T}} \tag{3.12}$$

and it can be written in a compact form:

$$\boldsymbol{\eta} = \boldsymbol{D}(\epsilon)^{-1}(\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}) \tag{3.13}$$

where $\boldsymbol{D}(\epsilon) = \mathrm{blockdiag}(\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$, with $\boldsymbol{D}_i = \begin{bmatrix} \epsilon & 0 \\ 0 & 1 \end{bmatrix}$, for $i = 1, 2, 3$.

The dynamics of the scaled error can be derived according to (3.8), (3.5), (3.6), (3.13):

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{D}^{-1}(\dot{\boldsymbol{\xi}} - \dot{\hat{\boldsymbol{\xi}}}) \\ &= \boldsymbol{D}^{-1}(\boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\Delta}(\boldsymbol{\xi}, \boldsymbol{u}) - \boldsymbol{A}\hat{\boldsymbol{\xi}} - \boldsymbol{B}\boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}}, \boldsymbol{u}) - \boldsymbol{H}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{\xi}})) \\ &= \boldsymbol{D}^{-1}((\boldsymbol{A} - \boldsymbol{H}\boldsymbol{C})\boldsymbol{D}\boldsymbol{\eta} + \boldsymbol{B}(\boldsymbol{\Delta}(\boldsymbol{\xi}, \boldsymbol{u}) - \boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}}, \boldsymbol{u}))) \end{aligned} \tag{3.14}$$

Multiplying both sides by ϵ, we have:

$$\begin{aligned}\epsilon\dot{\boldsymbol{\eta}} &= \epsilon\boldsymbol{D}^{-1}(\boldsymbol{A}-\boldsymbol{HC})\boldsymbol{D}\boldsymbol{\eta} + \epsilon\boldsymbol{D}^{-1}\boldsymbol{B}(\boldsymbol{\Delta}(\boldsymbol{\xi},\boldsymbol{u}) - \boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}},\boldsymbol{u})) \\ &= \boldsymbol{A}_0\boldsymbol{\eta} + \epsilon\boldsymbol{B}\boldsymbol{g}(\boldsymbol{\xi},\hat{\boldsymbol{\xi}},\boldsymbol{u})\end{aligned} \tag{3.15}$$

where $\boldsymbol{A}_0 = \epsilon\boldsymbol{D}^{-1}(\boldsymbol{A}-\boldsymbol{HC})\boldsymbol{D}$, $\boldsymbol{g}(\boldsymbol{\xi},\hat{\boldsymbol{\xi}},\boldsymbol{u}) = \boldsymbol{\Delta}(\boldsymbol{\xi},\boldsymbol{u}) - \boldsymbol{\Delta}_0(\hat{\boldsymbol{\xi}},\boldsymbol{u})$. It is easy to verify that $\boldsymbol{D}^{-1}\boldsymbol{B} = \boldsymbol{B}$, and $\boldsymbol{A}_0$ is a constant matrix, which is independent of the high gain parameter ϵ. $\boldsymbol{A}_0$ =blockdiag$(\boldsymbol{A}_{01}, \boldsymbol{A}_{02}, \boldsymbol{A}_{03})$, where $\boldsymbol{A}_{0i} = \begin{bmatrix} -\alpha_1^i & 1 \\ -\alpha_2^i & 0 \end{bmatrix}$ for $i = 1, 2, 3$. The eigenvalues of $\boldsymbol{A}_0$ are the union of the eigenvalues of $\boldsymbol{A}_{01}$, $\boldsymbol{A}_{02}$ and $\boldsymbol{A}_{03}$, and the eigenvalues of $\boldsymbol{A}_{0i}$ can be determined by solving the characteristic equation (3.10).

Since the selection of α^i enables all the eigenvalues of $\boldsymbol{A}_0$ are placed in the left half plane, $\boldsymbol{A}_0$ is thus Hurwitz. It is observed that a small ϵ can attenuate the effects of $\boldsymbol{g}(\boldsymbol{\xi},\hat{\boldsymbol{\xi}},\boldsymbol{u})$ and increase the convergence rate of the state estimation [46].

3.4 Nonlinear Model Predictive Control

3.4.1 Problem Formulation

To apply MPC, the outer-loop dynamics described by (3.3) is discretized by Euler forward approximation with a sampling period T_s:

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{\mathcal{F}}(\boldsymbol{x}_k,\boldsymbol{u}_k)T_s \\ &= \boldsymbol{\mathcal{F}}_d(\boldsymbol{x}_k,\boldsymbol{u}_k)\end{aligned} \tag{3.16}$$

where $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1}$ denote the states $\boldsymbol{x}$ at time step k and $k+1$, respectively, and $\boldsymbol{u}_k$ denotes the control inputs $\boldsymbol{u}$ at time step k.

The expanded form of (3.16) is given as:

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ s_{3,k+1} \\ s_{4,k+1} \\ v^v_{1,k+1} \\ v^v_{2,k+1} \\ v^v_{3,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \dot{\psi}_k T_s & 0 & 0 & -T_s/Z^* & 0 & 0 \\ -\dot{\psi}_k T_s & 1 & 0 & 0 & 0 & -T_s/Z^* & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -T_s/Z^* \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \dot{\psi}_k T_s & 0 \\ 0 & 0 & 0 & 0 & -\dot{\psi}_k T_s & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{1,k} \\ s_{2,k} \\ s_{3,k} \\ s_{4,k} \\ v^v_{1,k} \\ v^v_{2,k} \\ v^v_{3,k} \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -T_s & 0 & 0 & 0 \\ 0 & T_s & 0 & 0 \\ 0 & 0 & T_s & 0 \\ 0 & 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} \dot{\psi}_k \\ \mathrm{f}_{1,k} \\ \mathrm{f}_{2,k} \\ \mathrm{f}_{3,k} \end{bmatrix} \tag{3.17}$$

Notice that the above discrete-time model is nonlinear, as it involves products of state and input variables on the right hand side of the equation.

The regulation task can be formulated as a nonlinear programming problem [11] as follows:

$$\min_{\boldsymbol{U}_k} J(\boldsymbol{x}_k, \boldsymbol{U}_k) \tag{3.18}$$

with

$$J(\boldsymbol{x}_k, \boldsymbol{U}_k) = \sum_{j=1}^{N_p} (\overline{\boldsymbol{x}}_{j,k} - \boldsymbol{x}_{ref})^{\mathrm{T}} \boldsymbol{Q} (\overline{\boldsymbol{x}}_{j,k} - \boldsymbol{x}_{ref}) + \sum_{j=0}^{N_p-1} (\overline{\boldsymbol{u}}_{j,k})^{\mathrm{T}} \boldsymbol{R} (\overline{\boldsymbol{u}}_{j,k}) \tag{3.19}$$

subject to

$$\overline{\boldsymbol{x}}_{j+1,k} = \boldsymbol{\mathcal{F}}_d(\overline{\boldsymbol{x}}_{j,k}, \overline{\boldsymbol{u}}_{j,k}), \qquad j = 0, 1, ..., N_p - 1 \tag{3.20}$$

$$\overline{\boldsymbol{x}}_{0,k} = x_k, \tag{3.21}$$

$$\overline{\boldsymbol{u}}_{j,k} \in \mathbb{U}, \qquad j = 0, 1, ..., N_p - 1 \tag{3.22}$$

where $J(\boldsymbol{x}_k, \boldsymbol{U}_k)$ is the cost function, $\boldsymbol{x}_k = [s_{1,k}, s_{2,k}, s_{3,k}, s_{4,k}, \hat{v}^v_{1,k}, \hat{v}^v_{2,k}, \hat{v}^v_{3,k}]^{\mathrm{T}}$ is the state available at time k, which consists of the image moments measurements, and the estimated linear velocity. $\overline{\boldsymbol{x}}_{j,k}$ and $\overline{\boldsymbol{u}}_{j,k}$ denote the predicted values of the model states and inputs, respectively, at time $k+j$ based on the state information available at time k. $\boldsymbol{U}_k = [\overline{\boldsymbol{u}}_{0,k}, \cdots, \overline{\boldsymbol{u}}_{N_p-1,k}] \in \mathbb{R}^{4\times N_p}$ is the decision variable of the optimization problem. The first element $\overline{\boldsymbol{u}}_{0,k}$ is sent out and implemented on the quadrotor. N_p is the prediction horizon. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the weighting matrices for states and inputs, respectively. $\mathbb{U}$ is the constraint set for control inputs. $\boldsymbol{x}_{ref} = [0, 0, 1, 0, 0, 0, 0]^{\mathrm{T}}$.

3.4.2 Control Input Transformation

The output of the nonlinear MPC controller is $\boldsymbol{u}^* = [\dot{\psi}_d, \mathrm{f}_{1d}, \mathrm{f}_{2d}, \mathrm{f}_{3d}]^{\mathrm{T}}$, which needs to be transformed into desired Euler angles $\boldsymbol{\eta}_d = [\phi_d, \theta_d, \psi_d]^{\mathrm{T}}$, as required by the attitude

tracking controller in the inner-loop. The transformation is given as:

$$\begin{cases} \mathrm{f}_{1d} = -\dfrac{1}{m} T_d \sin(\theta_d)\cos(\phi_d) \\ \mathrm{f}_{2d} = \dfrac{1}{m} T_d \sin(\phi_d) \\ \mathrm{f}_{3d} = -\dfrac{1}{m} T_d \cos(\theta_d)\cos(\phi_d) + g \end{cases} \Rightarrow \begin{cases} T_d = m\sqrt{\mathrm{f}_{1d}^2 + \mathrm{f}_{2d}^2 + (\mathrm{f}_{3d} - g)^2} \\ \phi_d = \arcsin\left(\dfrac{\mathrm{f}_{2d}}{\sqrt{\mathrm{f}_{1d}^2 + \mathrm{f}_{2d}^2 + (f_{3d} - g)^2}}\right) \\ \theta_d = \arctan\left(\dfrac{\mathrm{f}_{1d}}{\mathrm{f}_{3d} - g}\right) \end{cases} \tag{3.23}$$

where the left equation set is derived from (3.1), and T_d is the desired total thrust force supplied by the propellers.

ψ_d is obtained by the integration of $\dot{\psi}_d$.

$$\psi_d(k+1) = \psi_m(k) + \dot{\psi}_d(k) T_s \tag{3.24}$$

where $\psi_d(k+1)$ is the reference yaw angle for the inner-loop during the $k+1$ sampling period, $\psi_m(k)$ is the yaw angle measurement from IMU at the sampling instant k, and $\dot{\psi}_d(k)$ is the $\dot{\psi}_d$ obtained at sampling instant k.

3.4.3 Input Constraints

As discussed before, saturating the roll/pitch angles is an effective technique to maintain the target object within the FOV of the camera. Therefore, constraints are imposed on the desired roll/pitch angle signals.

$$|\phi_d| \leq \phi_{\max}, \quad |\theta_d| \leq \theta_{\max} \tag{3.25}$$

where $\phi_{\max}$ and $\theta_{\max}$ are upper bounds of the reference roll and pitch angles. With reference to (3.23), (3.25) can be formulated as nonlinear constraints in the optimization problem.

Other input constraints can be

$$|\dot{\psi}_d| \leq \dot{\psi}_{\max}, \quad T_{\min} \leq |T_d| \leq T_{\max} \tag{3.26}$$

where $\dot{\psi}_{\max}$ is the upper bound of the yaw rate, $T_{\min}$ and $T_{\max}$ are the lower bound and upper bound of the thrust magnitude, respectively. (3.25) and (3.26) are explicit forms of (3.22) as they address constraints on the control inputs.

3.5 Roll/Pitch Constraints Adjustment Law

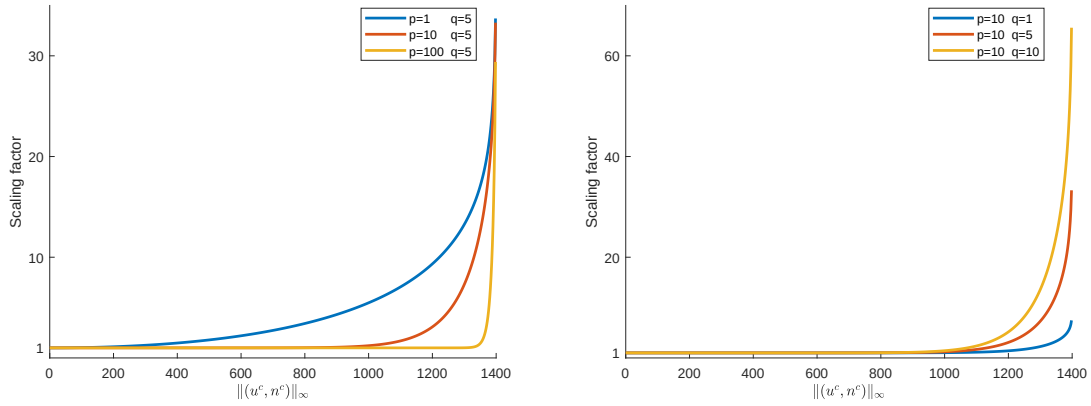
As discussed before, imposing bounds on the desired roll/pitch angles is effective for keeping the object of interest within the field of view (FOV). Even though a stringent limit can enforce the ground target to be more probably within the FOV, it leads to conservative maneuver and slow convergence. Therefore, instead of imposing constant constraints on roll/pitch angles, we design a roll/pitch limit adjustment law that can automatically adjust $\phi_{\max}$ and $\theta_{\max}$ based on the feedback of the point coordinates. The adjustment law defines the scaling factor (SF), by which the initial limit is shrunk, to obtain the updated limit. That is,

$$\begin{bmatrix} \phi_{\max,new} \\ \theta_{\max,new} \end{bmatrix} = \frac{1}{SF} \begin{bmatrix} \phi_{\max,initial} \\ \theta_{\max,initial} \end{bmatrix} \tag{3.27}$$

The adjustment law is given as:

$$SF = -q \left(\frac{norm}{boundary} \right)^p \log \left(\frac{boundary - norm}{boundary} \right) + 1 \tag{3.28}$$

where $norm = \max\{\|[u_k^c, n_k^c]\|_\infty\}_{k=1}^K$ with K being the number of the points contained in the ground target, and *boundary* is the pixel value at the boundaries of the FOV.



(a) Plots of scaling factors with varied p. (b) Plots of scaling factors with varied q.

Figure 3.2: Plots of scaling factor functions.

$p > 0$ and $q > 0$ are two tuning parameters that can be used to modify the shape of the scaling function.

Assuming that $boundary = 1400$, we plot the scaling function based on varieties of parameters p and q, as shown in Figure 3.2. It shows that the parameter p affects the width of the flat region, where the roll/pitch limit is maintained unchanged, while the parameter q affects the growth rate of the scaling factor in the fast growing region.

We calculate p and q based on the following specifications:

$$norm = 1000, \qquad SF = 1.2;$$
$$norm = 1200, \qquad SF = 3.$$

Then the values of p and q can be determined and the explicit form of (3.28) is given as:

$$SF = -5\left(\frac{norm}{boundary}\right)^{10} \log\left(\frac{boundary - norm}{boundary}\right) + 1 \tag{3.29}$$

The 2-D distribution of (3.29) in the FOV is illustrated in Figure 3.3.

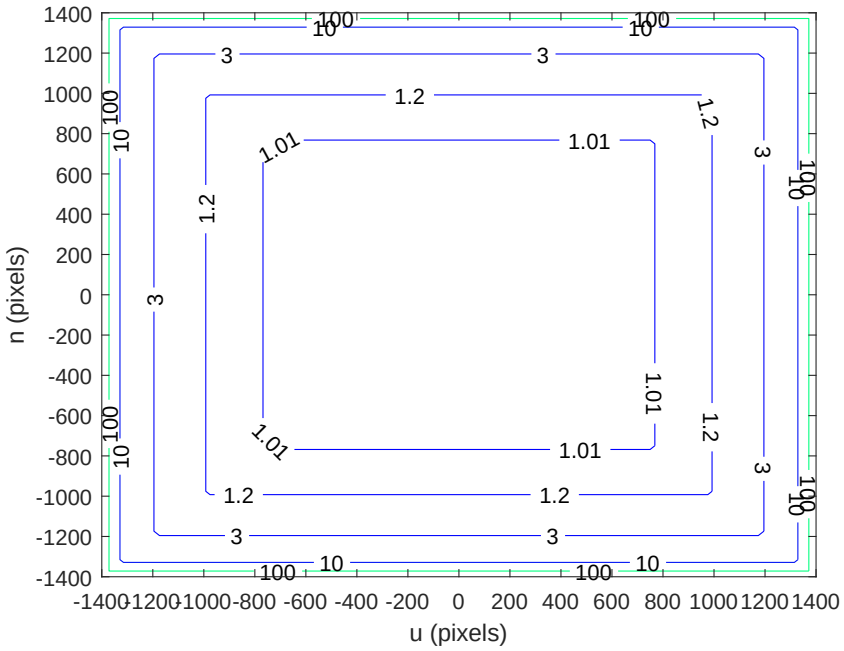


Figure 3.3: Scaling factor distribution in the FOV.

3.6 Explicit MPC Formulation

The online implementation of MPC algorithms requires considerable computation resources on board, which imposes challenges when confronting the real-time control. In this section, we decouple the yaw control from the position control of the quadrotor so that a linear model of the positional dynamics can be obtained. Accordingly, an explicit MPC controller is designed to regulate the position, and a proportional feedback controller is employed for the yaw regulation. To demonstrate the feasibility of separate control of the two subsystems, a switching mechanism is proposed to switch between the two controllers based on the current state of the quadrotor. More specifically, the switching mechanism will trigger the yaw regulator after the quadrotor has converged to the desired position under the explicit MPC visual servoing law. On the other hand, ideally, the yaw regulation can be accomplished without changing the position of the quadrotor. However, the quadrotor may drift away from the desired

position when undertaking a pure yaw rotation with the appearance of wind gusts. For this reason, the switching mechanism is designed to re-activate the explicit MPC controller, when the position error exceeds a pre-specified bound during the yaw regulation.

To simplify the system dynamics, we assume that the yaw remains unchanged during the position regulation, that is, $\dot{\psi} = 0$. Moreover, this assumption can be justified by enforcing the desired yaw rate to be zero, i.e., $\dot{\psi}_d = 0$. A linearized model of (3.17) can thus be obtained as follows:

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ s_{3,k+1} \\ v^v_{1,k+1} \\ v^v_{2,k+1} \\ v^v_{3,k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -T_s/Z^* & 0 & 0 \\ 0 & 1 & 0 & 0 & -T_s/Z^* & 0 \\ 0 & 0 & 1 & 0 & 0 & -T_s/Z^* \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{1,k} \\ s_{2,k} \\ s_{3,k} \\ v^v_{1,k} \\ v^v_{2,k} \\ v^v_{3,k} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T_s & 0 & 0 \\ 0 & T_s & 0 \\ 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} \mathrm{f}_{1,k} \\ \mathrm{f}_{2,k} \\ \mathrm{f}_{3,k} \end{bmatrix} \tag{3.30}$$

Since the above model is linear, we can conveniently solve the MPC optimization problem offline and implement the explicit solution during online implementation.

After the quadrotor is regulated to the desired position, the yaw regulation will be triggered and applied. The yaw of the quadrotor is regulated by a simple proportional feedback controller:

$$\dot{\psi}_d = K_\psi s_4 \tag{3.31}$$

where K_ψ is a positive gain. Substituting (3.31) into (2.36) gives the closed-loop yaw dynamics:

$$\dot{s}_4 = -K_\psi s_4 \tag{3.32}$$

which is globally exponentially stable.

The overall outer-loop system dynamics described by (3.17) has been divided into two subsystems: a translational subsystem described by (3.30) and a yaw subsystem described by (2.36). Theoretically, each subsystem can be stabilized without affecting the states of the other. Switching rules between the two controllers are defined as below:

- Switch from the position regulator to the yaw regulator, when the position regulation has been achieved and the yaw regulation had not been achieved yet.
- Switch from the yaw regulator to the position regulator, if the position error exceeds a pre-specified upper bound during yaw regulation or the yaw regulation has been achieved.

The first switching rule indicates that the position regulation is executed prior to the yaw regulation, since this sequence benefits keeping the target object within the FOV. Moreover, the second switching rule can effectively address a practical issue: The target object may leave the FOV during yaw regulation due to the presence of external disturbance, e.g., wind gusts. By constantly monitoring the position error, the position regulator can be triggered and applied when the yaw regulation is interrupted by wind gusts.

Explicit MPC approximates the solution to the nonlinear programming problem (3.18)–(3.22) by a series of piecewise affine (PWA) functions through parametric programming. This process is done offline and thus dramatically saves the computation time during online implementation. The pre-specified state space is firstly divided into multiple control regions, and then for each control region, the corresponding PWA is computed (see Figure 3.4). The obtained explicit MPC controller stores the coefficients of parametric representations of the control regions, as well as the coefficients of the corresponding PWAs. During online implementation, given the current

state information, the control region containing the state will be found and then the corresponding PWA coefficients will be used to evaluate the control action.

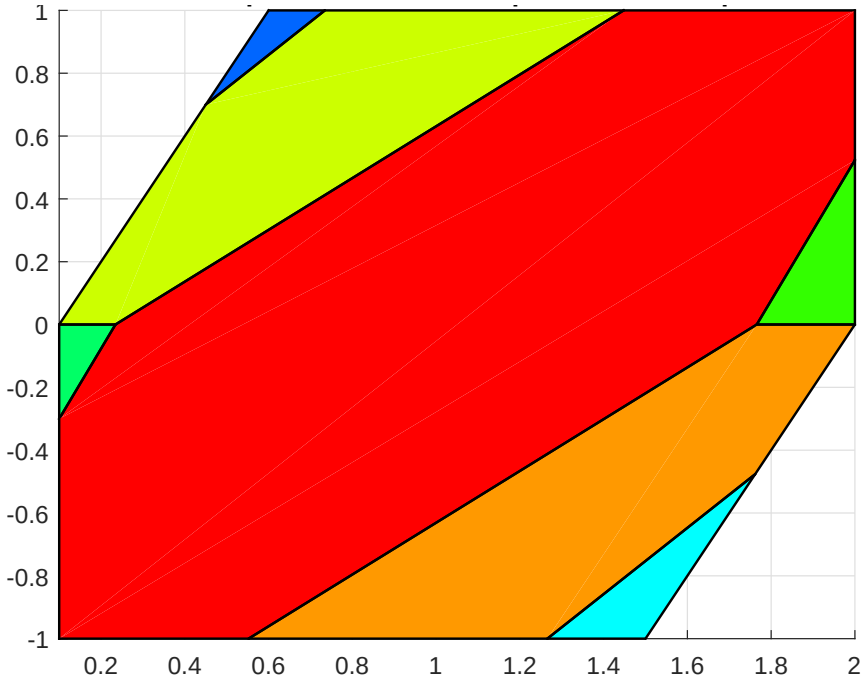


Figure 3.4: Control regions partitioned in the state space.

3.7 Inner-Loop Attitude Tracking Control

In the preceding sections, two MPC-based visual servoing controllers are proposed to regulate the outer-loop translational dynamics of the quadrotor. In this section, a proportional-derivative (PD) controller is designed for the attitude tracking in the inner-loop. We first show that the inner-loop dynamics can be linearized to a double integrator by feedback linearization, and then a PD controller is applied to the linearized system.

3.7.1 Feedback Linearization of Inner-Loop Dynamics

Recall that the rotational dynamics of the quadrotor is governed by (2.6) and (2.7):

$$\dot{\boldsymbol{R}}_C^N = \boldsymbol{R}_C^N \left[\boldsymbol{\Omega}^c\right]_\times$$
$$\dot{\boldsymbol{\Omega}}^c = -\boldsymbol{I}^{-1} \left[\boldsymbol{\Omega}^c\right]_\times \boldsymbol{I}\boldsymbol{\Omega}^c + \boldsymbol{I}^{-1}\boldsymbol{\tau}^c$$

In addition, (2.6) has the following equivalent expression as proved in (2.10):

$$\boldsymbol{\Omega}^c = \boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} \tag{3.33}$$

Differentiating (3.33) on both sides gives:

$$\dot{\boldsymbol{\Omega}}^c = \dot{\boldsymbol{M}}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} + \boldsymbol{M}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} \tag{3.34}$$

Substituting (3.33) and (3.34) into (2.7) gives:

$$\dot{\boldsymbol{M}}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} + \boldsymbol{M}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} = -\boldsymbol{I}^{-1}[\boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}]_\times \boldsymbol{I}\boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} + \boldsymbol{I}^{-1}\boldsymbol{\tau}^c \tag{3.35}$$

Multiplying both sides by $\boldsymbol{I}$ and rearranging the equation gives:

$$\boldsymbol{I}\boldsymbol{M}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + (\boldsymbol{I}\dot{\boldsymbol{M}}(\boldsymbol{\eta}) + [\boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}]_\times \boldsymbol{I}\boldsymbol{M}(\boldsymbol{\eta}))\dot{\boldsymbol{\eta}} = \boldsymbol{\tau}^c \tag{3.36}$$

The above equation can be written in a compact form:

$$\boldsymbol{D}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) = \boldsymbol{\tau}^c \tag{3.37}$$

where $\boldsymbol{D}(\boldsymbol{\eta}) = \boldsymbol{I}\boldsymbol{M}(\boldsymbol{\eta})$, and $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) = \dot{\boldsymbol{D}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} + [\boldsymbol{M}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}]_\times \boldsymbol{D}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}$.

3.7.2 Dynamics Inversion-Based PD Controller

A dynamics inversion controller can be designed as:

$$\boldsymbol{\tau}^c = \hat{\boldsymbol{D}}(\boldsymbol{\eta})\tilde{\boldsymbol{\tau}} + \hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) \tag{3.38}$$

where $\hat{\boldsymbol{D}}(\boldsymbol{\eta})$ and $\hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ are the approximation of $\boldsymbol{D}(\boldsymbol{\eta})$ and $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$, respectively. $\tilde{\boldsymbol{\tau}}$ is the virtual control input that is to be generated by the PD controller. $\boldsymbol{\eta}$ and $\dot{\boldsymbol{\eta}}$ are assumed to be measurable from the IMU.

Suppose that the system parameters are perfectly identified, i.e., $\hat{\boldsymbol{D}}(\boldsymbol{\eta}) = \boldsymbol{D}(\boldsymbol{\eta})$, and $\hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$=$\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$, then we have the resulting closed-loop dynamics:

$$\ddot{\boldsymbol{\eta}} = \tilde{\boldsymbol{\tau}} \tag{3.39}$$

Notice that it is a double integrator, and accordingly a PD controller can be designed as:

$$\tilde{\boldsymbol{\tau}} = \ddot{\boldsymbol{\eta}}_d + \boldsymbol{K}_p(\boldsymbol{\eta}_d - \boldsymbol{\eta}) + \boldsymbol{K}_d(\dot{\boldsymbol{\eta}}_d - \dot{\boldsymbol{\eta}}) \tag{3.40}$$

where $\boldsymbol{\eta}_d = [\phi_d, \theta_d, \psi_d]^{\mathrm{T}}$ is generated by the outer-loop controller, and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are positive gain matrices. $\dot{\boldsymbol{\eta}}_d$ and $\ddot{\boldsymbol{\eta}}_d$ can be either computed by numerical differentiation, or assumed to be $\mathbf{0}$, which leads to a simplified version of (3.40):

$$\tilde{\boldsymbol{\tau}} = \boldsymbol{K}_p(\boldsymbol{\eta}_d - \boldsymbol{\eta}) - \boldsymbol{K}_d\dot{\boldsymbol{\eta}} \tag{3.41}$$

The overall control system is illustrated by a block diagram, as shown in Figure 3.5.

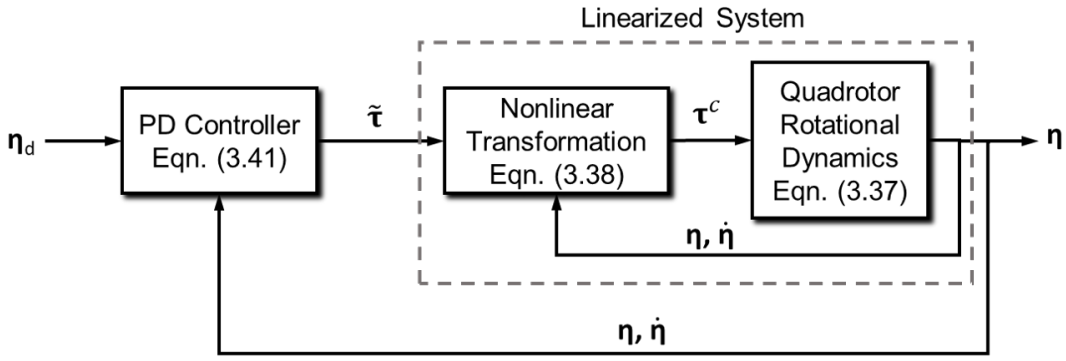


Figure 3.5: Inner-loop attitude tracking using PD control.

If we define the attitude tracking error as:

$$\boldsymbol{e} = \boldsymbol{\eta} - \boldsymbol{\eta}_d \tag{3.42}$$

By substituting (3.40) into (3.39), the error dynamics can be derived as:

$$\ddot{\boldsymbol{e}} = -\boldsymbol{K}_p \boldsymbol{e} - \boldsymbol{K}_d \dot{\boldsymbol{e}} \tag{3.43}$$

We can write the above equation in the state-space form:

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{e} \\ \dot{\boldsymbol{e}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix} \begin{bmatrix} \boldsymbol{e} \\ \dot{\boldsymbol{e}} \end{bmatrix} \tag{3.44}$$

It is straightforward to verify that the state transition matrix is Hurwitz, i.e., all its eigenvalues are in the left half plane, and therefore the system is asymptotically stable. In other words, $\boldsymbol{\eta}$ will eventually converge to $\boldsymbol{\eta}_d$.

If the system is perfectly identified, the inverse dynamics method enables the nonlinearity of the original system to be effectively compensated, and the controller can be designed based on the linearized model. However, the drawback of inverse dynamics method is its sensitivity to modelling uncertainties. For example, the uncertainty

of the moment of inertia $\boldsymbol{I}$ will lead to imperfect compensation of $\boldsymbol{D}(\boldsymbol{\eta})$ and $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$, that is,

$$\boldsymbol{D}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) = \hat{\boldsymbol{D}}(\boldsymbol{\eta})\tilde{\boldsymbol{\tau}} + \hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) \tag{3.45}$$

The resulting closed-loop system has an additional error term on the right hand side, compared to (3.39):

$$\begin{aligned} \ddot{\boldsymbol{\eta}} &= \boldsymbol{D}^{-1}(\boldsymbol{\eta})\hat{\boldsymbol{D}}(\boldsymbol{\eta})\tilde{\boldsymbol{\tau}} + \boldsymbol{D}^{-1}(\boldsymbol{\eta})(\hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) - \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})) \\ &= \tilde{\boldsymbol{\tau}} + (\boldsymbol{D}^{-1}(\boldsymbol{\eta})\hat{\boldsymbol{D}}(\boldsymbol{\eta}) - \boldsymbol{I}_m)\tilde{\boldsymbol{\tau}} + \boldsymbol{D}^{-1}(\boldsymbol{\eta})(\hat{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) - \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})) \\ &= \tilde{\boldsymbol{\tau}} + \boldsymbol{E}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}, \tilde{\boldsymbol{\tau}}) \end{aligned} \tag{3.46}$$

where $\boldsymbol{I}_m$ is the identity matrix.

The error term $\boldsymbol{E}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}, \ddot{\boldsymbol{\eta}}_d)$ can possibly accumulate with time and destabilize the overall system [47]. However, it is believed that the adverse effects of $\boldsymbol{E}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}, \ddot{\boldsymbol{\eta}}_d)$ can be well treated by some control techniques such as sliding mode control [48] and neural network-based control approach [11]. One of future work would be to explicitly consider $\boldsymbol{E}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}, \ddot{\boldsymbol{\eta}}_d)$ in our attitude tracking controller design.

3.8 Numerical Simulation

Simulations are performed in MATLAB/Simulink to verify the effectiveness of the controller designs proposed in Section 3.1–3.7.

3.8.1 Simulation Set-up

The control objective is to regulate the quadrotor to 1 m above the barycenter of the ground target, with its yaw direction aligned with the principal axis of the ground target. That is, $\boldsymbol{s}_{ref} = [0, 0, 1, 0]^{\mathrm{T}}$, and $Z^* = 1$ m. The mass of the quadrotor is $m =$

1.2 kg, and the moment of inertia $\boldsymbol{I}$ is estimated to be diag($[0.013, 0.013, 0.023]$), based on the mass and the overall dimension of the quadrotor. The gravitational constant $g = 9.8$ m/s^2. The ground targets are four coplanar points on the level ground, with their coordinates in $\mathbf{N}$ being $[0.25, 0.2, 0]^{\mathrm{T}}$, $[-0.25, 0.2, 0]^{\mathrm{T}}$, $[-0.25, -0.2, 0]^{\mathrm{T}}$, and $[0.25, -0.2, 0]^{\mathrm{T}}$. The initial position of the quadrotor in $\mathbf{N}$ is $[-1, -0.6, -3]^{\mathrm{T}}$ m, and the initial Euler angles are $[0, 0, -0.174]^{\mathrm{T}}$ rad. The corresponding image features obtained at the initial pose are $\boldsymbol{s}_{initial} = [0.881, 0.764, 3, 0.174]^{\mathrm{T}}$. The quadrotor is assumed to start from static and thus the initial velocity is set to $[0, 0, 0]^{\mathrm{T}}$. By definition, a^* is calculated to be 3.572×10^{-7} at the desired pose.

For parameters used in the camera model: The focal length λ is set to be 2.8 mm; pixels are assumed to be squares with side length of 1.4×10^{-6} m, and the field of view is a square of 2800×2800 in pixels.

3.8.2 Simulation Study: High-Gain Observer-Based Nonlinear Model Predictive Controller

In the first simulation, we use the high-gain observer (3.8)–(3.11) and the nonlinear MPC controller (3.18)–(3.26). In the high-gain observer: $\epsilon = 10^{-2}$, $\alpha_1^1 = \alpha_1^2 = \alpha_1^3 = 10$, $\alpha_2^1 = \alpha_2^2 = \alpha_2^3 = 9$. In the NMPC controller: $N_p = 10$, $T_s = 0.03$ s. N_c is set to 1 to reduce the computational load. $\boldsymbol{Q}$ =diag($[10, 10, 10, 1, 1, 1, 1]$). $\boldsymbol{R} = \mathbf{0}_{4\times4}$. The sampling period of the inner-loop is set to $\frac{1}{10}$ of T_s. Input constraints are set as: $\phi_{max} = \theta_{max} = 0.1$ rad, $\dot{\psi}_{max} = 0.4$ rad/s, $T_{min} = 0.5mg$ and $T_{max} = 1.5mg$.

Figure 3.6 shows the evolution of state estimates when $\hat{\boldsymbol{\xi}}$ is initialized by $[0, 0, 0, 0, 0, 0]^{\mathrm{T}}$. It is observed that the estimated values can converge to the actual states promptly, but a considerable observation error may be witnessed during the transient phase. We can alleviate the initial observation error by initializing $\hat{\boldsymbol{\xi}}$ with image moment measurements. Hence, the initial value of $\hat{\boldsymbol{\xi}}$ is set to $[0.881, 0, 0.764, 0, 3, 0]^{\mathrm{T}}$, and the

obtained state estimate trajectories are shown in Figure 3.7.

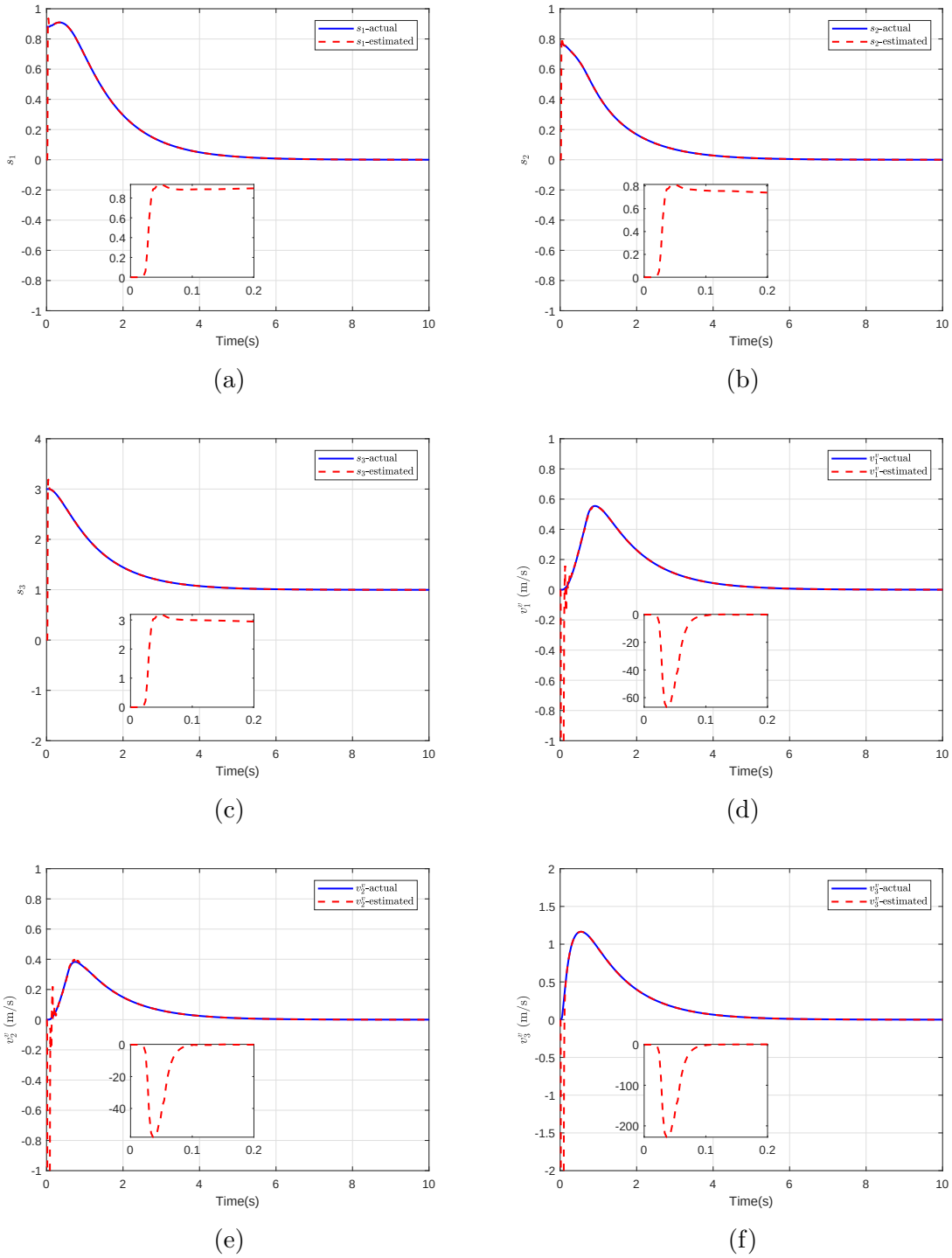


Figure 3.6: Estimated and actual states obtained when $\hat{\boldsymbol{\xi}}$ is initialized by $[0, 0, 0, 0, 0, 0]^{\mathrm{T}}$.

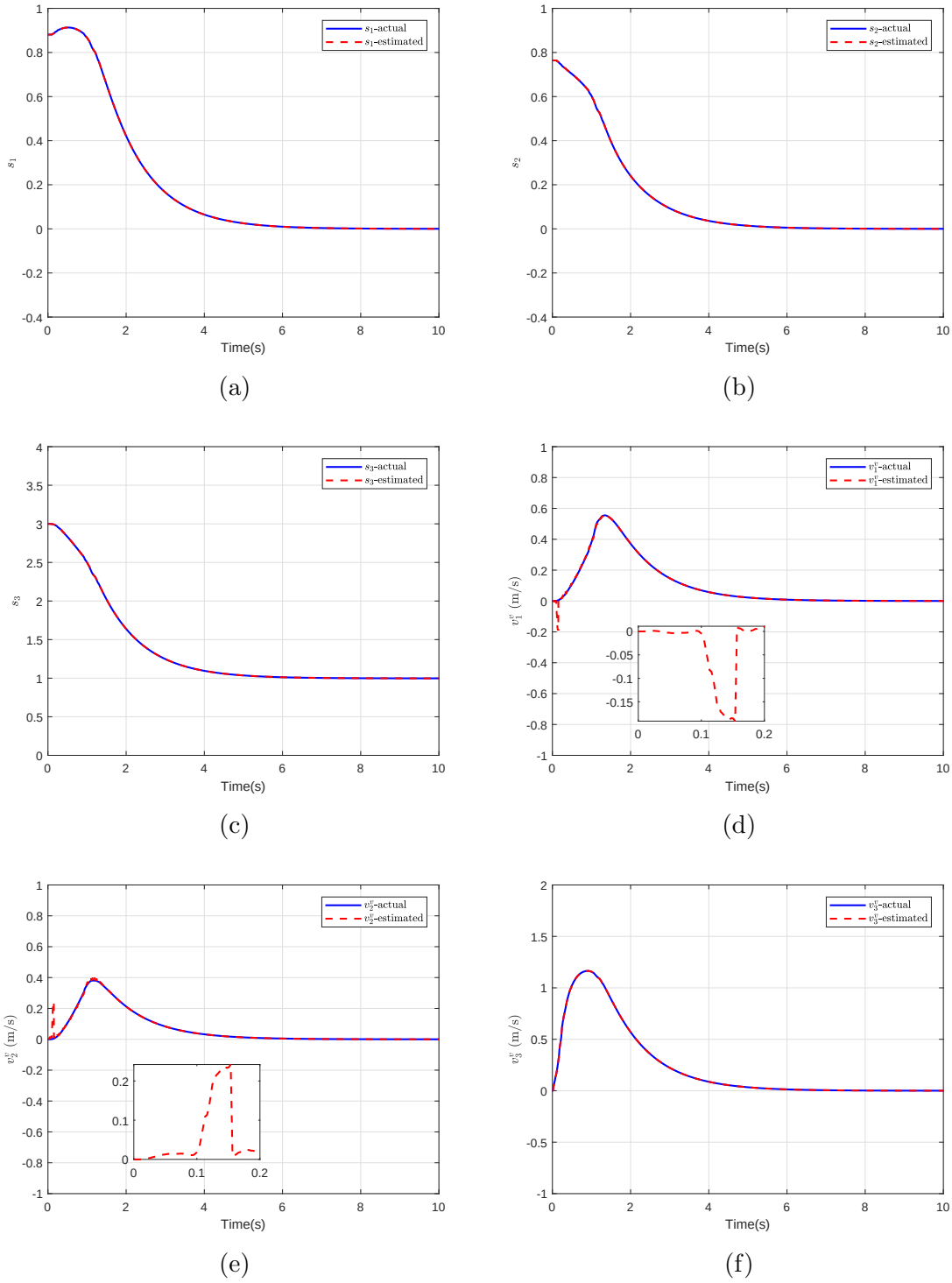
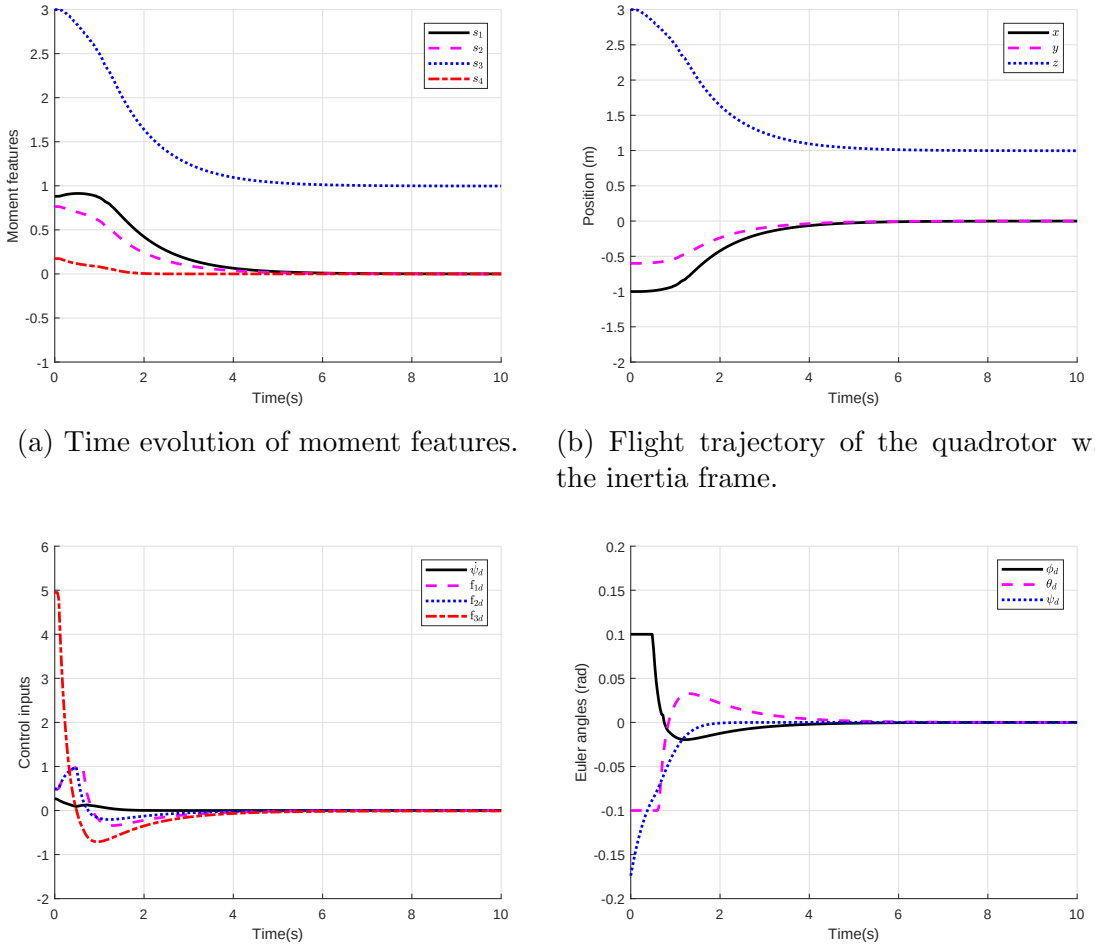


Figure 3.7: Estimated and actual states obtained when $\hat{\boldsymbol{\xi}}$ is initialized by $[0.881, 0, 0.764, 0, 3, 0]^{\mathrm{T}}$.

Figure 3.8 shows the closed-loop trajectories of the system states. Figure 3.8(a) shows that the moment features successfully converge to the desired value $[0, 0, 1, 0]^{\mathrm{T}}$. Fig. 3.8(b) shows the flight trajectory of the quadrotor in frame **N**. It can be seen that the quadrotor successfully converges to the desired pose, which is 1m above the barycenter of the ground target. Figures 3.8(c) and 3.8(d) show the control input trajectories and the desired Euler angle trajectories, respectively. The desired roll and pitch angles abide by the pre-defined bounds, 0.1 rad.



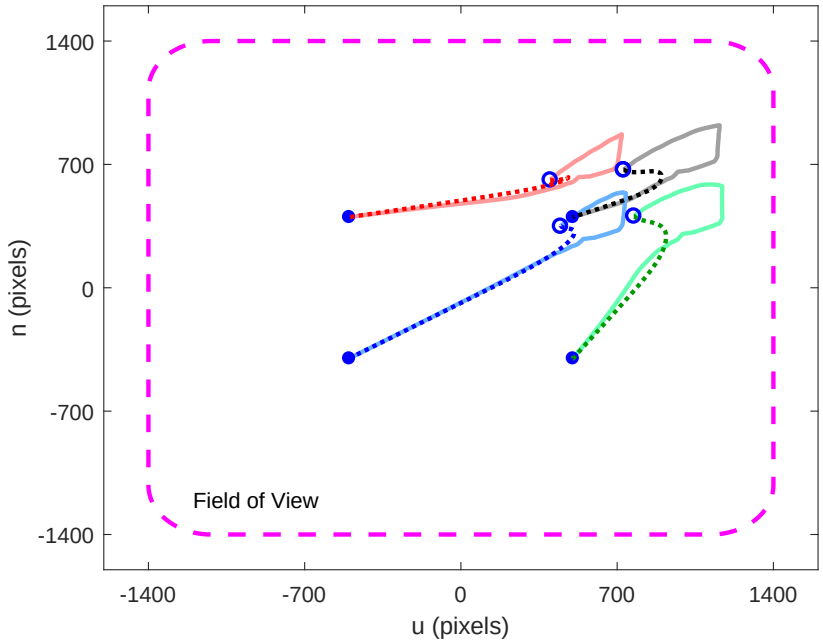
(a) Time evolution of moment features.

(b) Flight trajectory of the quadrotor w.r.t the inertia frame.

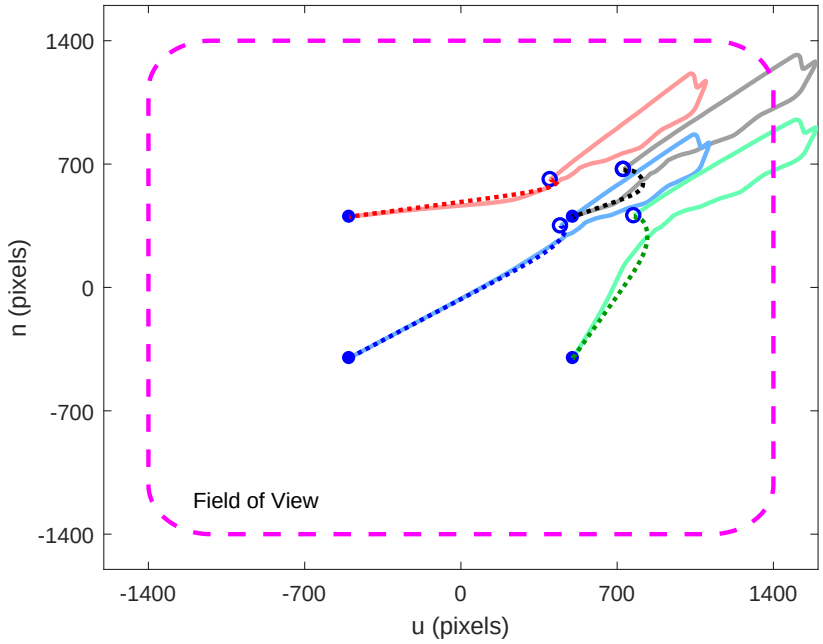
(c) Closed-loop control inputs trajectories.

(d) Attitude setpoints sent to the inner-loop.

Figure 3.8: Closed-loop states and inputs trajectories obtained by using the nonlinear MPC controller



(a)



(b)

Figure 3.9: (a) Point coordinate trajectories in the real image plane (solid) and in the virtual image plane (dotted) when $\phi_{\max} = \theta_{\max} = 0.1$ rad. (b) Point coordinate trajectories in the real image plane (solid) and in the virtual image plane (dotted) when $\phi_{\max} = \theta_{\max} = 0.3$ rad.

In Figure 3.9, the point coordinate trajectories in the real image plane are plotted by solid lines, while point coordinate trajectories in the virtual image plane are plotted by dotted lines. Figure 3.9(a) and Figure 3.9(b) show cases when the roll/pitch angles are bounded by 0.1 rad, 0.3 rad, respectively. It is observed that the dotted lines in the two figures are quite similar due to that the virtual image plane is not affected by the roll/pitch motion of the quadrotor. On the other hand, the solid lines in Figure 3.9(a) are better confined in the FOV compared to those in Figure 3.9(b), which demonstrates the improved visibility performance through properly tightening the roll/pitch constraints.

3.8.3 Simulation Study: Incorporating the Roll/Pitch Constraints Adjustment Law

In the first simulation, we apply constant constraints on the roll/pitch angles. It shows that by properly tightening this constraints, the target scene can be maintained in the FOV. Nevertheless, as discussed in Section 3.5, we can design a roll/pitch constraints adjustment law to automatically adjust the constraints according to the location of object scene with respect to the FOV. Figure 3.10 shows the point coordinate trajectories obtained when the roll/pitch adjustment law (3.29) is applied. It can be seen that the trajectories in Figure 3.10 are quite similar to those in Figure 3.9(a). Yet, the scaling factor is continuously updated and the bounds on the pitch/roll are correspondingly adjusted, as shown in Figure 3.11.

Figure 3.11(a) shows the evolution of the scaling factor defined in (3.29). There are several peaks in the curve as the scaling factor function is very responsive to the location of the point coordinates in the FOV. To avoid the chattering of the scaling factor, we can calculate the scaling factor by taking the average of the current and previous values. As a result, the curve of the scaling factor is smooth out, as shown

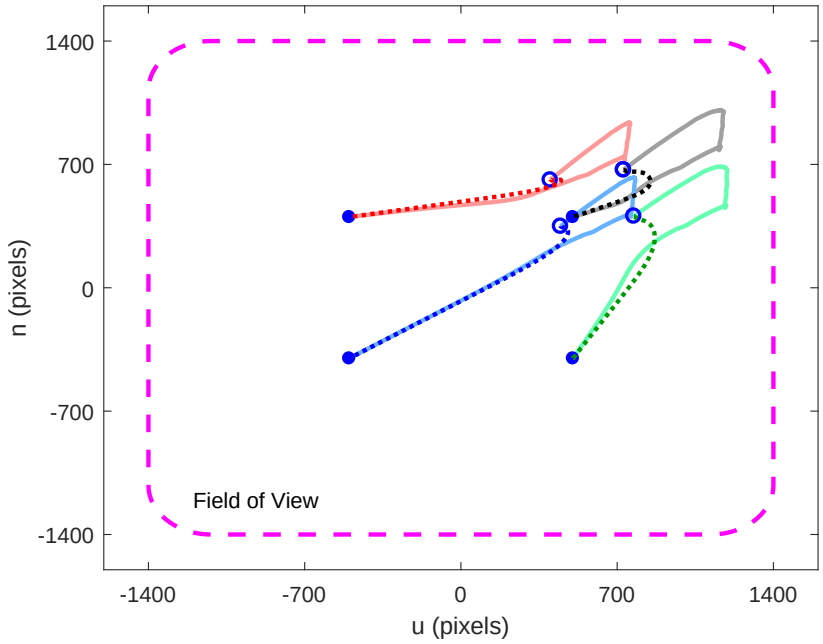
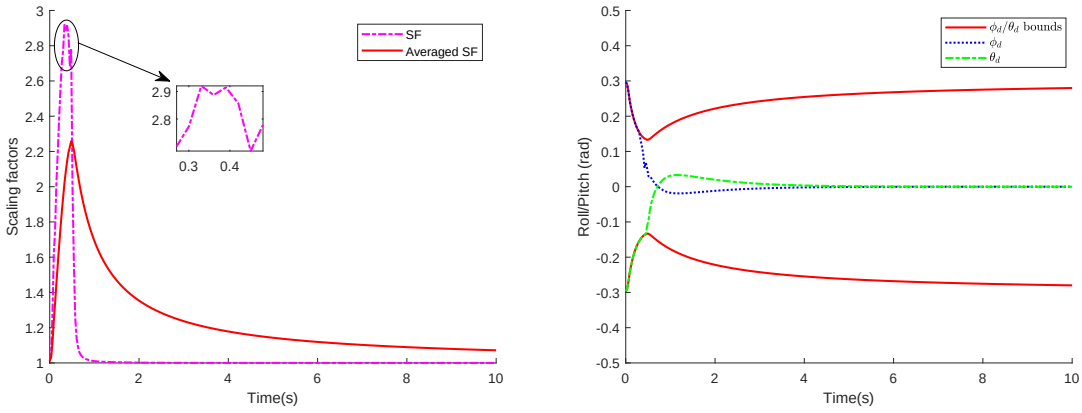


Figure 3.10: Point coordinate trajectories with Euler constraints adjustment law.

in Figure 3.11(a). The corresponding roll/pitch angle bounds are shown in Figure 3.11(b). The roll/pitch bounds rapidly decrease at the acceleration phase to mitigate aggressive rotation, and then recover to the original bounds after the quadrotor has gained velocity. The resulting desired roll/pitch angles strictly satisfy the bounds throughout the flight simulation.



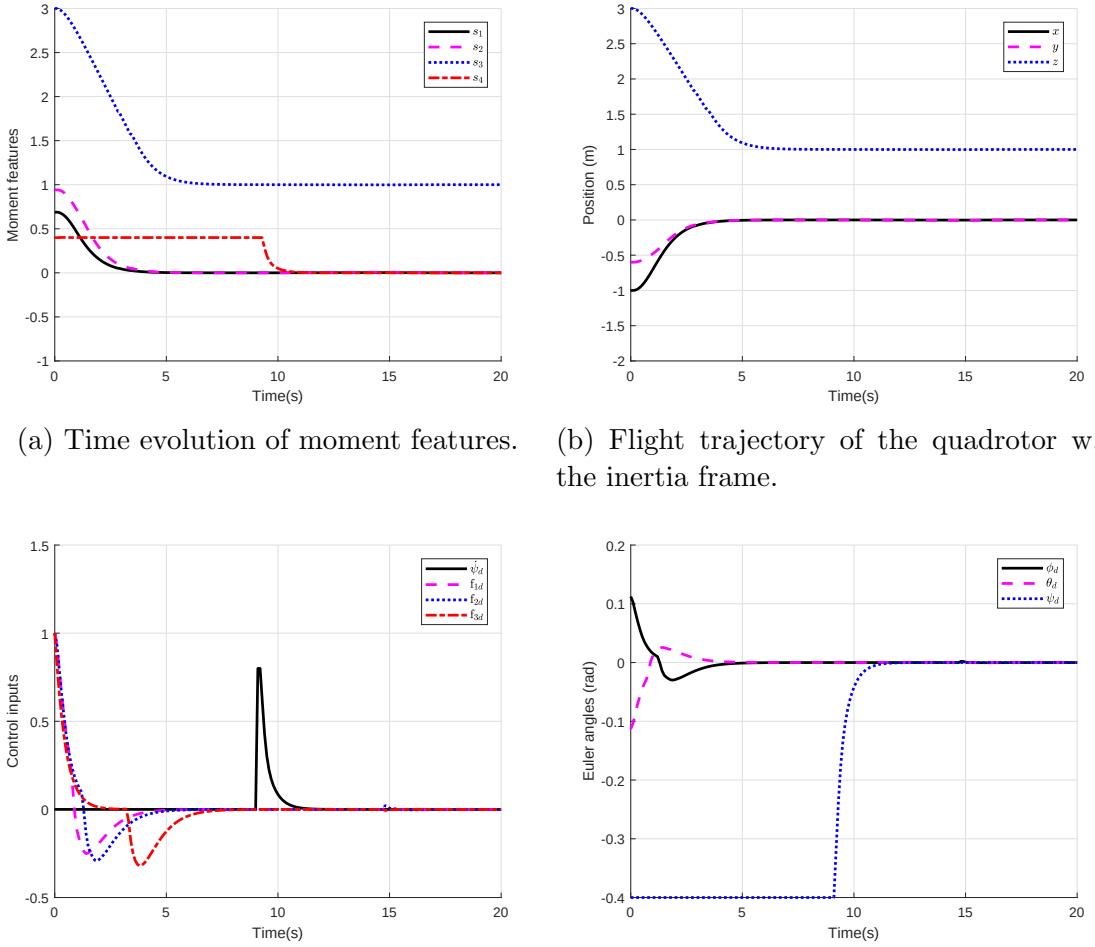
(a) (b)

Figure 3.11: (a) Scaling factor is averaged to avoid jumps. (b) Roll/pitch angles under the adjustment law.

3.8.4 Simulation Study: Explicit MPC Controller

The Multi-Parametric Toolbox (MPT) is used to generate an explicit MPC controller based on the linear model in (3.30). Most of the parameters are the same as those in Section 3.8.1, except a few to be discussed below. In the discrete-time model used in MPT, the prediction time is set to 5 s with a time step of 0.5 s. Since a linear model is used in our prediction, a larger time step would not induce as much prediction error as in the nonlinear model. On the other hand, this extended prediction horizon contributes to more robust performance in the presence of disturbance, e.g., wind gusts, as observed in the experimental trials. $\boldsymbol{Q}$ =diag([100, 100, 100, 10, 10, 10]), and $\boldsymbol{R}$=diag([1, 1, 1]). s_1 and s_2 are constrained in the set $\{s_i | -2 \leq s_i \leq 2\}$ for $1 \leq i \leq 2$, while s_3 is constrained in $\{s_3 | -5 \leq s_3 \leq 5\}$. The velocity v_1^v–v_3^v are constrained in the set $\{v_i^v | -0.5 \leq v_i^v \leq 0.5\}$ for $1 \leq i \leq 3$, and the control inputs f_1–f_3 are constrained in $\{\mathrm{f}_i | -2 \leq \mathrm{f}_i \leq 2\}$ for $1 \leq i \leq 3$. The explicit MPC controller is comprised of 3825 control regions.

For the proportional yaw controller, the control gain K_ψ is set to 2.



(a) Time evolution of moment features.

(b) Flight trajectory of the quadrotor w.r.t the inertia frame.

(c) Closed-loop control inputs trajectories.

(d) Attitude setpoints sent to the inner-loop.

Figure 3.12: Closed-loop states and inputs trajectories obtained by using the explicit MPC controller and the proportional yaw controller

Figure 3.12(a) and Figure 3.12(b) show the convergence of the states in the image space and the 3-D space, respectively. Figure 3.12(c) shows the control inputs generated by the explicit MPC position controller and the proportional yaw controller. It can be seen that the yaw controller is triggered at 9 s, and then the yaw regulation is accomplished at 14.8 s, when the position regulator is switched on again. Since a pure yaw motion does not change the position of the quadrotor, small control efforts are observed when the position regulator is switched on again. Figure 3.12(d) shows the corresponding desired Euler angles.

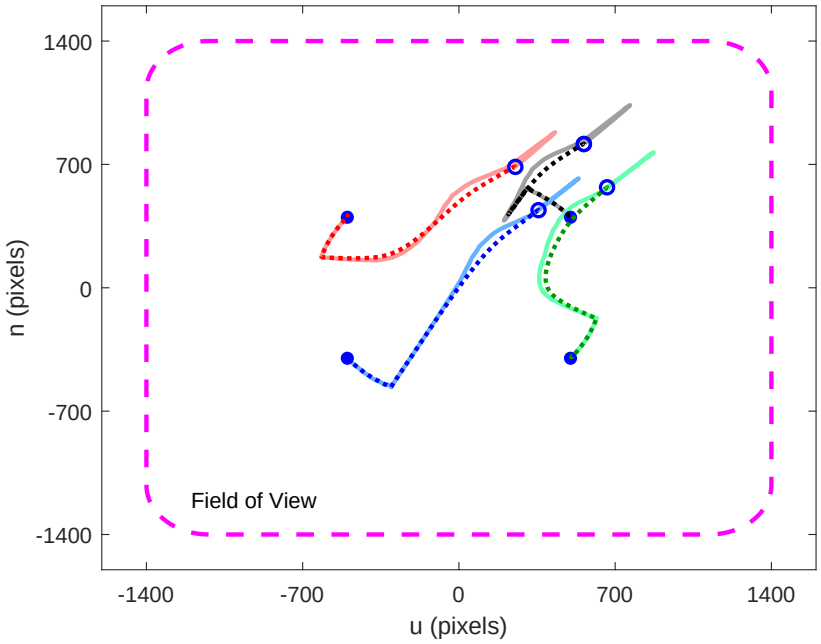


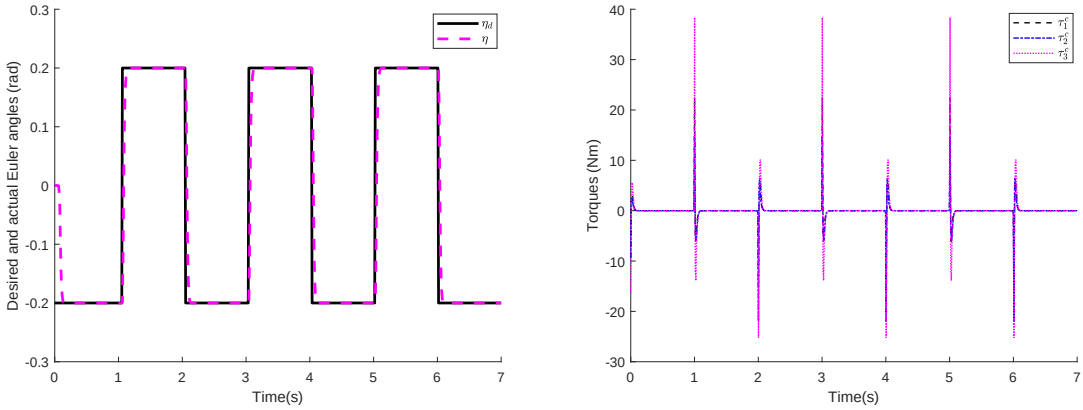
Figure 3.13: Point coordinate trajectories.

The motion of the target object in the virtual image plane and the real image plane is shown in Figure 3.13. It is observed that the trajectories in virtual image plane deviate from those in the real image plane at the starting phase due to large roll/pitch motion as illustrated in 3.12(d). On the other hand, it is observed that a pure yaw rotation is experienced at the ending phase of the flight when the proportional yaw controller is switched on.

3.8.5 Simulation Study: Inner-Loop PD Controller

The PD controller governed by (3.41) can be used for the attitude tracking in the inner-loop. The control gains are selected as: $\boldsymbol{K}_p$ =diag([3600, 3600, 3600]); $\boldsymbol{K}_d$ =diag([100, 100, 100]). Figure 3.14(a) shows that the PD controller enables the desired Euler angles to be well tracked in a prompt manner. The desired roll, pitch, yaw signals are set to the same square waves, and accordingly the curves representing the actual Euler

angles are coinciding with each other. The time evolution of the generated torques is shown in Figure 3.14(b).



(a) Attitude tracking trajectories.

(b) Torques generated by the inner-loop PD controller.

Figure 3.14: Attitude tracking performance obtained by the PD controller.

3.9 Conclusion

In this chapter, we first present a high-gain observer-based NMPC scheme for the visual servoing of the quadrotor. The high-gain observer is employed to estimate the linear velocity of the quadrotor, and the NMPC generates the desired Euler angles to provide guidance to the lower level control. To address the visibility constraints, a roll/pitch constraints adjustment law is designed to properly bound the Euler angles so that the target object is maintained within the FOV of the camera. Then, to improve the real-time performance of the controller, an explicit MPC controller is designed such that the solution of the online optimization problem can be obtained offline through parametric programming. As a result, the online evaluation time is significantly reduced and a faster sampling rate can be achieved. Moreover, a PD controller is designed for the inner-loop rotational dynamics to track the desired

attitude signals generated by the visual servoing controller. Lastly, simulation results are presented in the end to demonstrate the effectiveness of the proposed control schemes.

Chapter 4

Experimental Implementation

4.1 Overview

Experiments are conducted to verify the effectiveness of the controller designs proposed in Chapter 3. The experimental set-up is firstly introduced in Section 4.2. Then, in Section 4.3, several preparation works for image feature extraction are elaborated: Section 4.3.1 illustrates the camera calibration that is performed to determine the intrinsic parameters of the camera; Section 4.3.2 presents the image processing algorithms that are developed to extract the point coordinates; and Section 4.3.3 propose a modification in the image moment calculation to compensate model mismatch. Lastly, experimental results are provided to evaluate the performance of the nonlinear MPC controller and the explicit MPC controller.

4.2 Experimental Set-up

The hardwares and softwares used in the experiments are introduced in this section.

Figure 4.1: Bebop 2 drone [49].

4.2.1 Hardware

The quadrotor used in the experiment is a Bebop 2 drone developed by Parrot, as shown in Figure 4.1. The Bebop 2 drone weighs 500 g and offers 25 minutes of flight time. The on-board components are summarized in Table 4.1. The drone is equipped with a wide-angle fisheye lens with digital image stabilization feature. The digital image stabilizer can virtually pitch the camera upwards and downwards, enabling the observation of the objects in front of and below the drone body. Digital stabilization system functions as an economical and compact alternative to a physical gimbal system.

The video stream is transmitted to a ground station at the rate of 30 frames per second (fps). The ground station is a ThinkPad laptop with an Intel i7-6600U processor (2.60 GHz) and 16 GB memory. The ground station is used for processing image sequences, extracting image features, and generating control commands.

Specifications	
Processor	Parrot P7 dual-core CPU Cortex 9 with quad-core GPU 8GB flash
Sensors	MPU6050 for accelerometers and gyroscope (I2C) AKM 8963 compass MS5607 barometer Furuno GN-87F GPS Sonar Optical-flow HD camera
Interfaces	1x UART serial ports USB Built-in Wi-Fi
Dimensions	33×38×3.6cm 500g
Operating System	Linux (Busybox)

Table 4.1: Bebop 2 Hardware Specifications.

4.2.2 Software

Pyparrot [50] is a Python application programming interface (API) for Parrot Bebop drones. API is a library that provides all the building blocks for the development of a software. For example, Pyparrot includes the communication protocols that facilitate the communication between the drone and the ground station, and some basic piloting commands such as take-off and landing.

Figure 4.2 shows the data flow between the hardwares and the softwares. The red dashed box contains the processes taking place within the ground station, whereas the blue dashed box contains the processes taking place on board the quadrotor. In addition, The rectangles represent hardwares; the ovals represent software modules, and the round rectangles represent control modes.

As shown in the blue dashed box, the autopilot collects sensor measurements and sends them to the ground station. On the other hand, the autopilot receives the

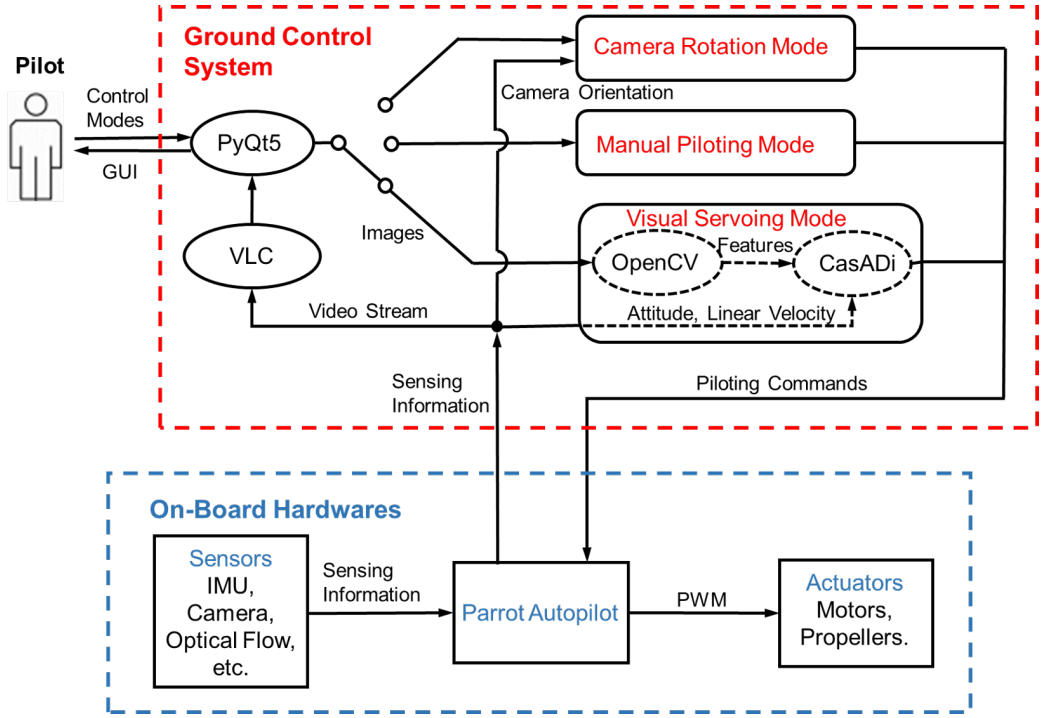


Figure 4.2: Data flow diagram.

piloting commands from the ground station and assigns PWM signals to actuate the motors.

As shown in the red dashed box, VLC, PyQt5, OpenCV, CasADi are installed in the ground station. VLC is a cross-platform media player that is used to handle the video stream, and its functionalities can be accessed by Python using libVLC module. PyQt5 is a set of Python bindings for Qt, a C++ library containing a variety of APIs for building functionalities commonly seen on modern desktops and mobile systems. Using PyQt5, a control panel interface (see Figure 4.3) is designed to allow the human pilot to modify the operating modes of the quadrotor. Since the camera is facing forward by default, the camera will be rotated to face downward by using a proportional controller when the "camera rotation" mode is enabled. This mode is usually activated before the take-off to extend the operation time in the air. The "manual piloting" mode allows the human operator to manually pilot the

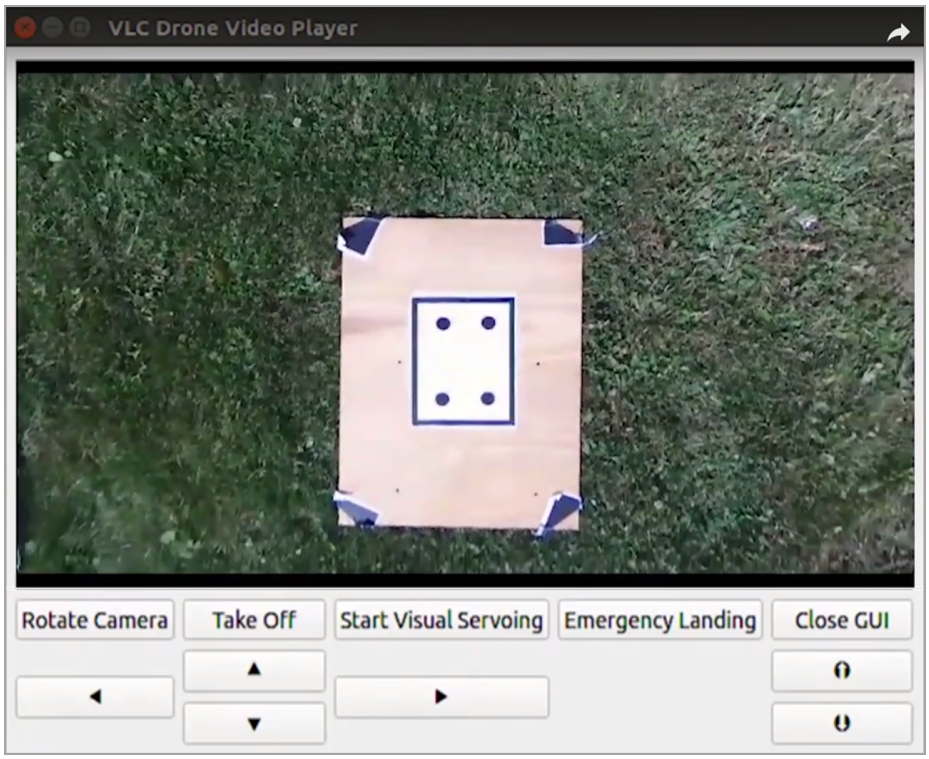


Figure 4.3: Ground station control panel.

quadrotor using several piloting commands. This mode helps to obtain a suitable initial position for triggering the visual servoing mode, which requires the target object to be observed in the FOV of the camera. After visual servoing is triggered, image processing algorithms developed in OpenCV extract the point coordinates from the images and then the image moments are calculated. The image moments together with the IMU measurements, linear velocity information are sent to the CasADi module that is responsible for solving the optimization problem and generating the control commands.

CasADi [51] is an open-source tool for nonlinear optimization. For a continuous-time system, the dynamics model can be firstly discretized by Runge-Kutta method, and then the nonlinear programming problem (3.18)–(3.26) can be formulated as a boundary value problem by direct multiple shooting method and solved by an interior point optimizer (IPOPT).

4.3 Image Feature Extraction

4.3.1 Camera Calibration

In Chapter 3, the point coordinates in the image space are measured in meters or millimeters when we design the control law. However, in the real experiment, the location of the a blob with respect to the field of view is described in pixels. A camera calibration is required to acquire the intrinsic parameters of the camera to realize the conversion between the two quantities. We employ MATLAB Camera Calibrator App [52] to compute the camera intrinsic parameters, such as pixel dimension and principal points.

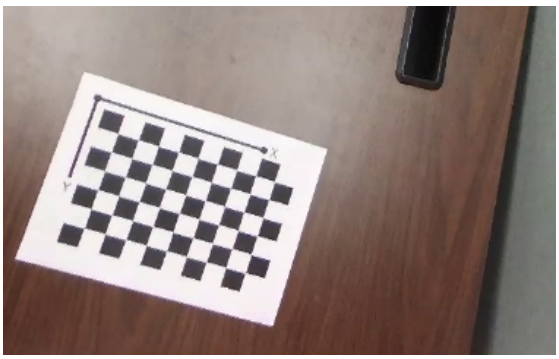
The camera model considered in the calibration is a standard camera model:

$$p_z^c \begin{bmatrix} u_p^c & n_p^c & 1 \end{bmatrix} = \begin{bmatrix} p_x^n & p_y^n & p_z^n & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_C^N \\ (\boldsymbol{t}_c^n)^{\mathrm{T}} \end{bmatrix} \boldsymbol{K} \tag{4.1}$$

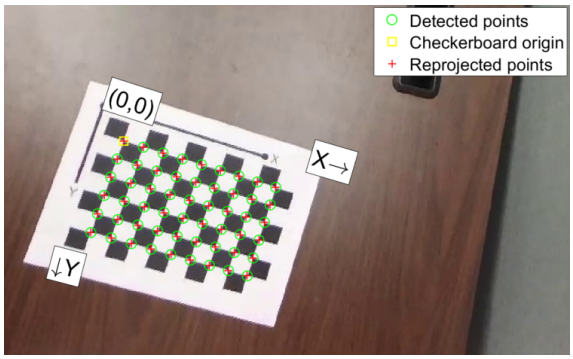
where u_p^c and n_p^c are the pixel coordinates of a point in the real image plane; $[p_x^n, p_y^n, p_z^n]$ is the 3-D coordinates of a point in the inertia frame; $(\boldsymbol{t}_c^n)^{\mathrm{T}}$ is a row vector that denotes the position of the origin of the inertia frame O_n expressed in the real camera frame $\mathbf{C}$. $\boldsymbol{R}_C^N$ and $(\boldsymbol{t}_c^n)^{\mathrm{T}}$ are referred to as the extrinsic parameters of the camera, and the intrinsic parameters are characterized by the intrinsic matrix $\boldsymbol{K}$ which is defined as:

$$\boldsymbol{K} = \begin{bmatrix} \lambda m_u & 0 & 0 \\ \gamma & \lambda m_n & 0 \\ c_u & c_n & 1 \end{bmatrix} \tag{4.2}$$

where λ is the focal length of the camera, expressed in millimeters; m_u and m_n are the scaling factors relating the distance to pixels, expressed in pixels/millimeters; γ is the skew parameter, which is equal to 0 when u axis and n axis are perpendicular,



(a) Checkerboard pattern used for camera calibration.



(b) Processed checkerboard pattern with detected points.

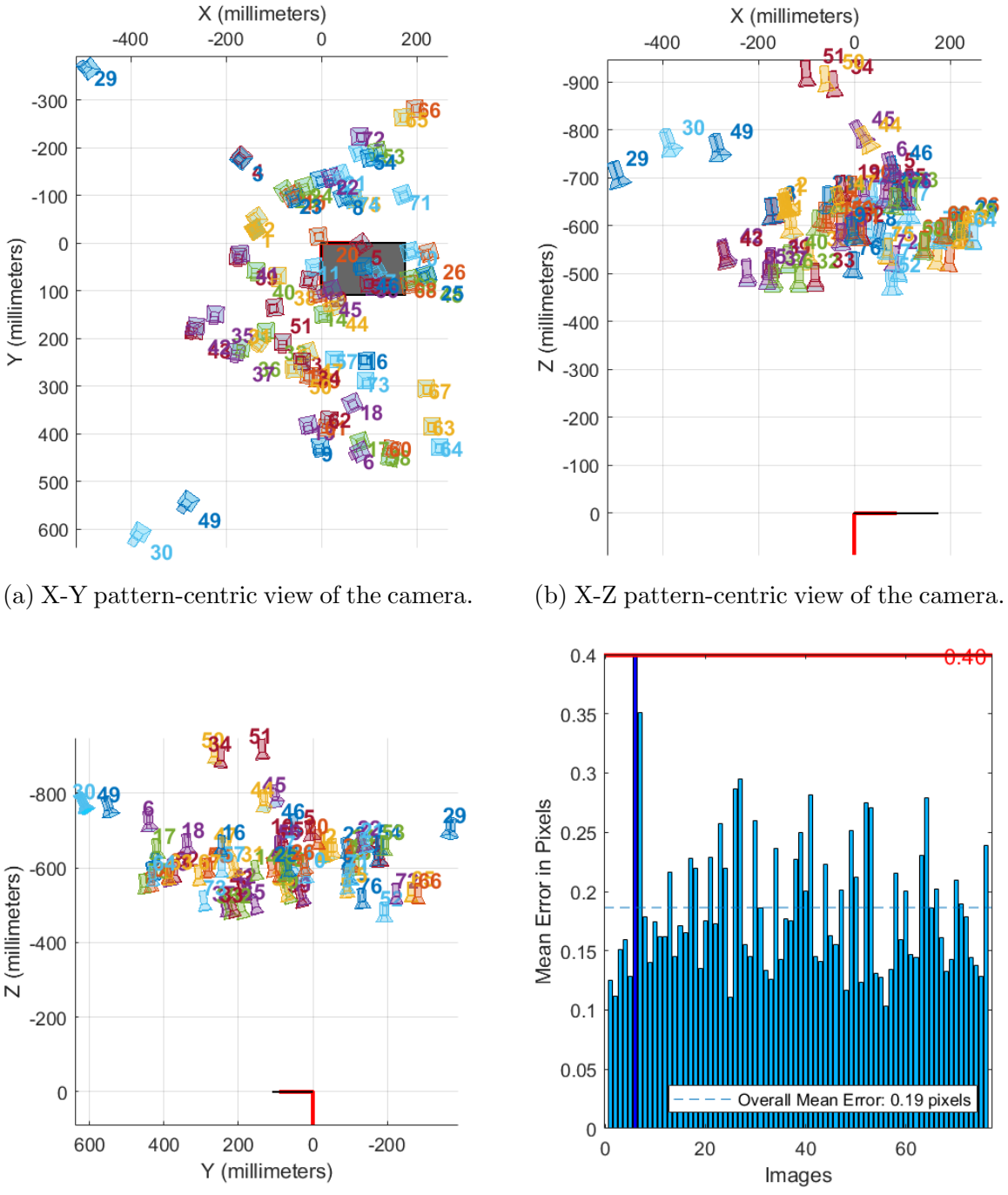
Figure 4.4: Camera calibration using checkerboard pattern.

and (c_u, c_n) is the principal point that would be ideally in the center of the image. Notice that (4.1) is an augmented version of (2.15) and (2.16) which depict a perfect pin-hole camera and do not consider the distance to pixels conversion.

The camera calibration is performed with a checkerboard pattern, as shown in Figure 4.4(a). The side length of the checkerboard squares is 22 mm. The calibration algorithm first detects a set of points from each image, as shown in Figure 4.4(b). Then, based on (4.1), the intrinsic and extrinsic parameters of the camera can be solved using the least square minimization [52].

Since a large number of images increases the accuracy of the calibration, we take 76 photos of the checkerboard from a variety of distance and direction. The algorithm computes the re-projection error to evaluate the performance of the calibration. The re-projection error compares the detected points with the re-projected points which are obtained by projecting the checkerboard points from the world coordinate frame into the image coordinate frame. As shown in Figure 4.5(d), the calibrated camera model gives an average re-projection error of 0.19 pixel that is considerably below the acceptable level, i.e., 1 pixel. Figures 4.5(a)–4.5(c) show the pattern-centric views which illustrate the relative position and orientation of the camera to the checkerboard

pattern. These figures are consistent with how the camera is actually positioned when 76 photos are taken.



(a) X-Y pattern-centric view of the camera.

(b) X-Z pattern-centric view of the camera.

(c) Y-Z pattern-centric view of the camera.

(d) Re-projection errors.

Figure 4.5: Camera calibration results.

The intrinsic matrix is obtained as:

$$\boldsymbol{K} = \begin{bmatrix} 535.32 & 0 & 0 \\ 0 & 525.75 & 0 \\ 434.46 & 238.34 & 1 \end{bmatrix} \tag{4.3}$$

It is observed that the principal point $(434.46, 238.34)$ is not exactly at the center of the image, which has the dimension of 856 pixels $\times$ 480 pixels. The focal length of the camera is given as 1.8 mm [53], and therefore m_u and m_n can be calculated as 297.4 pixels/mm and 292.1 pixels/mm, respectively.

4.3.2 Image Processing Algorithms

A flow chart is created to illustrate the procedures of the experiment, as shown in Figure 4.6. The designed procedures prepare the quadrotor and the camera for triggering the visual servoing mode.

The visual servoing process is illustrated by another flow chart, as shown in Figure 4.7. Since it cannot be guaranteed that every time the point coordinates are correctly extracted from the image, a loop structure is employed to make sure that valid moment information is used in the control algorithm. Moreover, the state error is constantly monitored to identify whether the visual servoing task has been accomplished. After the state error has decreased to below a threshold value, the quadrotor is commanded to perform automatic landing.

The point coordinates extraction algorithm is shown in Algorithm 1. It is observed that not only the blobs but also the contour of the white paper are detected in the algorithm. This is because the detection of the white paper contour helps to narrow down the area where the valid blobs are being searched. The first step of point coordinates extraction is to do thresholding and search through all the found contours

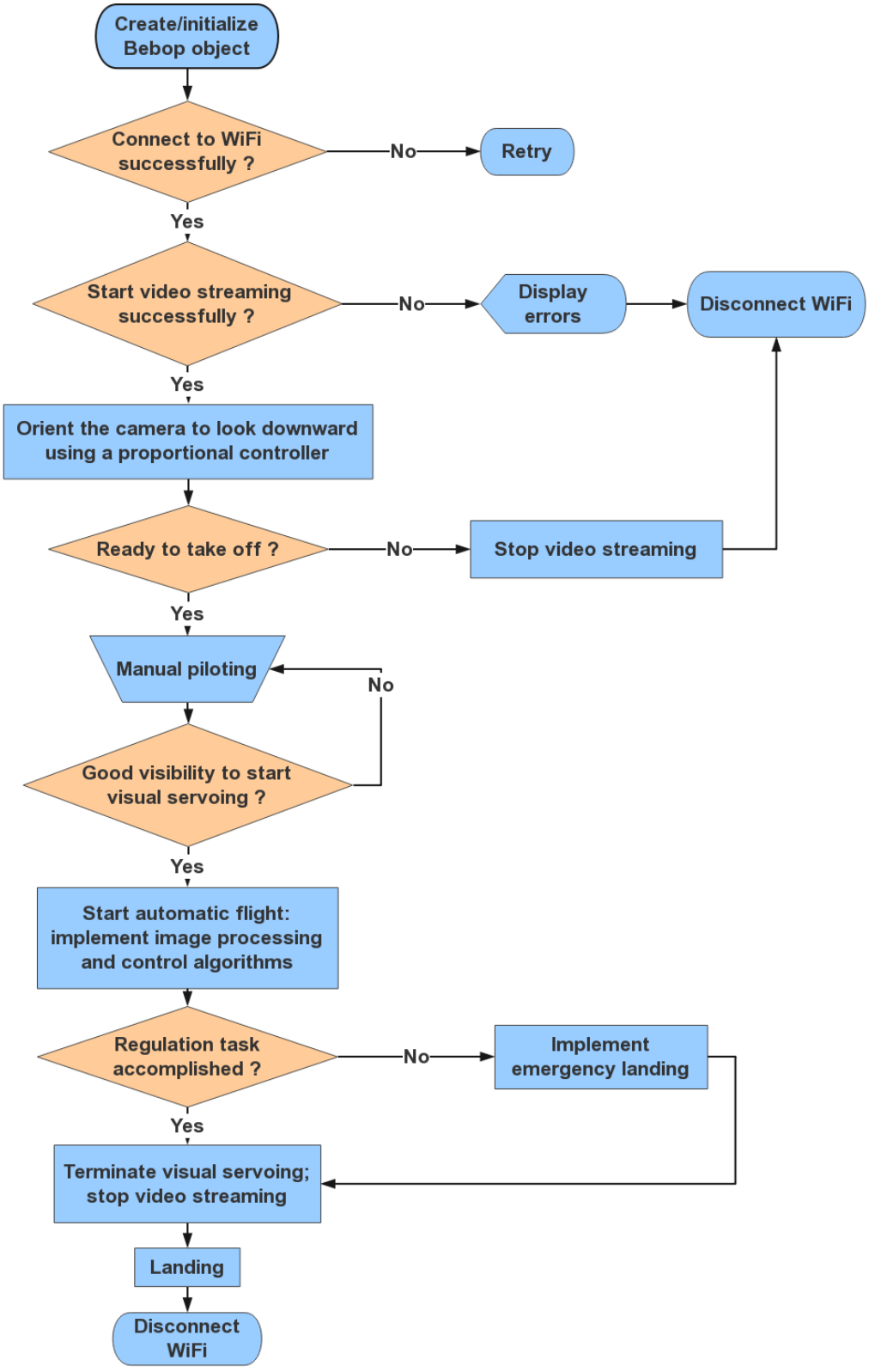


Figure 4.6: Experiment implementation flow chart.

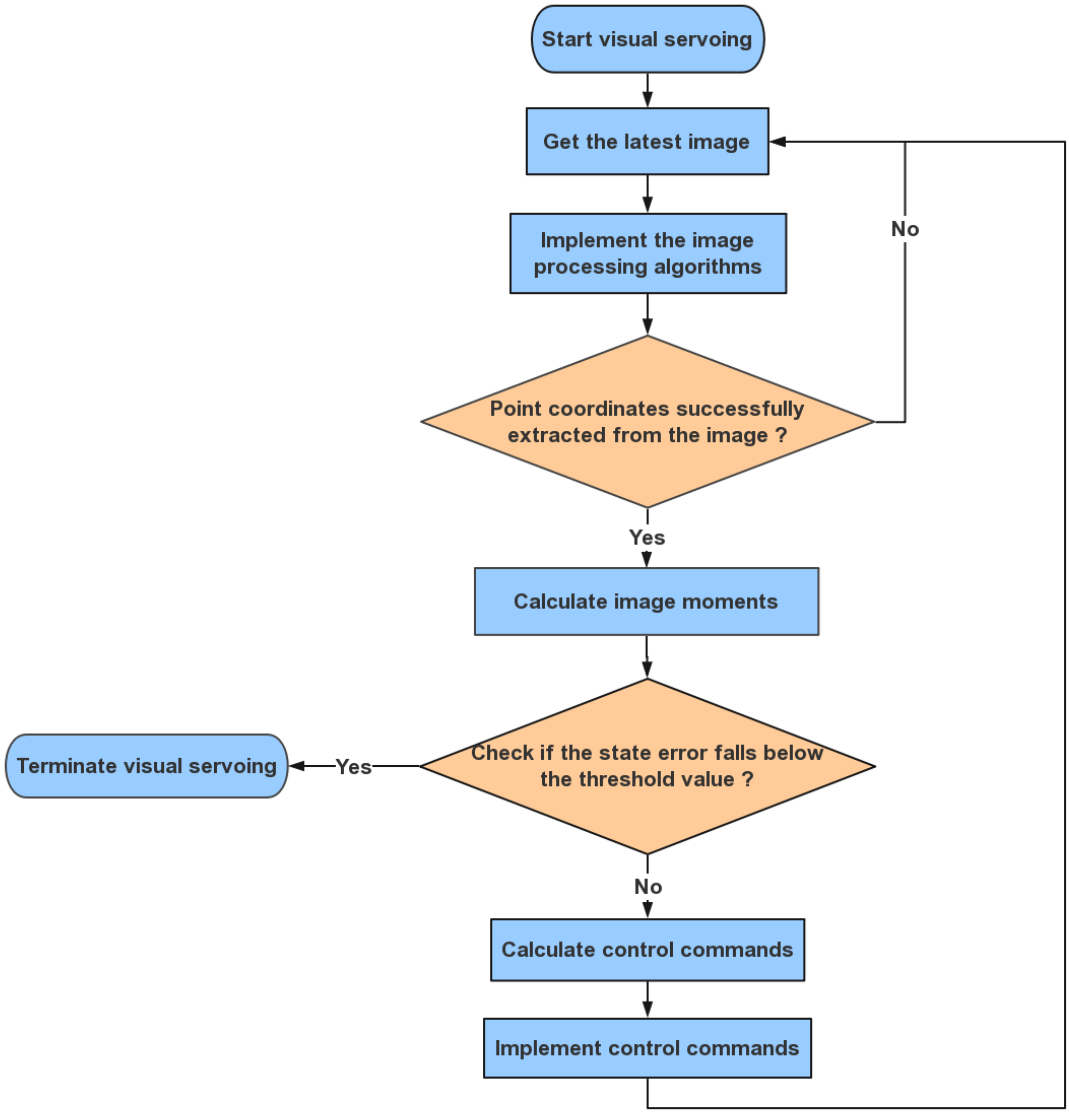


Figure 4.7: Visual servoing algorithm flow chart.

to find a parent contour that contains multiple valid child contours (see Algorithm 2). Then, the four corners of the parent contour are extracted using Algorithm 3, and these coordinates help to estimate the missing blobs using homography method. This auxiliary procedure complements the imperfection of Algorithm 2 when confronting varieties of lighting conditions.

Algorithm 1 Point Coordinates Extraction Algorithm

Input: A color image

Output: Coordinates of the four valid blobs

1: Read the color image as a grayscale image
2: Search for a parent contour that contains more than one but less than five valid child contours through iterations of thresholding (Algorithm 2)
3: Compute the convex hull of the valid parent contour
4: Find four corner coordinates of the rectangle simplified from the convex hull (Algorithm 3)
5: **if** The valid parent contour contains less than 4 valid child contours **then**
6: Compute the remaining blob coordinates based on the four corner points using homography method
7: **end if**

Algorithm 2 Find Valid Parent Contour Algorithm

Input: A grayscale image, $threshold\_value = 120$

Output: A valid parent contour containing valid blobs

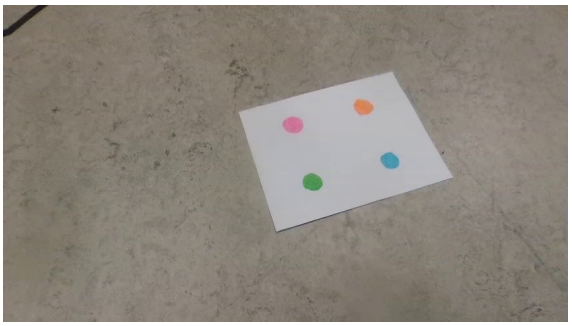
1: **repeat**
2: Threshold the image by $threshold\_value$
3: Find all the contours and generate the hierarchy tree
4: Search through the hierarchy tree to find the parent contour whose area ≤ 100000, convexity ≥ 0.92, and with the most number of valid child contours satisfying that $30 \leq$ area (child contours) ≤ 1000
5: $threshold\_value = threshold\_value + 10$
6: **until** A parent contour that contains more than one but less than five valid child contours is found

Algorithm 3 Find Rectangle Algorithm

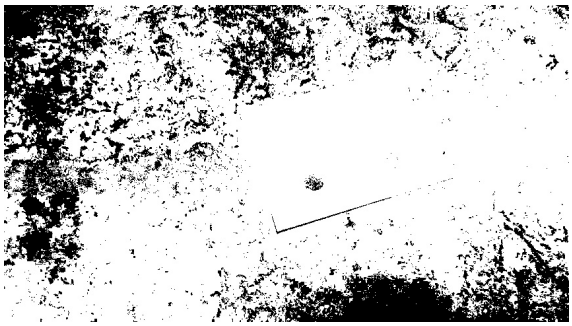
Input: Convex hull of the parent contour

Output: Corner coordinates of a rectangle contour

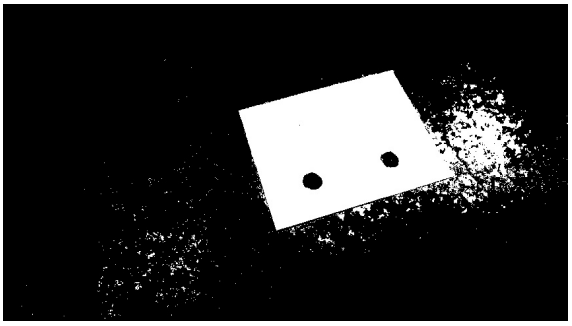
1: **repeat**
2: **for** each point P on the convex hull **do**
3: Compute the normal distance from P to the line connecting the points before and after P
4: **end for**
5: Remove the point P with the smallest normal distance
6: **until** Only four points remain



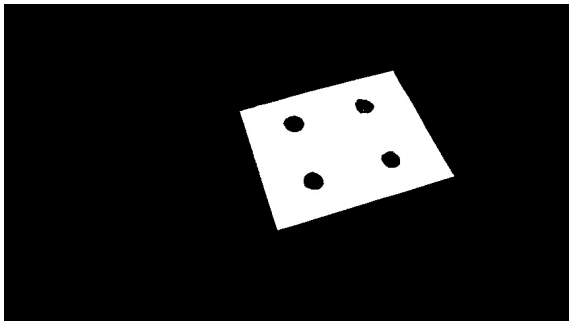
(a) Original image taken by the on-board camera.



(b) Processed image after thresholding at 120.



(c) Processed image after thresholding at 140.

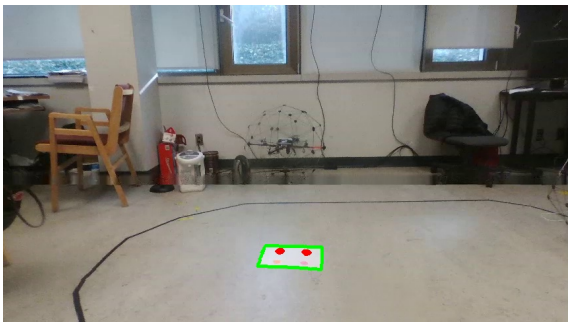


(d) Processed image after thresholding at 160.

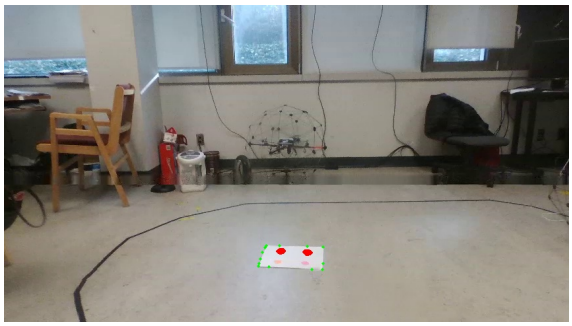
Figure 4.8: Illustration of effects of thresholding at different threshold levels.

To demonstrate the effects of thresholding, Figure 4.8 shows the processed images obtained by executing thresholding at different levels. As the thresholding level increases, only the white paper remains to be white, while the other parts in the image turn to be black, which benefits the contour of the white paper being detected.

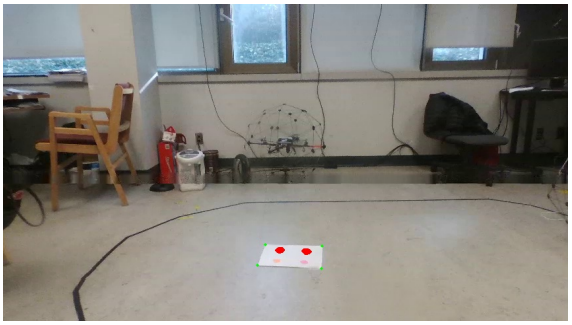
As shown in Figure 4.9(a), Algorithm 2 can accurately detect the parent contour containing the most number of valid blobs. However, it cannot be guaranteed that all the four target blobs can be detected by incremental thresholding. To address this issue, the corner coordinates of the parent contour are used to estimate the remaining undetected blobs using homography method given the knowledge of the location of the blobs with respect to the white paper (see Figures 4.9(c), 4.9(d)). It is worth noting that the geometric information of the target object is not used in our controller design



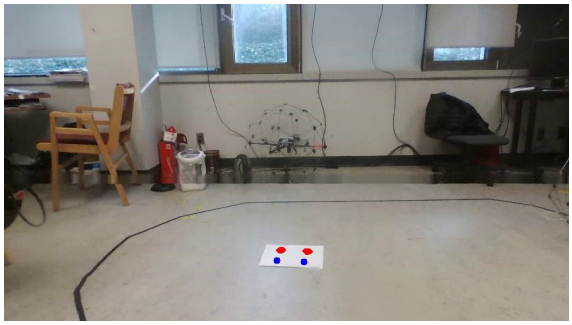
(a) Detect a valid parent contour through incremental thresholding.



(b) Extract the convex hull from the parent contour.



(c) Identify the largest inner approximate rectangle of the convex hull.



(d) Estimate the remaining child contours using homography.

Figure 4.9: Illustration of procedures of image feature extraction.

as assumed by the case of IBVS. It is exploited to improve the success rate of the point extraction algorithm subject to undesired illumination conditions. In addition, notice that we used different colors for the target blobs, but actually there is no need to differentiate the blobs as only high level information of the blobs is needed. That is to say, we can paint the target blobs all in black, which would significantly increase the likelihood that all the target blobs are detected by Algorithm 2.

4.3.3 Consideration of Model Mismatch

In Section 2.2, we assume that the origin of the virtual camera frame O_v coincides with that of the real camera frame O_c. This assumption is not valid as the camera is mounted in the front of the drone body, and has a considerable displacement from

the center of gravity (COG). This displacement should be taken into account in our analysis, otherwise it would induce steady state error in the regulation task and lead to imperfect landing.

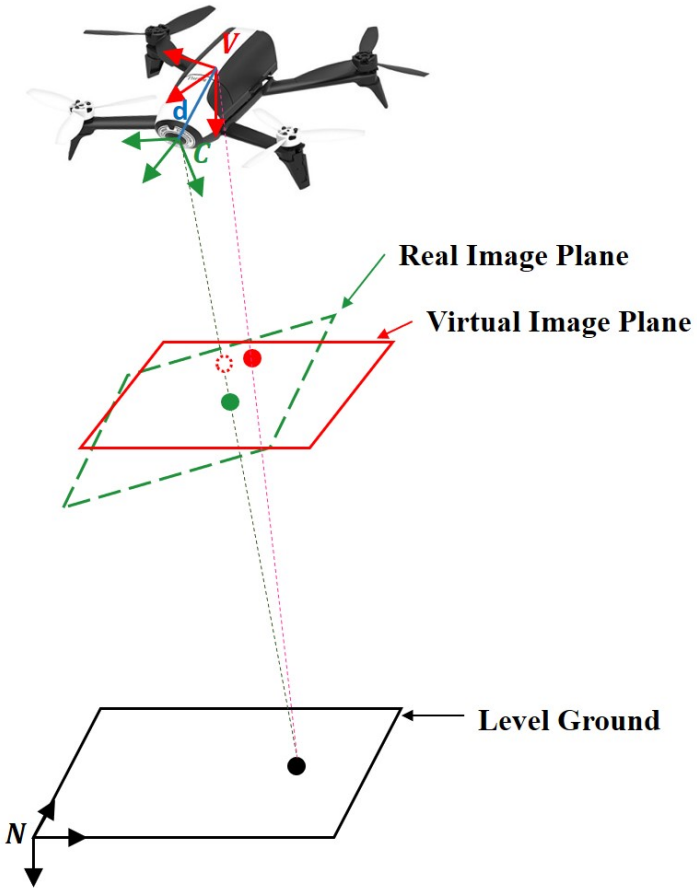


Figure 4.10: Coordinate frame modelling mismatch.

Since the control objective of the experiment is to regulate the COG of the quadrotor to right above the barycenter of the target object, it is appropriate to attach the origin of the virtual camera to the COG of the drone body, while keeping the origin of the real camera frame at the optical center, as shown in Figure 4.10. The displacement from O_v to O_c is denoted by d. The coordinates in the real and virtual image planes are symbolized by the green dot and the red dot, respectively. Moreover, the red dashed circle denotes the projection of the green dot onto the virtual image plane

as defined in (2.20) and (2.21), and it differs from the red dot due to the presence of d. This difference should be considered in our image moments calculation to mitigate the steady state error induced by the model mismatch. The point coordinates in the newly defined virtual image plane are derived as below:

$$\begin{aligned}
\begin{bmatrix} u^v & n^v & \lambda \end{bmatrix}^{\mathrm{T}} &= \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \left(\boldsymbol{P}^c + \boldsymbol{d}^c\right)} \left(\boldsymbol{R}_C^V \left(\boldsymbol{P}^c + \boldsymbol{d}^c\right)\right) \\
&= \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \left(\boldsymbol{P}^c + \boldsymbol{d}^c\right)} \left(\boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \left(\boldsymbol{P}^c + \boldsymbol{d}^c\right)\right) \\
&= \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \left(\begin{bmatrix} p_x^c & p_y^c & p_z^c \end{bmatrix}^{\mathrm{T}} + \boldsymbol{d}^c\right)} \left(\boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \left(\begin{bmatrix} p_x^c & p_y^c & p_z^c \end{bmatrix}^{\mathrm{T}} + \boldsymbol{d}^c\right)\right) \\
&= \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \left(\begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{p_z^c} \boldsymbol{d}^c\right)} \left(\boldsymbol{R}_C^V \left(\begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{p_z^c} \boldsymbol{d}^c\right)\right)
\end{aligned} \tag{4.4}$$

where $\boldsymbol{P}^c = [p_x^c, p_y^c, p_z^c]^{\mathrm{T}}$ is the coordinates of a point P with respect to frame $\mathbf{C}$; $\boldsymbol{d}^c = [d_1^c, d_2^c, d_3^c]^{\mathrm{T}}$ is the displacement from O_v to O_c, expressed in frame $\mathbf{C}$. The right-hand side of (4.4) can be simplified by assuming that the magnitude of $\boldsymbol{P}^c$ is considerably larger than $\boldsymbol{d}^c$ so that $\frac{\lambda}{p_z^c}\boldsymbol{d}^c$ can be neglected in the denominator.

Accordingly, it yields that:

$$\begin{aligned}
\begin{bmatrix} u^v & n^v & \lambda \end{bmatrix}^{\mathrm{T}} &\approx \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}}} \boldsymbol{R}_C^V \begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}}} \boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \boldsymbol{d}^c \\
&= \begin{bmatrix} \hat{u}^v & \hat{n}^v & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{\boldsymbol{E}_3^{\mathrm{T}} \boldsymbol{R}_C^V \begin{bmatrix} u^c & n^c & \lambda \end{bmatrix}^{\mathrm{T}}} \boldsymbol{R}_C^V \frac{\lambda}{p_z^c} \boldsymbol{d}^c
\end{aligned} \tag{4.5}$$

where $[\hat{u}^v, \hat{n}^v, \lambda]^{\mathrm{T}}$ is the point coordinates in the previously defined virtual image

plane when d is neglected. Notice that the last term on the right-hand side contains $\boldsymbol{R}_C^V$ and p_z^c, which are time-varying. Since we aim to mitigate the steady state error, we can simply consider the case at the desired pose, that is, $\boldsymbol{R}_C^V = \boldsymbol{I}_m$, and $p_z^c = Z$.

As a result, the right-hand side can be further reduced to:

$$\begin{bmatrix} u^v & n^v & \lambda \end{bmatrix}^{\mathrm{T}} \approx \begin{bmatrix} \hat{u}^v & \hat{n}^v & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{Z}\boldsymbol{d}^c \tag{4.6}$$

From (2.30), we have $Z = s_3 Z^*$. Hence, it yields that:

$$\begin{bmatrix} u^v & n^v & \lambda \end{bmatrix}^{\mathrm{T}} \approx \begin{bmatrix} \hat{u}^v & \hat{n}^v & \lambda \end{bmatrix}^{\mathrm{T}} + \frac{\lambda}{s_3 Z^*}\boldsymbol{d}^c \tag{4.7}$$

where $\frac{\lambda}{s_3 Z^*}\boldsymbol{d}^c$ is the mismatch term that should be considered in the moment calculation. $\boldsymbol{d}^c$ is a constant vector and is measured as $[0.12, 0, 0]^{\mathrm{T}}$ m. Since it only has one nonzero element d_1^c, it will only affect the calculation of u_g^v and further the calculation of s_1.

Based on (4.7) and the definition $u_g^v = \frac{1}{K}\sum_{k=1}^{K} u_k^v$, it is straightforward to derive that:

$$u_g^v \approx \hat{u}_g^v + \frac{\lambda d_1^c}{s_3 Z^*} \tag{4.8}$$

where $\hat{u}_g^v = \frac{1}{K}\sum_{k=1}^{K} \hat{u}_k^v$.

Then, based on (2.27a) and (4.8), the corrected form of s_1 is given as:

$$s_1 \approx s_3 \frac{\hat{u}_g^v}{\lambda} + \frac{d_1^c}{Z^*} \tag{4.9}$$

Replacing (2.27a) by (4.9) enables the model mismatch to be effectively compensated, and thus more accurate localization of the Bebop drone with respect to the ground target can be achieved. Furthermore, better performance in automatic landing and object tracking can be expected.

4.4 Experimental Results

The control objective of the experiment is to regulate the quadrotor to 1.35 m above the barycenter of the ground target with its yaw aligned with the principal axis of the target. Prior to the automatic flight, the quadrotor is manually piloted to the desired pose, where a^* is determined to be 5.5×10^{-8}.

The high-gain observer was not able to be deployed due to that it requires high frequency sampling of the image features, and thus imposing intensive computation load for image processing. Recall that the high-gain observer designed in (3.8) considers the continuous-time system dynamics (3.3), which implies that the acquisition of image features needs to be sufficiently fast to enable the convergence of the observer. However, since we execute both the image processing algorithm and the control algorithms on the ground station, it would be challenging to guarantee an uninterrupted supply of image features. In addition, the current image processing algorithm cannot guarantee 100% accuracy of the extracted point coordinates, and therefore it may deteriorate the performance of the high-gain observer which is sensitive to measurement noise.

Nonlinear MPC and explicit MPC are implemented in the experiments and the corresponding experimental results are presented in Section 4.4.1 and Section 4.4.2, respectively.

4.4.1 Experimental Study: Nonlinear MPC Controller

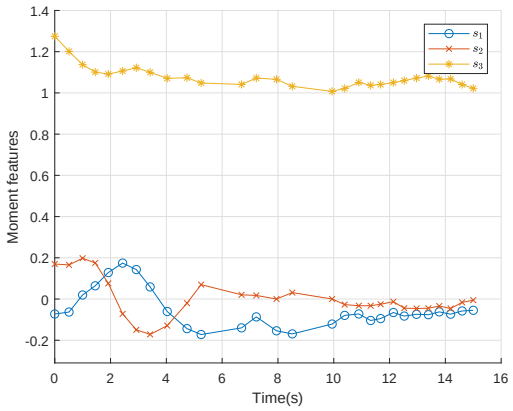
Online tuning has been executed to achieve a satisfactory flight performance in the sense that the tuned controller is responsive to the given setpoint and has certain robustness against the external disturbance, e.g., wind gusts. In the simulation, we assume that the prediction time step equals the sampling period, and we specify the

prediction horizon by the number of prediction steps. However, during online tuning, it is observed that an extended prediction time step can generate smooth maneuvers and effectively reduce the computational burden. In the experiment, we specify the prediction horizon by seconds and set it to 4 s. The control horizon is also set to 4, which indicates that four sets of control inputs are generated and each set of control inputs accounts for 1 s which is considerably larger than the sampling period. The first set of control inputs is sent to the on-board autopilot and implemented for 0.1 s. The tuning parameters are set as: $\boldsymbol{Q}$ =diag($[200, 200, 200, 2, 10, 10, 10]$); $\boldsymbol{R}$ =diag($[1, 1, 1, 1]$). Since it is not straightforward to specify the nonlinear constraints (3.25) and (3.26) in CasADi, instead we directly bound the control inputs. With reference to the definitions of ϕ_{max} and θ_{max} in (3.25), we can specify the bounds of states and control inputs as: $s_{1,max} = s_{2,max} = 2$; $s_{3,max} = 5$; $s_{4,max} = 3.14$; $v^v_{1,max} = v^v_{2,max} = v^v_{3,max} = 0.5$; $\psi_{max} = 0.06$; $\mathrm{f}_{1,max} = \mathrm{f}_{2,max} = \mathrm{f}_{3,max} = 2$.

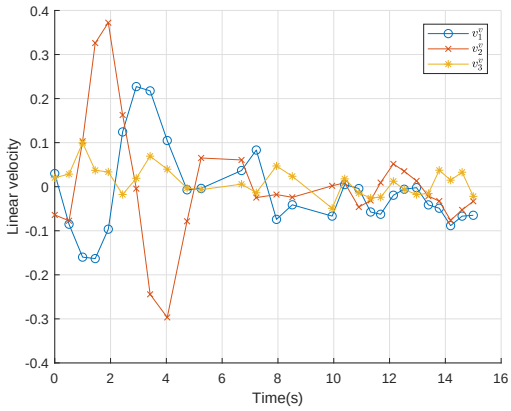
The experimental data is collected and plotted in Figure 4.11. As shown in Figures 4.11(a) and 4.11(b), the states are converging to the desired values, although small steady state errors can be observed in the end. This is because we design the visual servoing process to be terminated when the state error weighted sum in (3.19) has diminished to below 0.8. This threshold value is selected based on multiple successful trials with acceptable convergence time.

The trajectories of control inputs f_{1d}–f_{3d} are shown in 4.11(c), and the trajectories of s_4 and $\dot{\psi}_d$ are shown in 4.11(d) to illustrate the evolution of yaw. Different from the case using explicit MPC controller, the position and the yaw of the quadrotor are regulated simultaneously thanks to the use of a nonlinear model. Also, it is observed that all the states and control inputs are constrained within the prescribed bounds. In addition, the average sampling frequency is calculated to be 1.8 per second.

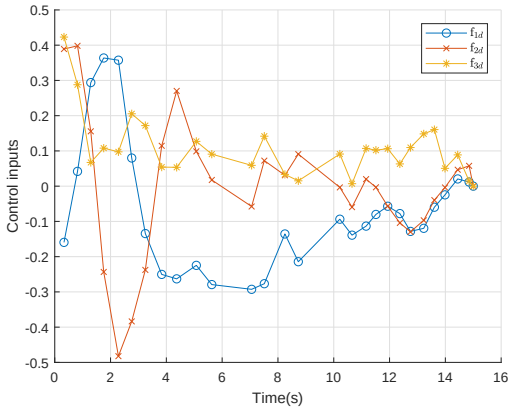
The evolution of the object scene in the real image plane is illustrated in Figure



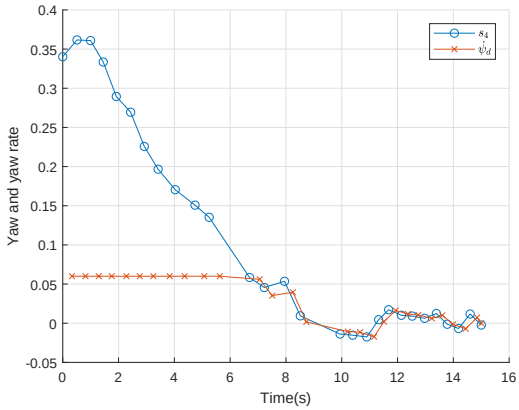
(a) Time evolution of moment features.



(b) Time evolution of linear velocity.



(c) Control inputs for translational dynamics.



(d) Time evolution of yaw.

Figure 4.11: Experimental results obtained by implementing nonlinear MPC.

4.12, in which blue circles represent the initial scene, and blue dots represent the final scene. By convention, the origin of the FOV is located at the left upper corner, and the two axes are measured in pixels. Because the camera is mounted at the front of the drone body, when the COG of the quadrotor is converging to right above the barycenter of the ground target, the object scene will not converge to the center of the FOV, but slightly closer to the bottom edge of the FOV. Besides, a video of this experiment is available at `https://www.youtube.com/watch?v=BE9NwYZGwT0`.

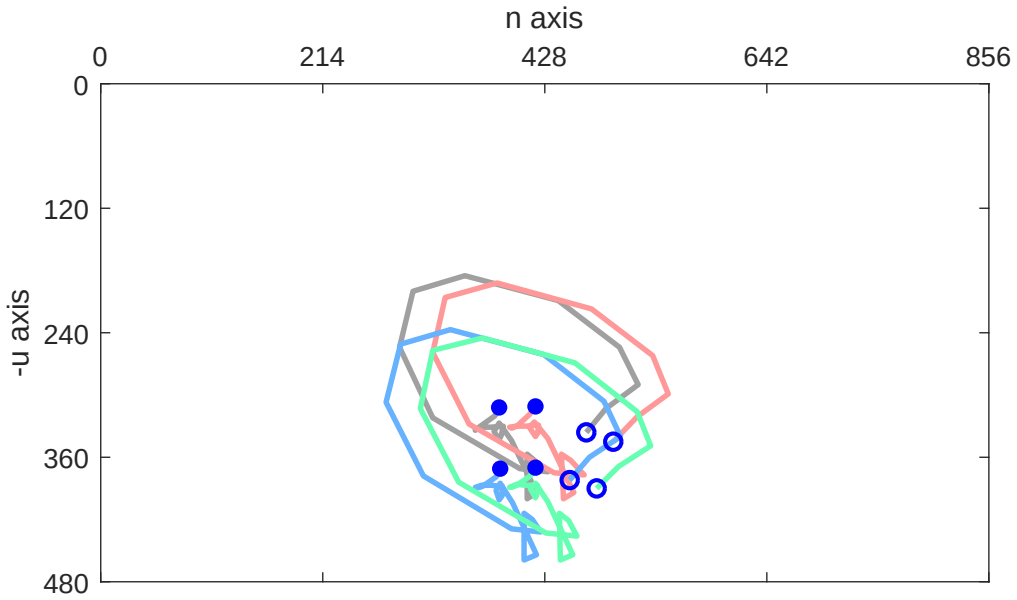


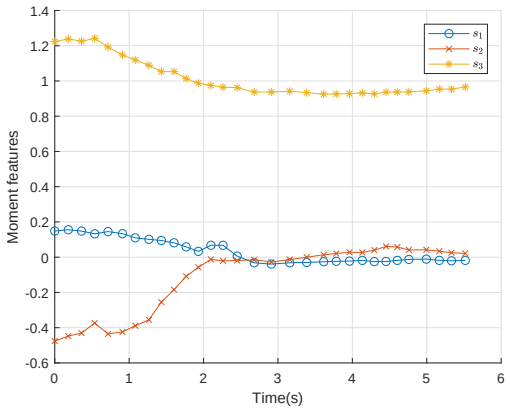
Figure 4.12: Point coordinate trajectories in the real image plane obtained by implementing nonlinear MPC.

4.4.2 Experimental Study: Explicit MPC Controller

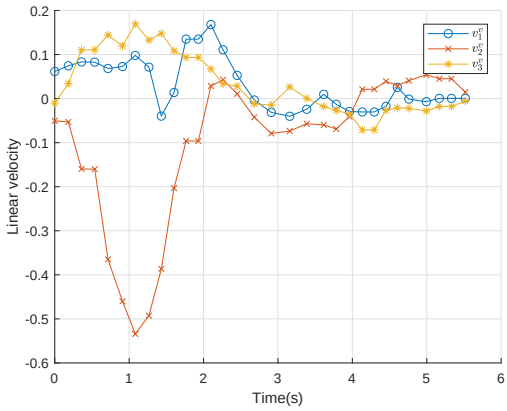
In this experiment, the tuning parameters are set identical to those used in the simulation in Section 3.8.4, except that K_ψ is set to 10.

Figures 4.13(a) and 4.13(b) illustrate the convergence of the states, and Figure 4.13(c) shows the evolution of the control inputs f_{1d}–f_{3d}. Figure 4.13(d)) shows that the yaw of the quadrotor remains unchanged during position regulation until 2.4 s, when the state error weighted sum has diminished below 0.1 and the yaw regulator is triggered. Then, the yaw regulation is accomplished at 3.6 s, and the explicit MPC controller is switched on again to alleviate the position deviation occurring during the yaw regulation.

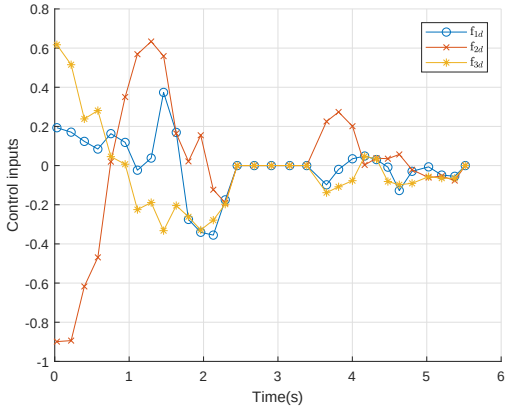
The average sampling frequency is 5.4 per second, which is three times faster than that of using nonlinear MPC. In addition, it is observed in the experiment that fast feedback enhances the robustness against the wind gusts.



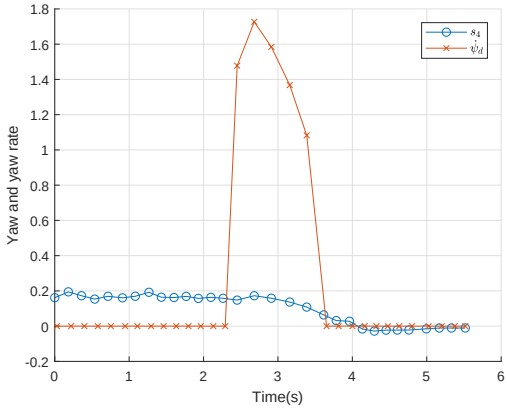
(a) Time evolution of moment features.



(b) Time evolution of linear velocity.



(c) Control inputs for translational dynamics.



(d) Time evolution of yaw.

Figure 4.13: Experimental results obtained by implementing explicit MPC

The evolution of the object scene in the real image plane is shown in Figure 4.14, in which blue circles represent the initial scene, and blue dots represent the final scene. It is observed that the object scene successfully converges to the desired configuration, and the final scene is comparable to what is obtained in Figure 4.12. Besides, a video of this experiment is available at `https://www.youtube.com/watch?v=-zWwaA7CCgQ&t=10s`.

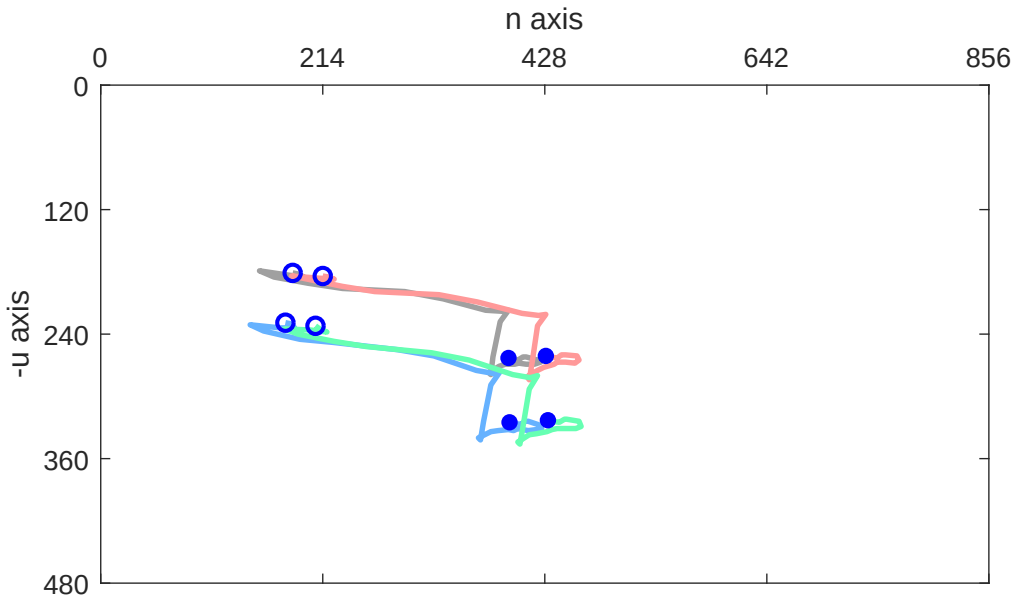


Figure 4.14: Point coordinate trajectories in the real image plane obtained by implementing explicit MPC.

4.5 Conclusion

Experiments have been executed to evaluate the performance of the proposed controller designs. In this chapter, the experimental set-up is firstly introduced. Then, the camera calibration is performed to determine the intrinsic parameters of the camera, as required by the evaluation of control action. Next, the image processing algorithms are presented to illustrate how the image features are extracted from the images. Furthermore, considering that there exists mismatch between the model and the real system, a compensation term is suggested being added to the moment calculation to reduce the induced steady state error. Lastly, the experimental results are presented to show the validity of the proposed controller designs.

Chapter 5

Conclusions

5.1 Concluding Remarks

In this thesis, we propose control algorithms that address challenging problems arising in the field of image-based visual servoing of a quadrotor. Due to the under-actuation nature of the quadrotor, the conventional image feature and control method cannot be directly applied to the visual servoing of the quadrotor. In **Chapter 2**, we adopt the virtual camera approach and select four image moments defined in the virtual image plane as image features. The resulting image kinematics is independent of the roll/pitch of the quadrotor, which gives rise to a nicely decoupled system dynamics. The image kinematics together with the dynamics of the quadrotor comprise the equations governing the IBVS of the quadrotor.

In **Chapter 3**, a high-gain observer-based nonlinear model predictive control scheme is proposed for the IBVS of the quadrotor. The high-gain observer is employed to estimate the linear velocity of the quadrotor which is part of the system states. The nonlinear model predictive controller generates the desired force to regulate the quadrotor to the desired pose while ensuring the tilt motion of the quadrotor

to be effectively constrained. The constraint on the tilt motion of the quadrotor effectively maintains the target object within the field of view of the camera. Considering that a stringent constant constraint may induce conservative maneuver and slow convergence, an adjustment law is designed to automatically adjust the roll/pitch constraints based on the visual feedback. However, the nonlinear MPC controller requires an optimization problem to be solved in real-time, which imposes intensive computation load in the online implementation. Instead, an explicit MPC controller is developed to improve the online implementation and alleviate the time delay. Moreover, a dynamics inversion-based PD controller is presented to demonstrate a feasible solution for the inner-loop attitude tracking control. The simulation results show the validity of the proposed control strategies.

Experiments have been conducted to further verify the effectiveness of controller designs, as shown in **Chapter 4**. We first introduce the experimental set-up, which includes the hardwares and softwares. Then, to accurately extract the image features from the images, we use a MATLAB camera calibrator toolbox to determine the intrinsic parameters of the camera. Following this, the image processing algorithms are illustrated to show how the image features are extracted from the images. Moreover, since there exists mismatch between the model and the real platform, modification in the image moments evaluation has been suggested to effectively compensate the steady state error. Lastly, experimental results are presented to show the validity of the proposed control strategies.

5.2 Future Work

In this thesis, an observer-based MPC control scheme has been proposed for the image-based visual servoing of a quadrotor. Simulation and experiments have been

executed to verify the effectiveness of the proposed control schemes. Despite that the convergence of the high-gain observer has been proved in Section 3.3.2, the proof for the stability of the overall closed-loop system has not been provided. [54] suggests that terminal constraints can be introduced to guarantee the recursive feasibility and asymptotic stability of the MPC controller subject to time-invariant input constraints. However, a comprehensive proof will need to address the following difficulties:

- Stability analysis of the MPC controller subject to time-varying input constraints when the roll/pitch adjustment law is implemented.
- Stability analysis of the MPC controller when the observation error in the linear velocity estimation is taken into account.
- Stability analysis of the overall closed-loop system when the inner-loop attitude tracking error is taken into account.

Hence, future work will focus on the stability proof by addressing the above-mentioned challenges.

In this thesis, we argue that visibility can be significantly improved by properly bounding the tilt motion of the quadrotor. However, this strategy would induce hover-like motion, thus limiting the agility of the quadrotor. The authors of [55] propose a PBVS control scheme that compensates the quadrotor's tilt motion by moving upwards so that the scene projected within the FOV is increased. This strategy can lead to agile maneuver that offers faster convergence. To the best of author's knowledge, none of IBVS controllers can provide such motion in the literature. Hence, more efforts are to be made to address the visibility constraint without sacrificing the agility of the quadrotor.

Bibliography

[1] F. Chaumette and S. Hutchinson, "Visual servo control part I: Basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[2] W. Wilson, C. Hulls, and G. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 684–696, Oct. 1996.

[3] N. Gans and S. Hutchinson, "An asymptotically stable switched system visual controller for eye in hand robots," in *Proceedings of the 16th IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, USA, 2003, pp. 735–742.

[4] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 D visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Apr. 1999.

[5] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, Aug. 2004.

[6] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1116–1127, 2005.

[7] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[8] M. Sauvee, P. Poignet, E. Dombre, and E. Courtial, "Image based visual servoing through nonlinear model predictive control," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, 2006, pp. 1776–1781.

[9] A. Hajiloo, M. Keshmiri, W. Xie, and T. Wang, "Robust online model predictive control for a constrained image-based visual servoing," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2242–2250, Apr. 2016.

[10] S. Heshmati-alamdari, G. C. Karras, A. Eqtami, and K. J. Kyriakopoulos, "A robust self triggered image based visual servoing model predictive control scheme for small autonomous robots," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 5492–5497.

[11] J. Gao, A. A. Proctor, Y. Shi, and C. Bradley, "Hierarchical model predictive image-based visual servoing of underwater vehicles with adaptive neural network dynamic control," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2323–2334, Oct. 2016.

[12] F. Chaumette and S. Hutchinson, "Visual servo control part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[13] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: An image-based approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 187–198, Apr. 2002.

[14] I. Fantoni, R. Lozano, and F. Kendoul, "Asymptotic stability of hierarchical inner-outer loop-based flight controllers," in *Proceedings of the 17th International Federation of Automatic Control World Congress*, Seoul, Korea, 2008, pp. 1741–1746.

[15] F. Le Bras, T. Hamel, R. Mahony, and A. Treil, "Output feedback observation and control for visual servoing of VTOL UAVs," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 9, pp. 1008–1030, Jun. 2011.

[16] H. Xie and A. F. Lynch, "Input saturated visual servoing for unmanned aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 952–960, Apr. 2017.

[17] H. Jabbari Asl and J. Yoon, "Bounded-input control of the quadrotor unmanned aerial vehicle: A vision-based approach," *Asian Journal of Control*, vol. 19, no. 3, pp. 840–855, May 2017.

[18] D. Zheng, H. Wang, W. Chen, and Y. Wang, "Planning and tracking in image space for image-based visual servoing of a quadrotor," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3376–3385, Apr. 2018.

[19] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 743–749, Jun. 2009.

[20] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, Apr. 2008.

[21] H. Jabbari, G. Oriolo, and H. Bolandi, "Dynamic IBVS control of an underactuated UAV," in *Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics*, Guangzhou, China, 2012, pp. 1158–1163.

[22] H. Jabbari Asl, G. Oriolo, and H. Bolandi, "Output feedback image-based visual servoing control of an underactuated unmanned aerial vehicle," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 228, no. 7, pp. 435–448, Aug. 2014.

[23] H. Xie, G. Fink, A. F. Lynch, and M. Jagersand, "Adaptive visual servoing of UAVs using a virtual camera," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2529–2538, Oct. 2016.

[24] X. Zhang, Y. Fang, X. Zhang, J. Jiang, and X. Chen, "A novel geometric hierarchical approach for dynamic visual servoing of quadrotors," *IEEE Transactions on Industrial Electronics*, to be published.

[25] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, Oct. 2010.

[26] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *Journal of Dynamic Systems, Measurement, and Control*, vol. 130, no. 1, pp. 011 005–01–011 005–12, Dec. 2007.

[27] NaturalPoint, Inc. OptiTrack Homepage. https://optitrack.com (accessed Nov. 1, 2019).

[28] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[29] F. L. Bras, R. Mahony, T. Hamel, and P. Binetti, "Dynamic image-based visual servo control for an aerial robot: theory and experiments," *International Journal of Optomechatronics*, vol. 2, no. 3, pp. 296–325, Sep. 2008.

[30] R. Mebarki, V. Lippiello, and B. Siciliano, "Nonlinear visual control of unmanned aerial vehicles in GPS-denied environments," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1004–1017, Aug. 2015.

[31] R. Mebarki and B. Siciliano, "Velocity-free image-based control of unmanned aerial vehicles," in *Proceedings of the 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Wollongong, Australia, 2013, pp. 1522–1527.

[32] D. Zheng, H. Wang, J. Wang, S. Chen, W. Chen, and X. Liang, "Image-based visual servoing of a quadrotor using virtual camera approach," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 972–982, Apr. 2017.

[33] H. Xie, K. H. Low, and Z. He, "Adaptive visual servoing of unmanned aerial vehicles in GPS-denied environments," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2554–2563, Dec. 2017.

[34] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.

[35] A. N. Atassi and H. K. Khalil, "A separation principle for the stabilization of a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 9, pp. 1672–1687, Sept. 1999.

[36] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss, "Output feedback stabilization of constrained systems with nonlinear predictive control," *International Journal of Robust and Nonlinear Control*, vol. 13, no. 3, pp. 211–227, Feb. 2003.

[37] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss, "State and output feedback nonlinear model predictive control: an overview," *European Journal of Control*, vol. 9, no. 2, pp. 190–206, Jan. 2003.

[38] H. Michalska and D. Q. Mayne, "Moving horizon observers and observer-based control," *IEEE Transactions on Automatic Control*, vol. 40, no. 6, pp. 995–1006, Jun. 1995.

[39] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Rigid motions and homogeneous transformations," in *Robot Modeling and Control*, 1st ed. Wiley, 2005, ch. 2, sec. 4, pp. 42–44.

[40] H. Jabbari, G. Oriolo, and H. Bolandi, "An adaptive scheme for image-based visual servoing of an underactuated UAV," *International Journal of Robotics and Automation*, vol. 29, no. 1, pp. 92–104, Jul. 2014.

[41] S. Zhao, "Time derivative of rotation matrices: A tutorial," *Computing Research Repository*, vol. abs/1609.06088, Sept, 2016. [Online]. Available: https://arxiv.org/abs/1609.06088 (accessed Mar. 11, 2019).

[42] T. Bresciani, "Modeling, identification and control of a quadrotor helicopter," Master's thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2008.

[43] D. Gross, W. Hauger, J. Schröder, W. Wall, S. Govindjee, "Dynamics of rigid bodies," in *Engineering Mechanics–Dynamics*, 3rd ed. Berlin, Germany, Springer, 2011, ch. 3, sec. 4, pp. 182–189.

[44] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear robust tracking control of a quadrotor UAV on SE(3)," *Asian Journal of Control*, vol. 15, no. 2, pp. 391–408, Mar. 2013.

[45] A. Tornambe, "High-gain observers for non-linear systems," *International Journal of Systems Science*, vol. 23, no. 9, pp. 1475–1489, Sept. 1992.

[46] H. K. Khalil and L. Praly, "High-gain observers in nonlinear feedback control," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 6, pp. 993–1015, 2014.

[47] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Multivariable control," in *Robot Modeling and Control*, 1st ed. Wiley, 2005, ch. 8, sec. 3, pp. 266–272.

[48] G. P. Incremona, A. Ferrara, and L. Magni, "MPC for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, Jun. 2017.

[49] Best Quadrotor. Parrot Bebop 2 Drone Review. https://www.best-quadcopter.com/reviews/2017/11/parrot-bebop-2-drone-review (accessed Nov. 15, 2019).

[50] A. McGovern. pyparrot 1.5.3 documentation. https://pyparrot.readthedocs.io/en/latest/# (accessed Oct. 4, 2019).

[51] CasADi Homepage. https://web.casadi.org/ (accessed Nov. 15, 2019)

[52] The MathWorks Inc. "Single camera calibrator app." R2019b Documentation. https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html (accessed Oct. 4, 2019).

[53] Parrot. "Focal length of camera." Parrot For Developers. https://forum.developer.parrot.com/t/focal-length-of-camera/311 (accessed Oct. 4, 2019).

[54] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, Oct. 1998.

[55] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor UAV," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, 2017, pp. 6199–6206.