Vision-Aided Inertial Navigation System Design for Indoor Quadrotors

by

Lianfeng Hou

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Control Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

The navigation task for unmanned aerial vehicles (UAVs), such as quadrotors, in an indoor environment becomes challenging as the global positioning system (GPS) and the magnetometer may provide inaccurate aiding measurements and the signals may get jammed. The navigation system design in this thesis integrates a visual navigation block with a inertial navigation system block, which adds information about aiding measurements information for indoor navigation design. The direct visual measurements are feature coordinates that are obtained from images taken from an onboard monocular camera with different positions in the 3D world space. The scaled relative pose measurements are generated through vision algorithm implementations presented in this thesis. The vehicle states are estimated using the extended Kalman filter (EKF) with inputs from a gyroscope and accelerometer. The EKF sensor fusion process combines inertial measurements and the visual aiding measurement to get an optimal estimation. This thesis provides two design results: one navigation system assumes that the 3D world feature coordinates are known and that the navigation system is map-based for the feature extraction. The other navigation system does not require prior knowledge of the feature location and captures the feature based on map-less vision algorithms with geometry constraints.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Alan Lynch for his guidance and support. His insight into the area of the navigation design inspires me and motivates me to conduct research. He encourages me to investigate the essential aspects of research problems and tackle them meticulously. I would like to thank Dr. Martin Jagersand in the Computing Science department for providing valuable suggestions and practical experiment help. I would also like to thank other professors including Dr. Tongwen Chen, Dr. Mahdi Tavakoli, Dr. Qing Zhao and Dr. Horacio Marquez in the Control System group for sharing their knowledge and experience in both the areas of research and course work. I would like to thank the lab technicians, Mr. Wing Hoy and Mr. Steve Drake for their kind help with the lab work. Moreover, I would like to give my thanks to other students and postdoctoral fellows,including Martin Barczyk, Bryan Godbolt, Nikolaos Vitzilaios, David Kastelan, Geoffery Fink, Ning Cao and Romeo Tatsambon in the Applied Nonlinear Control Lab (ANCL); and Camilo Perez, Azad Shademan, Xi Zhang and Huan Hu in the Vision Lab for their help and providing me with many good times.

I would like to thank my parents for their most selfless love and support. They provide me with the necessities of life, and have nursed me through physical illnesses and the emotional stresses of growing up.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An unmanned aerial vehicle (UAV) such as a helicopter or a quadrotor is an aircraft equipped with onboard processors, sensors, batteries, and other avionic system components. UAVs are widely used for surveillance and exploration in the harsh and cluttered environment such as the blast site covered with toxic chemicals, fulfilling multiple tasks. For example, quadrotors were used in the Fukushima Daiichi power plant in Japan [17] after the 2011 earthquake and tsunami. They were used to perform inspection of upper levels of buildings to check if they were going to collapse and other rescue tasks.

Quadrotor helicopters are becoming a popular platforms for UAV research, due to their simple structure and vertical taking off and landing (VTOL). A quadrotor has four rotors located in the vertex ends. The rotors are divided into two groups with opposite spinning directions, which is illustrated in Fig. 1.1. The quadrotor used for navigation system design from the Applied Nonlinear Control Lab (ANCL) is shown in Fig. 1.2.

Most quadrotors are small remote-controlled vehicles equipped with cameras. Relying on a direct line of sight to fly a quadrotor is problematic because a UAV can disappear in a house or behind obstacles while exploring an unknown environment. We want that the quadrotor is able to fly autonomously without manual control. The goal of autonomous flight is either hovering at a certain point or following a desired path in a robust and stable way. The design of autonomous flight relies on various nonlinear control techniques for indoor flights as well as quadrotor dynamic model identification [3]. One ex-

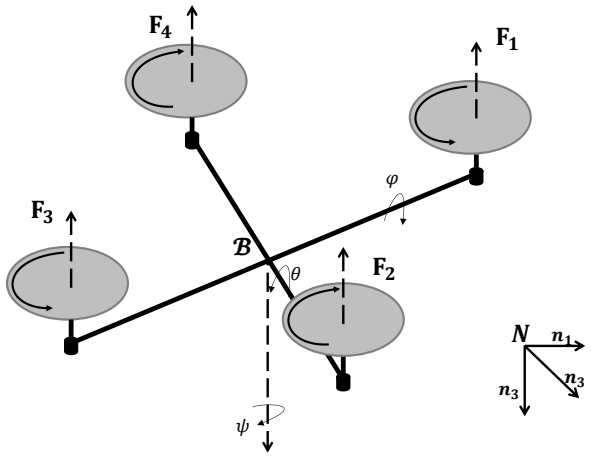


Figure 1.1: The schematic of a quadrotor

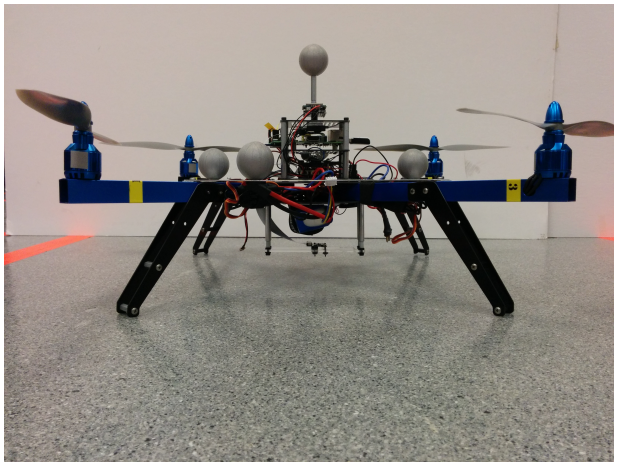


Figure 1.2: The quadrotor used for the navigation design

ample of nonlinear control design is to derive and implement a nonlinear backstepping controller to stabilize the inner loop of the quadrotor system and a proportional-derivative (PD) controller with a saturation function for the outer loop. The quadrotor control design is challenging because the quadrotor executes with six degrees of freedom but only obtains four independent inputs, i.e., four rotor speeds. Thus, quadrotors are inevitably underactuated. As important as the controller design itself, accurate feedback readings from sensors become a prerequisite.

The success of quadrotor autonomous flight depends on an accurate and reliable navigation system, the most essential components of which are the vehicle pose and motion estimation. The main problem for vehicle navigation designers is that it is difficult to achieve a highly consistent estimation of the vehicle state by fusing different sensor information. Typical estimated variables include positions, velocities and Euler angles. For the sake of successful quadrotor autonomous flight, different sensor combinations and their characteristics are explored in this thesis.

1.1 Background and Motivation

The goal of successful quadrotor navigation is twofold. On one hand, we need to obtain accurate estimates of the vehicle states, e.g., the positions, velocities and Euler angles. On the other, we need to force the vehicle to track the desired trajectories. For example, we want the vehicle to take off, hover or land in a designated way. The estimation problem for quadrotor navigation is often referred to as 'odometry' as in [15] and [40]. Traditionally, when UAVs are carrying out tasks in an open outdoor field, navigation systems are working with onboard sensors to provide state variable estimation. The main sensor used for obtaining the vehicle state is the inertial sensor, which is known as the inertial measurement unit (IMU). An IMU typically comprises three orthogonal accelerometers to measure the acceleration of the vehicle body, and three orthogonal gyroscopes to measure the rate of change of the body's orientation, also known as the angular velocities. By integrating the readings

of acceleration, we are able to obtain the velocity and thus the position state (after double integration). The angular velocity of the quadrotor is obtained directly from the rate gyro. Using the IMU as the core sensor is the best way to estimate the vehicle state because the IMU commonly has a faster sampling rate with lower time delays. The disadvantage of using only the IMU is that the IMU is subject to a slow gradual sensor drift.

It is normally not adequate to use a single sensor for the navigation task of quadrotors. For example, inertial sensors have significant measurement uncertainty at slow motion. In addition, when used as the visual sensor for navigation, the camera can only track detectable features accurately at low velocities. The indoor navigation occurs at low velocities. When the camera velocity increases, feature tracking and matching become less accurate due to the motion blur and the effect of lower camera sampling rates. The increase of the frame sampling rate that actually adds to the bandwidth may not help because it would complicate the real-time implementation.

To overcome these shortcomings, complementary filters are used to fuse different sensor measurements to obtain an optimal state estimation. Fusing different sensor measurements in an optimal way can maximize the advantage of each sensor and at the same time defuse each sensor's deficiency when they are being used alone.

One of the well-developed complementary filter schemes is to let the navigation system be aided by the Global Positioning System (GPS). Thus, the pose in the absolute scale [35], or as it is called in metric scale [17], is yielded through these sensor measurements. However, in narrow outdoor and indoor environments, the GPS is not fully functional because the signal is often shadowed by rock in a valley or tall buildings in urban areas. Also electromagnetic interference in urban areas can render other global aiding methods, such as aiding by the magnetometer, useless.

There are ways to overcome those shortcomings. One of the most powerful and resourceful methods is using visual information. Compared to the global aiding module, i.e., the GPS module, an onboard vision system involving a camera and images is more capable of geometry structure reconstruction in

indoor and urban environments. The onboard vision system imitates the human eyes and visual perception. It can provide rich sensory feedback and yield a full understanding of the surrounding environment, which aids the positioning and orientation. The current available vision algorithms such as those in the OpenCV library, allow the vision system to detect visual features in the real world such as corners, landmarks, lines, edges, bulks and feature motions. This complementary filter method is based on the fact that the quadrotor motion parameters can also be inferred from the image optical flow or scene features and structures. The advantages of this integrated visual-inertial navigation scheme are summarized as follows:

- Fusion of both the inertial sensor information and visual sensor information is able to give greater robustness to feature tracking.

- Fewer features are required to recover the camera and vehicle motions, because using random sample consensus (RANSAC) becomes unnecessary.

- The integrated system can reduce the ambiguity involved in recovering the camera motion. For example, as covered in Chapter 2 and Chapter 3, we can distinguish the correct motion from the essential matrix.

- The integrated system can help with the absolute scale estimation. Using visual sensors like monocular cameras cannot recover the absolute position of the camera and the quadrotor vehicle in the 3D world.

There are challenges with using vision as the aiding measurement for vehicle navigation. One challenge appears to be the limited onboard computation ability and the constrained storage room, as the most commonly seen processor is ARM-structured. Difficulties occur when processing high volume of visual data, especially the high resolution pictures. Also, in order to perform the control task in real time, we have to overcome the time delays generated through the transmission process, which interferes with the synchronization of the feedback signal for the control design.

Other challenges may be involved with the features in the scene. The features detected for the visual algorithm may disappear from the camera field of view (FOV). Sometimes we can barely detect any distinct features. For example, when the blue sky is the scenery view, minimum features can be detected and matched. Furthermore, vision-based method can reconstruct the observed scene and therefore helps with obstacle avoidance and path planning, which are more practical applications for the quadrotor vehicle.

The advantage of complementary filter design, i.e., using an integrated visual-inertial navigation scheme, is that the design can provide redundant measurements to estimate the state and improve the system reliability. A complementary filter, fusing the inertial measurement and visual measurement, can be designed in various ways. The integrated visual-inertial navigation system can be divided into two methods: loosely-coupled (LC) and tightly-coupled (TC).

The most popular integration method is LC. In this LC integration, the inertial sensor part and the visual sensor part operate separately, run at different sampling rates and then exchange information. Vehicle states such as positions, linear velocities and angular velocities are estimated from the inertial navigation block and can be used to predict feature motion. The motion estimation from the visual navigation block, i.e., typically a structure from motion (SfM) block, is used to bound the integration error and the inertial sensor drift in the inertial navigation block.

The tightly-coupled system is also described as a centralized system, which uses a single optimal filter rather than two filters for each sensor block. The TC system jointly processes the visual and inertial data in a single optimal filter for the state estimation. This means that when defining the filter state, the coordinates of the features are included in the same state vector where the quadrotor positions, velocities and orientations are stacked.

Since the quadrotor navigation system is nonlinear, we cannot claim that any of the three filter design methods outperforms the others. The three well-known filter design methods on state estimation for the navigation system include the extended Kalman filter (EKF), the unscented Kalman filter (UKF)

[48] and particle filters.

The first two are the most commonly used nonlinear filters in the navigation design [7]. The EKF is a classical design which uses the model linearization and usually serves as a first order estimator for the nonlinear quadrotor navigation system. When the system is highly nonlinear, the EKF estimator may diverge quickly. This is because the EKF relies on linearization to propagate the mean and covariance of the vehicle state.

Unlike the EKF method for predicting the behavior of the navigation system, the UKF design uses the Unscented transformation [9] as part of the estimation procedure. Based solely on estimation result accuracy, using UKF can provide better results than EKF.

The UKF uses a deterministic sampling approach. The additive Gaussian white noise is carved using a minimal set of carefully chosen sample points around the mean. These sample points capture the true mean and covariance of the error vector.

By using the Unscented transformation, the UKF can describe almost any nonlinearity up to the 3rd order, which is better than the linearization process by calculating the Jacobian matrices. Therefore, by using the UKF, the estimation results can converge more accurately to the true value.

Nonetheless, the UKF design is not a truly global approximation because it is executed based on a small set of trial points. And it does not work well with nearly singular covariance, i.e., with a nearly deterministic system. From the practical point of view, the UKF design method requires more computations than the classical EKF method, as we can see that it may requires Cholesky factorization on every step, which thus limits its application for real-time implementation.

In general, in order to achieve the goal of autonomous flight for the quadrotor in indoor or urban environments, it is beneficial to combine an inertial navigation system block and the vision SfM block. For this procedure, the sensor measurement data from all parties are processed in a single optimal EKF design loop.

1.2 Literature Review

This section provides a review of related work on visual and inertial navigation systems.

1.2.1 SLAM and Visual Odometry

Motion or pose estimation is a problem in quadrotor navigation. The most popular solution is called simultaneous localization and mapping (SLAM), which is an algorithm that aids in motion estimation. SLAM is defined as the problem of creating a map of the unknown environment while at the same time tracking the location of the robot vehicle within that map. SLAM is a process by which a visual map is used to determine the location of the robot vehicle and that an accurate vehicle pose estimation contributes to the map building of the unknown environment. One of the most popular methods for implementing the SLAM algorithm is using the EKF as an approximation solution [12].

The SLAM method is normally based on the monocular real-time SLAM method, which is proposed in [10, 11]. In [10], a monocular real-time SLAM is proposed for SfM work. The SfM work has been done successfully off-line. The propagated states include the camera position, the camera orientation represented by the quaternion, and the camera linear and angular velocities. The camera used is assumed to be a pinhole camera and the measured output is the image coordinates of feature points. The built map depends on the 3D feature coordinates and the orientation of the feature in the map. If full-covariance is used for EKF SLAM, the computational complexity of the filter update is of the order N_x^2, where N_x is the number of features in the map.

The SLAM can generally be called structure from motion (SfM) methods. The SfM is defined as a solution for both the scene structure reconstruction of the unknown environment and an estimation of the camera poses from unordered image frames. The final solution basically undergoes an offline bundle adjustment [47]. In addition to SLAM, another method for robotic vehicle navigation, is called Visual Odometry (VO) [39, 31, 40, 30], which estimates the

vehicle location relative to its mapped environment. The VO method essentially estimates the robot vehicle's pose states in an incremental way, through camera motion reflected on the image frames. The VO design keeps recording the trajectories of the pose states of a robot vehicle. As in [40], the flow chart of the VO is ıg a map

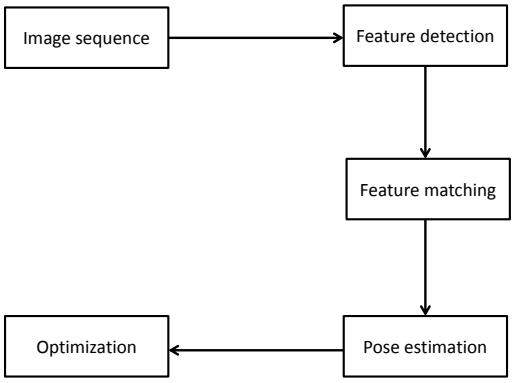


Figure 1.3: The flow chart of solving the visual odometry problem

of the unknown surrounding environment, its results are ordinarily considered as global and consistent estimation. The problem of consistency estimation is discussed in detail in [22, 31, 20]. As in [1], a state estimator is consistent if the estimation errors are zero-mean and have a covariance equal to the one calculated by the filter. In order to achieve the global results, SLAM requires the robot vehicle to execute in a closed loop trajectory which is used for global optimization. This loop closure detection contributes to enhancing the robustness of SLAM for the sake of reducing drifts in both the map and the camera pose state trajectories illustrated in Fig. 1.4. The blue line is the reference pose trajectory and the red dotted line the estimated pose trajectory. Due to the incremental error accumulation, there may be a gap in the estimated pose trajectory when SLAM revisits the same scene. The loop closure detection then refines the estimated robot map locations within this closed loop and builds maps as well.

VO, which is considered a more local concept [40] than SLAM with closed-loop, global optimization, estimates the robot vehicle trajectory incrementally

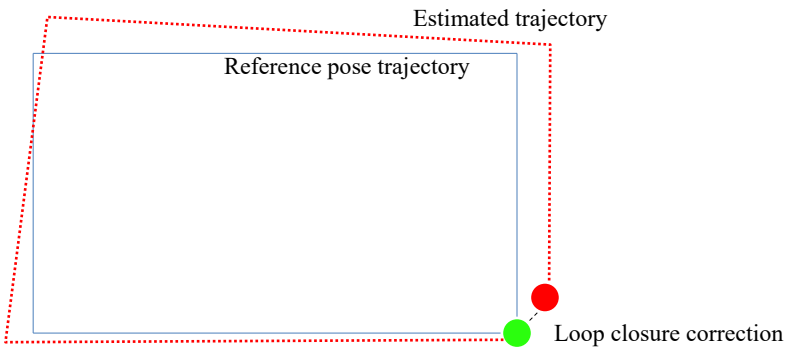


Figure 1.4: Loop closure detection for SLAM

and optimizes the pose estimation through windowed bundle adjustment. As a result, the closed loop detection becomes unnecessary for VO. The drawback of VO is that it trades off some accuracy of the vehicle pose estimation. The visual algorithms for SfM and VO are covered in Chapter 3.

A drawback of using only SLAM is that without the help of the known map, no absolute pose estimation can be obtained [46], and as a result, the whole vehicle system is unobservable. [46] assumes that features with known coordinates are available. The map-based pose estimation algorithm is thus provided with absolute-scale vehicle pose estimation.

Different versions of the SLAM library have been developed, most of which are compiled as packages of the Robot Operating System (ROS). The Parallel Tracking and Mapping (PTAM) ROS package is a typical application for SLAM. It can be modified for VO application and will then run as an SfM visual navigation block. The PTAM toolkit is presented in [29], proposing a method of estimating the camera pose in an unknown scene. This toolkit splits the pose trajectory tracking and mapping into two separate tasks and thus makes the computation more efficient.

In [50], a visual-inertial navigation system is proposed where PTAM is integrated with an optical-flow algorithm in the EKF filter loop. Results of the optical-flow algorithm compared to the ground truth are provided when the quadrotor vehicle is in an autonomous hover and the camera is pointing down.

1.2.2 Combination of Visual and Inertial Navigation Systems

The combination of visual and inertial navigation system, using the SLAM framework in the integrated navigation design, can be found in [28, 36, 25, 27]. The work presented in [36] gives the vision-aided navigation design in the EKF loop and the typical framework for visual-inertial odometry (VIO) where map-building is not absolutely essential. In this work, the monocular vision is used to reduce drifts of the pose states and it also performs nonlinear triangulation of landmarks from multiple camera pose constraints before the EKF filter loop update. This fusion work is known as the multi-state constraint Kalman filter (MSCKF). Other works based on the MSCKF include [32, 21, 20].

The visual-inertial combination in the SLAM framework is demonstrated in [25], where efforts have been made to improve the accuracy of the feature initialization and estimation. In this paper, the visual-inertial SLAM design is being used to build a map for an outdoor trajectory for a large area and the loop closure is detected. This paper gives the motion conditions for the visual-inertial navigation system under which the gravity, the IMU-to-camera calibration and other vehicle states are observable and can be estimated in real time. [27] also presents the visual-inertial SLAM work, but in the UKF framework. Unlike the observability analysis performed in [25], this work assigns the nonlinear observability analysis to the navigation system, which is modeled as a control-affine form using Lie derivatives. It shows that as well as the pose states, the biases, the camera-to-IMU calibration and the gravity are observable. Similar to this work, [50, 51] propose an onboard SLAM implementation on a micro aerial vehicle (MAV), using visual and inertial navigation system blocks, which can recover the absolute scale factor.

In contrast to the visual-inertial SLAM, one of the fundamental works for visual-inertial navigation is presented in [38]. This paper gives the real-time long-run implementation with a robust outlier rejection scheme. Image features are not tracked through frames but are detected independently. The matches between features in different frames are picked out. These matches

can reduce the feature drift. The RANSAC outlier rejection is used to exclude the false matching.

1.2.3 Stereo Vision Navigation Method

Other than using a monocular vision block, stereo vision has been widely adopted because a monocular camera cannot yield any depth information for feature points and has to observe features from more than one key frame. By adding a redundant camera and thus having a stereo rig, the visual navigation block imitates human vision and can acquire depth perception, provided that the unstructured scene is within a certain distance from the stereo camera rig for the sake of precise feature triangulation. The stereo vision based SLAM generates a dense map [37].

It is common for the stereo vision method that 3D points are triangulated for every stereo frame pair. Therefore, the relative motion for the robot vehicle is solved by determining the aligning transformation that minimizes the 3D-to-3D distance: i.e., the transformation $T_k(R_k,\ t_k)$ is given as

$$T_k = \underset{T_k}{\operatorname{argmin}} \sum_i \|\hat{X}_k^i - T_k \hat{X}_{k-1}^i\| \tag{1.1}$$

In [41] a quadrotor state estimation design is proposed which fuses the IMU with a time-delayed stereo vision sensor set. The stereo odometry runs independently, which feeds the EKF filter loop in real time. The key frame-based EKF with VO odometry gives an optimal, local and drift-free navigation and is computationally economical on onboard computers. In later work [42], the autonomous navigation design is tested in both indoor and outdoor unknown environments in the EKF, combining an inertial navigation block with a locally drift-free visual odometry navigation block.

1.3 Outline of the Thesis and Contributions

Chapter 2 provides the mathematical preliminaries for visual-inertial navigation system design. These mathematical formulas are used for the derivations

in the rest of the thesis. The coordinate frames are defined including the navigation frame N, the camera frame C and the body frame B. The coordinate transformation from between different frames is then introduced, which helps with the sensor modeling and the dynamic formation. The rotation kinematics and the properties are explored for the rotation matrix $R \in SO(3)$. These kinematic formulas and properties are crucial to the linearization in Chapter 4. The theory of computer vision, including the definition of the essential matrix E, the fundamental matrix F and the epipolar geometry are provided, which are the basic SfM theory used in the thesis. In addition, the modified eight-point algorithm is presented. The algorithm is used to yield visual aiding measurement for the visual navigation system block. The windowed bundle adjustment is also introduced. The windowed bundle adjustment is used to improve the accuracy of the visual aiding measurement.

Chapter 3 presents the modeling of the inertial and visual sensors. The modeling of the inertial sensor is based on the PX4FMU inertial platform used in ANCL lab, which consists of a set of orthogonal accelerometers and a set of gyroscopes. The inertial sensors are assumed corrupted by the white Gaussian noise. This noise is modeled by the random walk process rather than the first order Gaussian-Markov model. Chapter 3 also presents the camera model used for the visual navigation block. The camera is modeled for the Raspberry Pi camera module which is a perspective camera. The calibration procedures are explained in detail for retrieving the camera intrinsic parameters. Chapter 3 includes the method of sensor combination between the inertial sensors and visual sensors. The visual algorithms for generating the scaled visual pose measurements are covered for the sake of EKF implementation.

Chapter 4 provides the detailed EKF filter design and simulation results for both the map-based visual-inertial quadrotor navigation system and the map-less visual-inertial navigation system. Chapter 4 forms the dynamics for two kinds of navigation systems and the output equations for each system. As we use the EKF LC filter design method, the linearization procedure is provided to obtain the linearized error dynamics and implement the discrete-time Kalman filter. The observability analysis for the proposed visual-inertial navi-

gation systems is performed and proves the navigation system is locally weakly observable. The simulation results are provided and proves the effectiveness of the proposed navigation methods.

Chapter 5 summarizes the thesis and discusses the future research work directions. It outlines the scope of the optical flow-based method and the application of the three-view geometry constraint.

Chapter 2

Mathematical Preliminaries and Vision Algorithms

This chapter reviews the premise on mathematics that will be referred to in subsequent chapters. The chapter starts with the definition of the coordinate frames and the rotation matrix parameterizations. The mathematical transformation for vectors between different frames is provided elaborately in order to rigorously define coordinates and help with calibration.

2.1 Coordinate Frames

Because the problems investigated in this thesis are mainly involved with indoor navigation of a quadrotor, four coordinate frames are considered in which vectors can be interpreted. The four coordinate frames include the navigation frame, body frame, IMU frame and camera frame, which are denoted as N, B, I and C, respectively.

Navigation frame: The origin of the navigation frame N is located at a predefined point on the ground surface of the indoor experiment environment. The basis of $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$ of the navigation frame points to the north, east and down to the earth. The states of the quadrotor are estimated and expressed in this frame. The ground truth provided by the Vicon motion capture system is defined in this frame.

Body frame: The body frame, B, originates at the center of mass (COM) of the quadrotor. The body frame has its basis of $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ pointing to the

front, right and downside of the quadrotor. Due to the symmetric structure of the quadrotor as a whole, $\mathbf{b}_1$ and $\mathbf{b}_2$ are defined along two adjacent arms of the total four. The motion of the quadrotor captured by the Vicon system is exactly the motion of this body frame.

IMU frame: The IMU frame denoted as I is the frame in which the inertial sensors output their readings. It is attached to the IMU sets and has the same orientation as the body frame. In all the simulations in this thesis, it is an assumption that the body frame and the IMU frame coincide with each other by placing the IMU at the COM of the vehicle. In practice, because the IMU is not perfectly located in the COM of the quadrotor, there exists a translational displacement between the origins of the two frames.

Camera frame: The camera frame C with a basis of $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$ is defined and rigidly attached to the camera. The origin of the camera frame is located at the optical center of the camera. The base $\mathbf{c}_1$ and $\mathbf{c}_2$ are pointing to the right and down and are parallel to the image plane of the camera. $\mathbf{c}_3$ is along the optical axis of the camera and points out of the camera lens.

2.2 Coordinate Transformations

In this thesis, vectors located in one coordinate frame often need to be expressed in another frame. This kind of relationship is called the coordinate transformation. This transformation is represented by a rotation matrix $R \in SO(3)$ and a translation vector $t \in \mathbb{R}^3$. From the theory of computer vision, this transformation can also be interpreted as the motion between origins of two coordinate frames.

The rotation matrix R is also called the direction cosine matrix (DCM) because R consists the of inner product (or cosines) of each basis vector in one frame and each basis vector in another frame. In the navigation problem, the orientation from the body frame is described with respect to the navigation frame. Therefore, if there is no translation displacement between B and N, a vector expressed in B, and denoted as p_B and the same vector expressed in N, and denoted as p_N, have the following relationship from a rotational point

of view

$$p_N = R_N^B p_B \tag{2.1}$$

The rotation matrix R belongs to the special orthogonal group $SO(3)$ and thus has the following properties:

- $RR^T = R^T R = I$
- $\|R\| = 1$, $\det(R) = 1$

Transforming vectors from one coordinate frame to another requires the coordinate transformation between vectors. Take a point vector P, for example. As seen in Fig. 2.1, the point $P \in \mathbb{R}^3$ is initially observed in frame C and denoted as p_c. The transformation between B and C is $[R_B^C \; t_B]$ where R_B^C stands for the rotation matrix from C to B and t_B denotes the translation vec

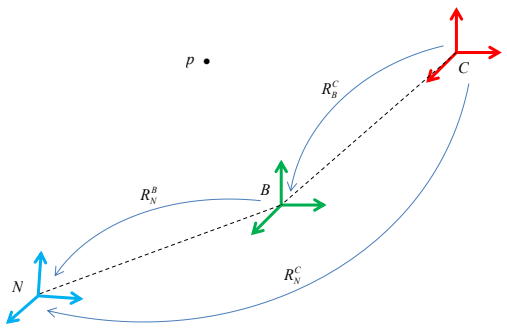


Figure 2.1: Coordinates transformation between different frames

The point can be expressed in the frame B, namely p_B, in the following form

$$p_B = R_B^C(p_C - t_B) \tag{2.2}$$

If we denote the origin of C in the frame B as t_C, then we obtain the following alternative formula between p_B and p_C.

$$p_B = R_B^C p_C + t_C \tag{2.3}$$

The orientation from the body frame with respect to the navigation frame, namely R_N^B, is the orientation state that will be estimated in the navigation system. The parameterization of the rotation matrix R_N^B uses the Euler angles. There are 12 possible Euler angle parameterizations of the rotation matrix such as rotation axis order $x \to y \to z$ and $y \to z \to x$. However, in this thesis, the conventional roll-pitch-yaw sequence, i.e., $z \to y \to x$, is used to form the rotation matrix. Specifically, the rotation matrix R_N^B is decomposed as a product of three elemental rotation matrices, i.e.,

$$R_N^B = R_z(\psi)R_y(\theta)R_x(\phi) \tag{2.4}$$

where the Euler angle roll ϕ, pitch θ and yaw ψ are defined as rotations around the x, y and z axes, respectively. These elemental rotation matrices are specified as

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.5a}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.5b}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5c}$$

Substituting (2.5) into (2.4) and after matrix multiplication, the rotation matrix R_N^B can be expressed as

$$R_N^B = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.6}$$

where $c_\phi \equiv \cos\phi$, $s_\phi \equiv \sin\phi$, etc. A special approximation of this matrix occurs when the Euler angles are small so that the cosine of an angle is approximated by one and the sine of an angle is approximated by the angle itself. Thus, we have the following expression of R

$$R \approx \begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} = I_{3\times 3} + [\alpha]_\times \tag{2.7}$$

where $\alpha \equiv [\phi\ \theta\ \psi]^T$, $I_{3\times3}$ is an identity matrix and $[\cdot]_\times$ is the skew-symmetric matrix such that $\mathbf{a}\times\mathbf{b} = [\mathbf{a}]_\times\mathbf{b}$ where $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$. The expression of this matrix is given as

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.8}$$

And it is straightforward that $[\mathbf{a}]_\times^T = -[\mathbf{a}]_\times$. Getting the Euler angles from the rotation matrix is an inverse problem of parameterization of the R matrix. From (2.6), the pitch angle is given by $\theta = -\arcsin r_{31}$. The range of this angle is limited by $-\pi/2 < \theta < \pi/2$. The other two angles can then be expressed as

$$\tan\phi = \frac{r_{32}}{r_{33}} \implies \phi = \arctan 2(r_{32}, r_{33}) \tag{2.9}$$

$$\tan\psi = \frac{r_{21}}{r_{11}} \implies \psi = \arctan 2(r_{21}, r_{11}) \tag{2.10}$$

The range of the angle ϕ and ψ is limited in $[-\pi, \pi]$. If θ is set to the value of $\pm\pi/2$, the rotation matrix would have a singularity problem: different Euler angle sequences would correspond to the same rotation matrix.

2.3 Rotation Kinematics

The kinematics of the rotation matrix are equally important to the parameterization of the rotation matrix. The kinematics constitute a major part of the navigation system dynamics. First, consider the case of a point vector p_B rigidly attached to the body frame. The point vector shares its origin with the static navigation frame. The coordinate of the vector expressed in the navigation frame p_N is given by (2.1). Here, we omit the superscript and the subscript for the rotation matrix R_N^B and denote it directly as $R_N^B \Leftrightarrow R$ for simplicity. Thus, differentiating the vector with respect to time gives

$$\dot{p}_N = \dot{R}p_B \tag{2.11}$$

where $\dot{p}_N \equiv v_N$ defines the change rate of the vector p_B in the navigation frame N. This represents the absolute velocity of a point vector p_B because all three of the components are measured with respect to a stationary origin.

The expression of $\dot{R}$ can be obtained from differentiating the first property of the rotation matrix R, i.e., $RR^T = I$. So we get

$$\begin{aligned} \dot{R}R^T + R\dot{R}^T &= 0 \\ \implies \dot{R}R^T &= -R\dot{R}^T = -(\dot{R}R^T)^T \end{aligned}$$

which means the matrix term $\dot{R}R^T$ is skew-symmetric. The last term $-(\dot{R}R^T)^T$ is parameterized using $[\omega]_\times$, where $\omega \in \mathbb{R}^3$. So we have

$$\dot{R} = [\omega]_\times R \tag{2.12}$$

Therefore, we can derive that

$$v_N = \dot{R}p_B = [\omega]_\times R p_B = [\omega]_\times p_N = \omega \times p_N$$

The last term relates the velocity vector v_N to the associated point vector p_N, which are rigidly attached to a body and are connected by a purely rotational motion. This allows us to see that the parameterization vector used, namely ω, has a physical meaning: it represents the body's angular velocity vector expressed in the navigation frame ω_N. The exact kinematics of the rotation matrix are given as

$$\dot{R} = [\omega_N]_\times R \tag{2.13}$$

The kinematic equation (2.13) has an alternative equivalence. Due to the property of R, $R^T \in SO(3)$, it is straightforward to obtain the relationship of $R^T[a]_\times R = [R^T a]_\times$. Thus, it can be derived that

$$\dot{R} = [\omega_N]_\times R = RR^T[\omega_N]_\times R = R[R^T\omega_N]_\times$$

That is

$$\dot{R} = R[\omega_B]_\times \tag{2.14}$$

where $\omega_B = R^T\omega_N$ is the angular velocity vector of the body expressed in the body frame B. The vector measurement of each component of ω_B can be retrieved at intervals using measurements from a triad of rate gyroscopes, which is part of the IMU.

2.4 Two-View Geometry

Human have two eyes to identify and locate the objects we see. In computer vision applications, we can recover the structure of the real world using two perspective images. The two images can be acquired from the two cameras in a stereo rig or two consecutive images from a monocular camera affixed to the rigid body. For a stereo rig, both of the cameras as well as the relative position between them need calibration whereas for the monocular setup, only the camera used needs calibration. In this thesis, we use the second method, minimizing the hardware requirement and also reducing the complication of calibration. The core theory in two-view geometry refers to the epipolar geometry, which provides the image coordinates' constraint from two image views. These constraints play a crucial role in recovering the motion of the camera.

2.4.1 Epipolar Geometry, the Essential Matrix and the Fundamental Matrix

Consider two consecutive image frame pairs, Π and Π', taken from a monocular camera. A 3D space point P has its images p and p' projected on Π and Π' with optical centers O and O', respectively. This creates a relationship between a certain point in the first image and the matched point shown in the second image. The plane these five points belong to is the epipolar plane defined by the span of PO and PO' shown in Fig. 2.2. O and O' constitute the baseline in this epipolar plane, which intersects with the two image planes Π and Π' at two points e and e'. These two points are called the epipoles of the two cameras. In particular, the epipole e' is the optical center of the first camera O projected on the second image Π', and likewise e is the optical center of the second camera O' projected on the first image Π. This epipolar constraint plays a fundamental role in motion reconstruction and pose estimation problems.

It should be noted that the most difficult part of virtually reconstructing the stereo vision system is matching the correspondences between the two images. Using the epipolar geometry constraint can restrict the range of searching for and matching such correspondences. To explain the epipolar constraint, it

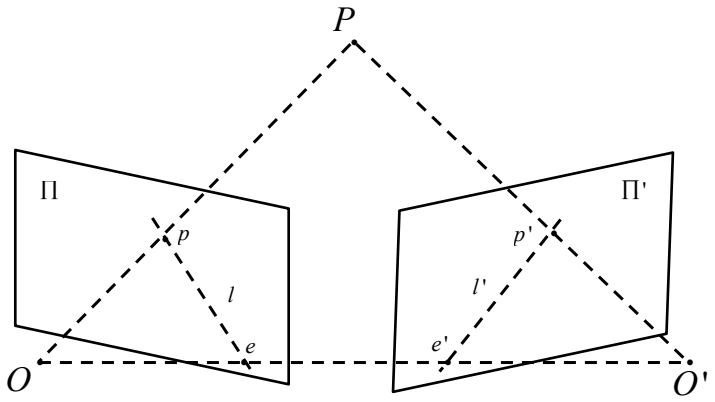


Figure 2.2: Epipolar geometry

is necessary to introduce the essential matrix E and the fundamental matrix F.

The essential matrix E connects the associated point coordinates in one equation. Applying the essential matrix requires knowledge of the camera's intrinsic parameters, namely the camera calibration matrix K. The point coordinates processed here are normalized image coordinates. The normalized image coordinate $\hat{p}$ is not the same as the image pixel pair $p = (u, v, 1)$, but in terms of its transformation, $\hat{p} = K^{-1}p$.

If R and t are denoted as the relative rotation and translation from the first camera frame to the second camera frame, the essential matrix corresponding to R and t is interpreted as

$$E = [t]_{\times}R = R[R^T t]_{\times} \tag{2.15}$$

Thus, the epipolar constraint is given as

$$\hat{p'}^T E\hat{p} = 0 \tag{2.16}$$

The above equation (2.16) defines the essential matrix, meaning that given a couple of correct correspondences, the essential matrix can be calculated from a set of equations. The number of such equations is determined by the degree of freedom of the essential matrix. It is straightforward to show that

the essential matrix has five degrees of freedom. We know either the rotation matrix R or the translation t has three degrees of freedom. Excluding an overall scale ambiguity and from (2.15), the exact degree of freedom of the essential matrix is five.

A critical property of the essential matrix is that two of its singular values are equal and the other one is zero [40]. This means that the essential matrix can be expressed by singular value decomposition (SVD) as $E = U\,\mathrm{diag}(1, 1, 0)V^T$. Once we obtain the essential matrix from the point correspondences, the motion between the two frames is extracted. That is to say, the relative pose between the two camera frames is therefore solvable.

The reconstructed motion of R and t between two frames is up to a scale and has four pairs of answers. The essential matrix has a property of $E^T t = 0$. Solving the null space of E can yield the recovered translation vector t. One of the solutions is obtained from the aforementioned SVD of E and is denoted as u_3, the third column of U and also a unit vector. Thus, t is given as $t' = u_3$ or $t'' = -u_3$.

There are also two solutions to the recovered rotation matrix R, namely

$$R' = UWV^T \text{ and } R'' = UW^TV^T$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So the four pairs of solutions of R and t are (R', t'), (R', t''), (R'', t') and (R'', t''). The four solutions are illustrated in Fig. 2.3. In order to get the physically feasible solution, we need to reconstruct the point in C which is in front of both cameras.

As the essential matrix is up to either a positive scale or a negative scale, we have four possible solutions of the rotation matrix R and the translation vector t. One improved method of extracting R and t from E is the Cayley transformation [44, 45], which can reduce the range of obtaining the physically feasible solution. This method can necessarily generate two possible solutions of R and t rather than four. If the two solutions are generated, we do not

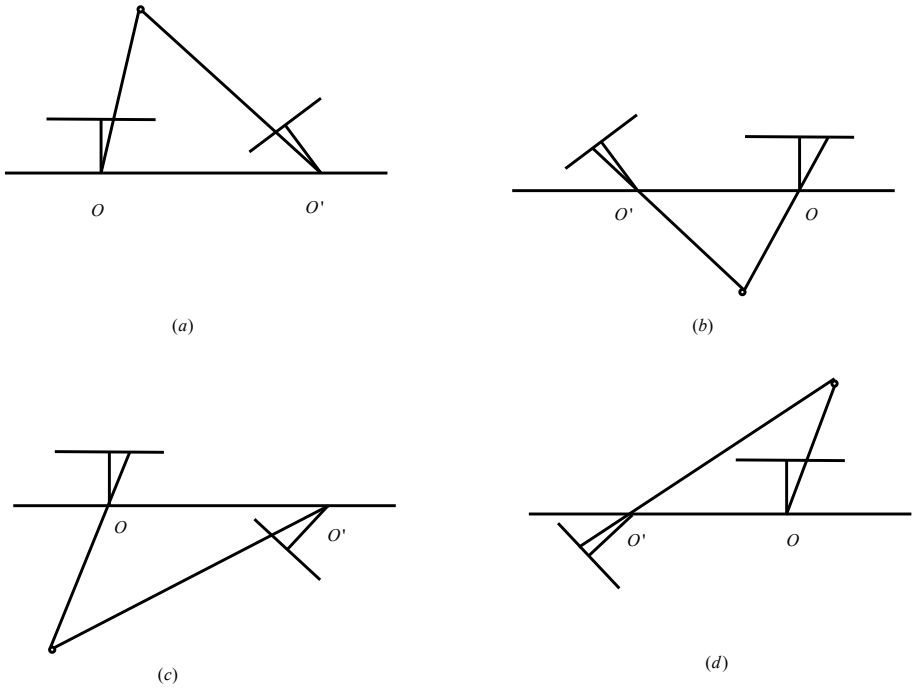


Figure 2.3: The reconstructed space point from the four possible solutions of R and t

need to take (at most) four 3D space point reconstruction steps to obtain the physically correct solution.

The first step in using the Cayley transformation method is to use the property of the essential matrix, i.e., $E^T t = 0$. By solving this equation we can obtain the basis of the null space of the essential matrix E. Denote this basis as $\tilde{t}$ and we then have that $t = s\tilde{t}$ where s is a non-zero scalar. Let $\breve{t}$ be the orthogonal vector of $\tilde{t}$. It is easy to see that $\breve{t}^T E = s\breve{t}^T[\tilde{t}]_\times R \neq 0$. The scalar s is expressed as

$$s = \pm \frac{\parallel \breve{t}^T E \parallel}{\parallel \breve{t}^T [\tilde{t}]_\times \parallel} \triangleq \pm s^* \tag{2.17}$$

Using the Cayley transformation, we denote $t = s^*\tilde{t}$ and $R = \frac{I - h_\times}{I + h_\times}$. Substituting these two transformation into (2.15) can yield

$$E(I + h_\times) = [s^*\tilde{t}]_\times (I - h_\times) \tag{2.18}$$

Reforming this equation gives

$$h_{\times}(\underbrace{E^T - [s^*\tilde{t}]_{\times}}_{[a_1\ a_2\ a_3]}) = (\underbrace{E^T + [s^*\tilde{t}]_{\times}}_{[b_1\ b_2\ b_3]}) \tag{2.19}$$

which can be equivalently rewritten as

$$[a_i]_{\times}h = -b_i \tag{2.20}$$

The least square solution of the above equation is

$$h_{(1)} = -(\sum_{i=1}^{3}[a_i]_{\times}^T[a_i]_{\times})^{-1}\sum_{i=1}^{3}[a_i]_{\times}^T b_i \tag{2.21}$$

One of the solution pairs for R and t is

$$R = \frac{I - [h_{(1)}]_{\times}}{I + [h_{(1)}]_{\times}},\ t = s^*\tilde{t} \tag{2.22}$$

Another solution pair is gained by using $t = -s^*\tilde{t}$

$$R = \frac{I - [h_{(2)}]_{\times}}{I + [h_{(2)}]_{\times}},\ t = -s^*\tilde{t} \tag{2.23}$$

where

$$h_{(2)} = -(\sum_{i=1}^{3}[b_i]_{\times}^T[b_i]_{\times})^{-1}\sum_{i=1}^{3}[b_i]_{\times}^T a_i \tag{2.24}$$

After introducing the essential matrix, it is necessary to interpret the fundamental matrix. The epipolar constraint (2.16) works with the normalized image coordinates. If the camera calibration matrix is known in advance in our monocular vision configuration, we can write the image coordinate as $p = K\hat{p}$ and $p' = K\hat{p'}$. The epipolar constraint is accordingly expressed as

$$p'^T F p = 0 \tag{2.25}$$

where the matrix $F = K^{-T}EK^{-1}$ is referred to as the fundamental matrix. As with the essential matrix, the fundamental matrix has a rank of two. We often use the singularity property of the fundamental matrix to assist with the estimation of it, using $n \geq 8$ point correspondences. In the next section the eight-point algorithm that we use is provided.

2.4.2 The Eight-Point Algorithm

The eight-point algorithm can calculate the fundamental matrix directly from image point correspondences and does not require other information such as camera calibration. This algorithm is one of the most important numerical algorithms in the theory of epipolar geometry. The basic principle for estimating the fundamental matrix is the epipolar constraint (2.25). Accumulating sufficient points can allow us to compute the unknown fundamental matrix F. To form enough linear equations with all the entries of the fundamental matrix F, $n \geq 8$ point correspondences are needed from the two image frames. The constraint (2.25) can be expanded into

$$u'uf_{11} + u'vf_{12} + u'f_{13} + v'uf_{21} + v'vf_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0 \quad (2.26)$$

where $p = (u, v, 1)^T$ and $p' = (u', v', 1)^T$. Denote

$$\mathbf{f} = [\ f_{11} \quad f_{12} \quad f_{13} \quad f_{21} \quad f_{22} \quad f_{23} \quad f_{31} \quad f_{32} \quad f_{33}\ ]$$

and (2.26) is reformed as

$$[\ u'u \quad u'v \quad u' \quad v'u \quad v'v \quad v' \quad u \quad v \quad 1\ ]\mathbf{f} = 0$$

For a set of n correspondences, the resulting equations are

$$A\mathbf{f} = \begin{bmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{bmatrix} \mathbf{f} = \mathbf{0} \qquad (2.27)$$

If only 8 points are used for minimizing $\|A\mathbf{f}\|$ and A has a exact rank of eight, the solution to (2.27) is unique. If $n \geq 9$ and A has rank of 9 due to coordinate noise, a least-squares minimization under the constraint $\|\mathbf{f}\|^2 = 1$ can give a feasible solution. It should be noted that the pixel coordinates of the corresponding points can have a wide range, which may lead to numerical instabilities. For example, in a 640×480 image, one feature point may have a pixel pair of (0.5, 0.5) whereas another feature point may have a pixel pair of (300,200) in a much larger numerical scale. Thus, in order to improve the stability of the result, we need to normalize the data before constructing

the equation (2.27) we are trying to solve. This normalization starts with the transformation of the image coordinates, i.e., performing $\tilde{p}_i = Tp_i$ and $\tilde{p}'_i = Tp'_i$ where T and T' are transformations for the normalization of the input image coordinates. To give the exact interpretation of T and T', we first need to define the centroid of the input image coordinates from both of the images. These coordinates are $\bar{p} = \frac{1}{n_p}\sum_{i=1}^{n_p} p_i$ and $\bar{p}' = \frac{1}{n_p}\sum_{i=1}^{n_p} p'_i$ where the vector denotations of them are $\bar{p} = [\ \bar{p}_1 \quad \bar{p}_2 \quad 1\ ]^T$ and $\bar{p}' = [\ \bar{p}'_1 \quad \bar{p}'_2 \quad 1\ ]^T$, respectively. The transformation is performed to achieve two goals:

- The centroid of the reference point coordinates is located at the origin after transformation.
- The RMS distance of the reference point coordinates to the origin is equal to $\sqrt{2}$.

Denote the scale factor σ and σ' as

$$\sigma = \left(\frac{1}{2n_p} \sum_{i=1}^{n_p} \|p_i - \bar{p}\|^2 \right)^{\frac{1}{2}}$$

$$\sigma' = \left(\frac{1}{2n_p} \sum_{i=1}^{n_p} \|p'_i - \bar{p}'\|^2 \right)^{\frac{1}{2}}$$

With these definitions according to the above criteria, the transformation T and T' are given as

$$T = \left[\begin{pmatrix} \sigma^{-1} & 0 & -\sigma^{-1}\bar{p}_1 \\ 0 & \sigma^{-1} & -\sigma^{-1}\bar{p}_2 \\ 0 & 0 & 1 \end{pmatrix} \right] \tag{2.28}$$

$$T = \left[\begin{pmatrix} \sigma'^{-1} & 0 & -\sigma'^{-1}\bar{p}'_1 \\ 0 & \sigma'^{-1} & -\sigma'^{-1}\bar{p}'_2 \\ 0 & 0 & 1 \end{pmatrix} \right] \tag{2.29}$$

After we obtain the fundamental matrix $\tilde{F}$ from the optimization process (2.27), we perform the singular value decomposition of it, which is

$$\tilde{F} = U\,\mathrm{diag}(r, s, t)V^T$$

We then set $\bar{F} = U\,\mathrm{diag}(r, s, 0)V^T$ and recover the fundamental matrix as $F = T'^T \bar{F} T$.

The eight-point method is typically used to extract of R and t from the essential matrix E based on the fact that $E = K^T F K$. This means that after we obtain the estimation of the fundamental matrix, we can recover the essential matrix. Because this algorithm directly calculates the fundamental matrix from pixel coordinates, it is often regarded as the very first step in processing information from images.

2.5 Sliding Window Bundle Adjustment

We obtain the current pose measurements R and t by frame-to-frame motion estimation incrementally. The absolute pose of the vehicle is accumulated from these pose measurements. Meanwhile, errors can be accumulated and thus lead to a significant drift to the vehicle absolute states estimation. It is never possible to remove the drift from accumulated pose measurements in practice, but we want to minimize the drift as much as possible. This is done in a local optimization over the last m visual pose measurements. We take $m = 5$ for the trade-off between the computation complexity and most optimized results. This process is called windowed bundle adjustment. Besides the optimization of the pose, the windowed bundle adjustment can also optimize the reconstructed 3D feature point locations which is useful in map-building in SLAM. The error function for the windowed bundle adjustment is to minimize the image reprojection error

$$\arg\min_{X_i, C_k} \sum_{i,k} \|p_{i,k} - g(X_i, C_k)\|^2 \tag{2.30}$$

where $p_{i,k}$ is the ith image point of the 3D space features X_i measured in the kth key frame image. The function $g(X_i, C_k)$ gives the image reprojection from the estimated 3D space point coordinates X_i based on the current camera pose C_k.

When the 3D reconstruction is absent from the visual navigation design, it is more efficient to do the optimization for the relative pose measurement only. We denote the pose measurement pair (R, t) at the current time as T_k. T_k is also the transformation between the previous vehicle location and the current

one. This optimization process involves pose measurements from non-adjacent key frames. The cost function to be minimized is given as

$$\sum_{i,j} \|C_i - T_{ij} C_j\|^2 \tag{2.31}$$

where T_{ij} is the transformation between ith pose and jth pose. The nonlinear optimization algorithm of Levenberg-Marquardt is used to obtain a solution.

Chapter 3

Sensors

This chapter elaborates on the modeling of the sensors as well as the related issues such as camera calibration, vision algorithms and so on. Visual algorithms in regards to detection and matching of feature points are provided and demonstrated.

3.1 Modeling the Inertial Sensors

The inertial sensor assumed used is the inertial measurement unit (IMU). An IMU typically comprises three orthogonal accelerometers measuring the acceleration of the vehicle, and three orthogonal gyroscopes measuring the rate of change of the vehicle's orientation. When dealing with the inertial system block of the visual-inertial navigation system, we perform a double integration of the accelerometer readings to estimate the position. The readings of the angular velocity from the rate gyros are integrated to obtain the orientation of the vehicle. The magnetometer is normally integrated as part of the IMU system. However, due to the irregularity in an indoor environment, the magnetometer could not perform well as a bearing sensor for the indoor vehicle. Thus, it could not be used in the design of the VI navigation system in this thesis.

The modeling of the IMU is based on the hardware in the Applied Nonlinear Control Lab (ANCL). In particular, the inertial sensor hardware used is the PX4FMU autopilot platform and the PX4IO interface board. This flight management unit is one of the latest low-cost IMUs, which has a high per-

formance suitable for a variety of unmanned aerial vehicles (UAVs) such as quadrotors and helicopters.

3.1.1 Electronics on PX4FMU

The PX4FMU autopilot is a platform with open-source codes. It uses the Berkeley Software Distribution (BSD)-licensed NuttX operating system for command input and output and connection with peripherals. This flight management unit is used as an inertial measurement unit and enables the control of an unmanned aircraft using a single-board solution. The PX4FMU and PX4IO can interface different buses such as universal asynchronous receiver/transmitter (UART) and I^2C. The PX4IO can output pulse width modulation (PWM) signals to control unmanned vehicles.

The PX4FMU autopilot platform has a processing capability of 168 MHz and a microSD slot which allows us to fully store the flight data onboard. The sensor data stored can be transferred to a desktop computer for offline processing. The platform in use is version 1.7 and it is equipped with a micro USB port on one edge of the platform board, which is shown in Fig. 3.1. This enables communications between the autopilot and the ground station via either a USB cable or, wirelessly, using radio modules. It means we can inspect the sensor readings on the screen in real time. Also, the processing speed of the autopilot platform allows us to set the gyroscope sampling rate to a sufficient high value, for example, up to 200 Hz.

The three axes accelerometers and three axes gyros are integrated as a block on the platform MPU-6000. The block can output motion data in either quaternion or Euler angles. When using the gyroscope and the accelerometer together, the operating current flows at 3.8 mA. There is another gyroscope L3GD20 backed up at the PX4FMU platform. This gyroscope provides redundancy for an alternative choice. The sensor startup defaults to the MPU-6000, but users can make modifications such that L3GD20 can publish its topic, which can be subscribed to publicly. The independent gyroscope has a sensing element and an IC interface capable of providing the measured angular rate through a digital interface. The IC interface is produced through a CMOS

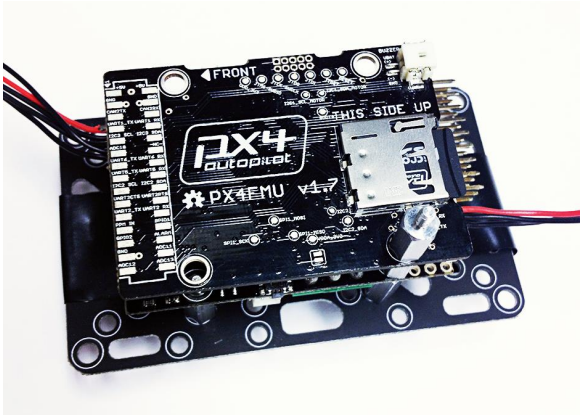


Figure 3.1: The PX4FMU autopilot platform

process that allows a high level of integration to design a dedicated circuit. This results in a better match of the sensing element characteristics.

The PX4FMU platform is equipped with a variety of libraries, drivers and controller executives. The built-in feature of the Object Request Broker (ORB) design pattern provides the framework to share data structures between threads and applications. The data transmission between the ground-station and the UAV quadrotor is achieved via the middleware of MAVLink. MAVLink enables customization of the packet structure, the sequencing, the identification and the check-sum.

To make the coupled set of the visual-inertial sensor combination, we can connect associated ports between the FX4FMU and camera sensor board. The port on the PX4FMU used for such a combination is involved with the multifunction connector. It has 15 pins on it with I^2C, UART and general-purpose input/output (GPIO) bus ports. These pins are always electrostatic discharge protected. Three pins are used to connect with the intelligent camera sensor board. The pins in use are pin 9 (USART2 RX), pin 12 (USART2 TX) and pin 15 (GND).

3.1.2 Modeling of the Inertial Sensors

The accelerometer of the MPU6000 measures the specific force of the vehicle $f_B = \ddot{p}_B - g_B$ in the body frame B and the gyroscope of the MPU6000

measures the angular velocity ω_B. We denote the measurements of these two sensor readings as y_f and y_ω where y indicates the measurement variable. The models for the inertial sensors, i.e., for the accelerometer and the gyroscope are assumed in terms of true values corrupted by the bias and noise based on work in [2].

$$\begin{aligned} y_f &= f + b_f + n_f \\ y_\omega &= \omega + b_\omega + n_\omega \end{aligned} \tag{3.1}$$

where n_f and n_ω are white Gaussian noise vectors with zero mean and the covariance matrix R. In the sensor models the additive terms b_f and b_ω are bias error vectors. The bias error was modeled as the first-order Guass-Markov process presented in [14]

$$\dot{b} = -\eta b + n_b \tag{3.2}$$

where n_b is a Gaussian white noise process. This model inherently has a stable dynamic term $\dot{b} = -\eta b$ which indicates constant biases corrupted by additive noise. In this thesis we model the biases as the random walk process $\dot{b} = n_b$ as in [2]. Fewer parameters are in the random walk process since we do not include the first order dynamic term for the biases. However, in the implementation simulations, we assumed the true biases are constant vectors. The bias models for the accelerometer and the rate gyroscope are respectively given as

$$\begin{aligned} \dot{b}_f &= n_{b_\omega} \\ \dot{b}_\omega &= n_{b_f} \end{aligned} \tag{3.3}$$

The specific force f vector is a special type of acceleration but not an actual force applied to the vehicle body. It belongs to the category of acceleration subject to the free-fall movement of the vehicle. It calculates the difference between the vehicle's inertial acceleration and the gravity, i.e., $f = a - g$. When the vehicle stands still, the accelerometer will give a measured value of $-g$. On the other hand, if the vehicle falls freely, the accelerometer will indicate the value of 0.

3.2 Modeling of the Camera

The 3D world is mapped to an image through a visual sensor such as a camera. The intelligent visual sensor proposed and used in this thesis is the Raspberry Pi camera platform (model B), shown in Fig. 3.2. The Raspberry Pi is a low-

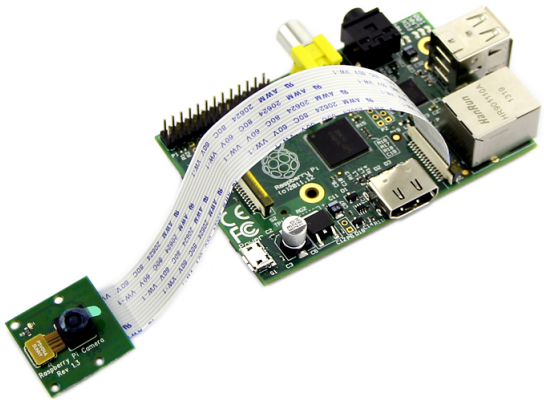


Figure 3.2: The Raspberry Pi camera platfrom

cost ARM-based computer with a camera module. This module is connected to the Raspberry Pi board through a flexible flat cable that can be plugged into the camera serial interface (CSI) connector. The platform supports languages such as Python, C, C++ and Java. In addition, the platform can connect to a network from peripheral of a Wi-Fi adapter transmitting the data wirelessly. The camera module can be easily configured in the Debian-like system, the Raspbian. This camera platform is modeled as the perspective camera model,

Parameter	Values
Maximum Resolution	2592×1944 pixels
Pixel Size	1.4×1.4 μm
Focal Length	3.6 mm
Angle of View	54×41 degrees

Table 3.1: The values of the parameters used in the model.

which is modeled with pixels in its model expression. Other types of cameras commonly used in the visual-inertial navigation design include the catadiop-

tric camera and the omni-directional camera. All these camera models are generalized from the basic pin hole camera model.

The pinhole camera model assumes that the projection center is located at the origin of the camera frame C. Along the projective orientation, the plane $z_c = f$, is the defined image plane where f is the focal length of the camera. In this definition a space point in the camera frame $P = [X, Y, Z]^T$ is mapped to the 2D point $p = [fX/Z, fY/Z]^T$ in the image plane, which is shown in Fig. 3.3. The projection center, namely the origin of the camera, is defined as the camera center or the optical center. The ray from the camera center into the image plane and also perpendicular to it is the principal axis. The cross point of the principal axis and the image plane is the principal point. In order to generalize the pinhole camera model to the perspective camera

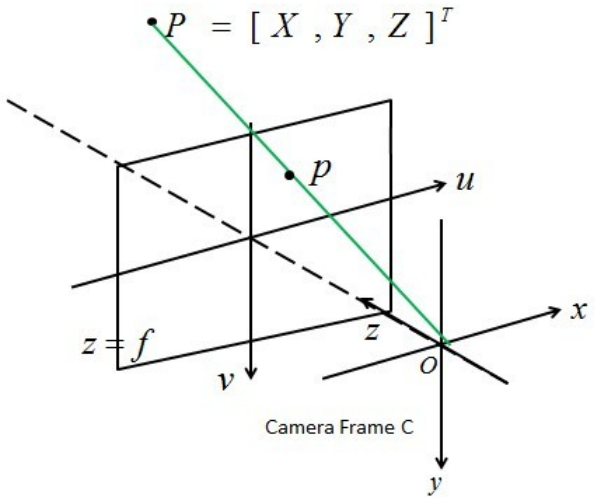


Figure 3.3: The perspective camera

model, we need to use the homogeneous coordinate system. The homogeneous coordinate system has a property that a coordinate vector $(x_1, \ldots, x_n)$ can still represent the same vector after multiplying it by a non-zero scalar. This means that the coordinate vectors $(x_1, \ldots, x_n)$ and $(cx_1, \ldots, cx_n)$ are regarded as the same vectors where c is a non-zero constant scaler.

Thus, the center projection of a 3D point to a 2D image point is represented

in homogeneous coordinates as

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Here the projection matrix can be written as $\text{diag}(f, f, 1)[I|\mathbf{0}]$. However, in practice, the principal point may not fall at the center of an image. A more common setup is that the principal point is located at one corner of the image. In this case, we have to compensate for the principal point offset. We denote the coordinates of the principal point as (o_x, o_y) in pixels. Thus, the 3D to 2D projective mapping in homogeneous coordinates can be expressed as

$$\begin{bmatrix} fX + Zo_x \\ fY + Zo_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & o_x & 0 \\ 0 & f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.5}$$

where the pixels are assumed to be squared. The associated camera projective matrix can be written as $K[I|\mathbf{0}]$ where

$$K = \begin{bmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix K is called the camera calibration matrix. Note that this camera calibration matrix does not contain the pixel size information. The parameters contained in the camera calibration matrix K are called the internal or intrinsic camera parameters. In contrast, those parameters that describe the camera location in the 3D world and the camera-to-IMU calibration are called the external camera parameters.

The sensor board of the Raspberry Pi camera module is made of a complementary metal-oxide-semiconductor (CMOS) camera. From practical consideration, we need to consider the case where the image pixels are non-square. Thus, there will be new parameters in the camera calibration matrix K, which represent the scales in both of the camera image directions. If we denote the m_x and m_y as the number of pixels in each unit distance in the image coordinates for horizontal and vertical directions, the generalized form of the camera

calibration matrix is given as

$$K = \begin{bmatrix} fm_x & 0 & o_x \\ 0 & fm_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

where fm_x and fm_y represent the focal length of the camera in terms of the pixel dimensions in the horizontal and vertical image directions.

3.2.1 Non-Perspective Camera Models

In this section we introduce two non-perspective camera models for cameras with different types of lenses because they are commonly seen in the visual-inertial navigation system design. Those two camera models reviewed in this thesis include the fisheye lens camera and the catadioptric camera. The motivation for using such camera lenses is to break through the limited field of view of the perspective camera. There is an inevitable issue when using a perspective camera that the landmarks or the features may exceed the boarder of the current image view.

The fisheye lens significantly extends the camera's filed of view. Although this lens can cause severe image distortion, we can still retrieve valuable visual measurements by correctly modeling it. One typical feature of the fisheye lens camera is that the shape of the image view outlines a circle image plot instead of a square plot. This is illustrated in Fig. 3.4. The shadow in the image plane is the actual image plot we obtain from the stored image data without any filter applied to it. In Fig. 3.4, O is the camera center and the origin of the camera frame. The z-axis is the optical axis pointing out of the camera. A world feature point $P = [X,\ Y,\ Z]^T$ is parameterized with the spherical coordinates $(R,\ \theta,\ \phi)$ in the camera frame where scalar parameters R, θ and ϕ are given as

$$R = \sqrt{X^2 + Y^2 + Z^2},\ \theta = \cos^{-1}\frac{R}{Z},\ \phi = \tan^{-1}\frac{Y}{X} \tag{3.6}$$

The projection of P on the image plane is denoted as p and expressed in the polar coordinates $(r,\ \phi)$. Therefore the Cartesian image plane coordinates $(u,\ v)$ of a point p are expressed as

$$u = r\cos\phi,\ v = r\sin\phi \tag{3.7}$$

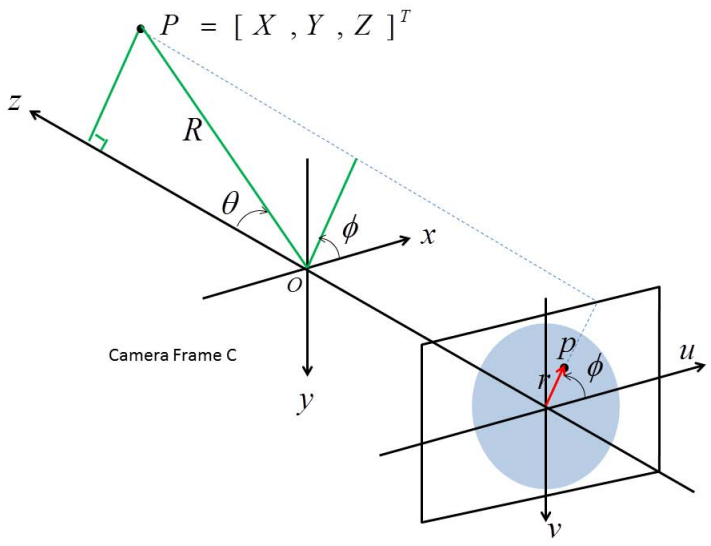


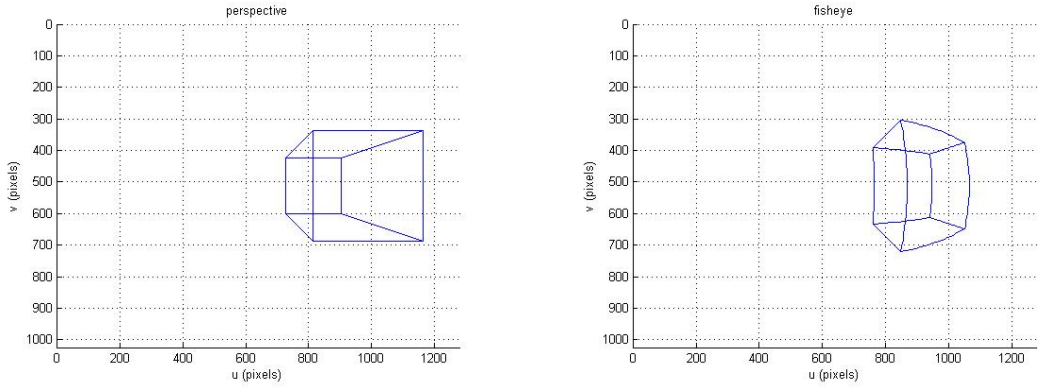
Figure 3.4: Geometric structure of the fisheye camera

Mapping functions	Equations
Equiangular	$r = k\theta$
Stereographic	$r = k\tan(\theta/2)$
Equisolid	$r = k\sin(\theta/2)$
Sine	$r = k\sin(\theta)$

Table 3.2: Parameters for different fisheye camera lenses

The radius r is a function of the angle θ and depends on the type of the fisheye lens. This function $r = r(\theta)$ has the following expression options in Table 3.2.

The following figure, Fig. 3.5, gives an intuitive illustration of how a cube appears on the image captured by a central perspective camera compared to how it looks captured by a camera with a fisheye lens. The other non-perspective camera that is widely used in the visual navigation system design is a catadioptric camera system. This optical system, which contains a mirror and a lens, can provide an omnidirectional view of the environment around the vehicle equipped with the camera. Depending on the mirror type, there are specific constraint equations for the catadioptric optical system such as the hyperboloidal mirror, the ellipsoidal mirror and the paraboloidal mirror



(a) The perspective camera model (b) The fisheye lens camera model

Figure 3.5: Comparison between the perspective camera model and the fisheye lens camera model

[24]. The most commonly used mirror is the hyperboloidal mirror. This type of camera system is illustrated in Fig. 3.6. As it is a 360° view camera, the view plot is a circle on the image plane. In Fig. 3.6, the orange curve is edge

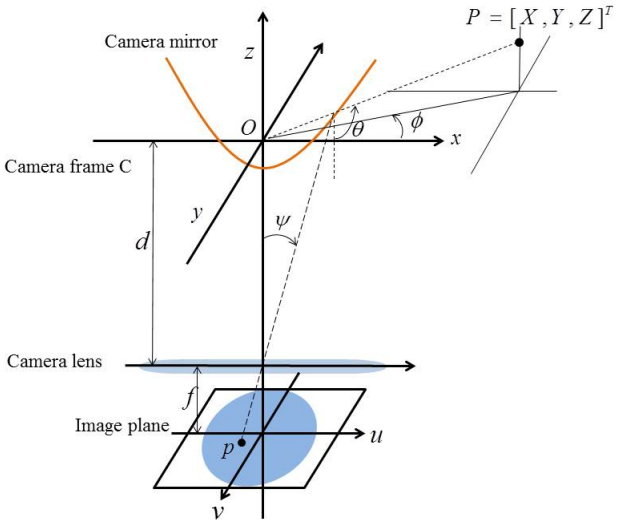


Figure 3.6: Geometric structure of the catadioptric camera with hyperboloidal mirror

of the cross section of the hyperboloidal mirror. The point O is the focal point of the hyperbola curve. O is the focal point of the camera mirror. Assume a 3D world point is located in the camera frame as $P = [X,\ Y,\ Z]^T$. A ray is

emitted from the point P to the focal point O and is reflected at the surface of the mirror. The ray goes through the center of the camera lens and forms an image point at the image plane. The ray has an elevation angle of

$$\theta = \tan^{-1} \frac{Z}{X^2 + Y^2} + \frac{\pi}{2} \tag{3.8}$$

The reflected ray emitted to the camera center has an angle of ψ with respect to the optical axis z. According to the tangent of the mirror surface, the expression of ψ has different terms such as the equianglar function of $\theta = k\psi$. The image point is represented in polar coordinates namely $p(r, \phi)$ where the radius is given as $r = f \tan \psi$. The image coordinates in Cartesian form are

$$u = f \tan \psi \cos \phi, \;\; v = f \tan \psi \sin \phi \tag{3.9}$$

The bearing angle of the point P is defined as $\phi = \tan^{-1} \frac{Y}{X}$.

Fig. 3.7 shows a cube subject projected on the image plane using the catadioptric camera system.

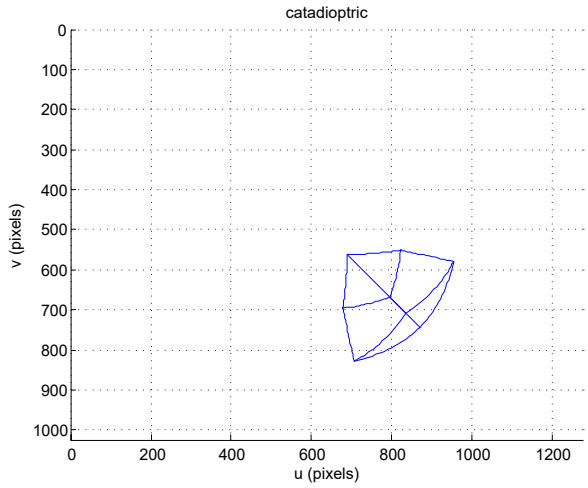


Figure 3.7: Image example of the catadioptric camera

Since the projected image view with the fisheye lens camera or the catadioptric camera are rounded, the actual image on the screen plot has a black background around the image view plot. As a result, some image pixels are wasted. For the catadioptric camera system, there is a black circle in the image center which is the projection of the mirror itself.

3.2.2 Camera Calibration

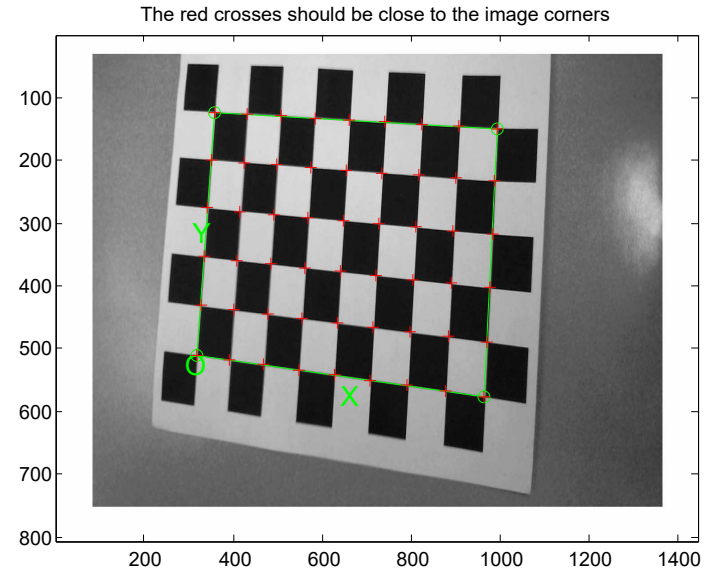
The camera's intrinsic parameters in K are calibrated and assumed known before designing the VI navigation system presented in Chapter 4. These parameters can be formed as augmented parameters of the states and thus are estimated during the state estimation process in the navigation tasks such as in the work presented in [33]. However, methods like this still require some prior knowledge of the range and the initial guess of the camera parameters. So it is worth doing a calibration before the navigation task. Moreover, for certain cameras, the calibration job needs to be done only once, as these camera parameters are inherent and will not drift or change during the state estimation process. This can reduce the computation consumption onboard and improve the efficiency of the estimation process.

Softwares such as MATLAB or OpenCV provide special tools for the camera calibration task. The camera calibration toolbox [4] is one of the most widely used tools for camera calibration. When using the MATLAB toolbox, it is necessary to prepare a mosaic black-and-white chessboard pattern with each mosaic square a certain size. The calibration process typically requires a total of twenty image snapshots created by placing the camera at different angles facing to the chessboard. Fig. 3.8 shows the sample calibration images.

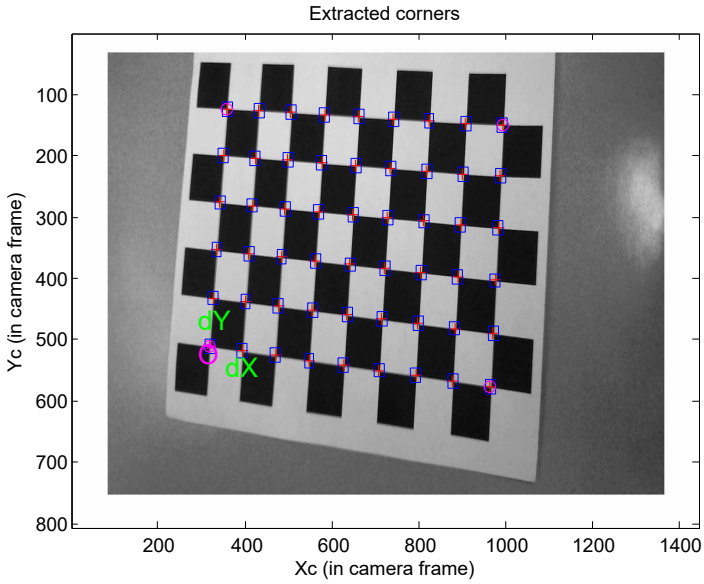
Apart from the camera's intrinsic parameter calibration, the camera's external parameter calibration is also noteworthy. In this thesis, the external parameter calibration is the camera-to-inertial calibration and we assume that this transformation is calibrated in advance and thus known. Consider works in [25, 31] for example. In [25], the camera-to-inertial calibration is identifiable under the position condition that the motion that the robot vehicle undergoes is in neither a zero acceleration range nor a constant rotation.

3.2.3 Combination of Sensors

Along the edge of the Raspberry Pi platform board, there are twenty-six pins, of which seventeen are general-purpose input/output (GPIO) pins and others are power or ground pins. The voltage for the power pin can be +3.3V or +5V.



(a) Sample of the corner detection in a calibration image



(b) Sample of the extracted corners in a calibration image

Figure 3.8: Two sample images used for camera calibration

Fig. 3.9 illustrates how visual and inertial sensors are integrated in the practical design. When the vehicle is maneuvering, the PX4FMU records the inertial

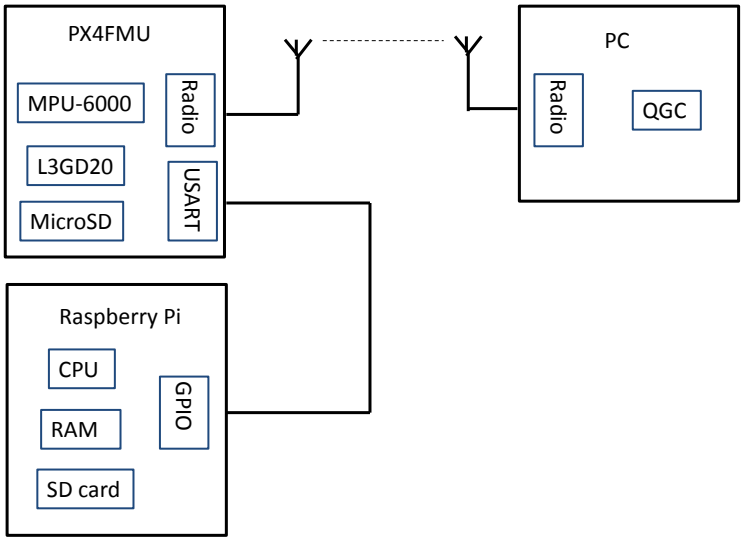


Figure 3.9: The schematic for a combination of visual and inertial sensors

sensor data to its MicroSD storage and the Raspberry Pi records the vision data to its local SD storage. The inertial data obtained by PX4FMU can be transmitted to the desktop computer and shown through the QGroundControl (QGC) software in real time.

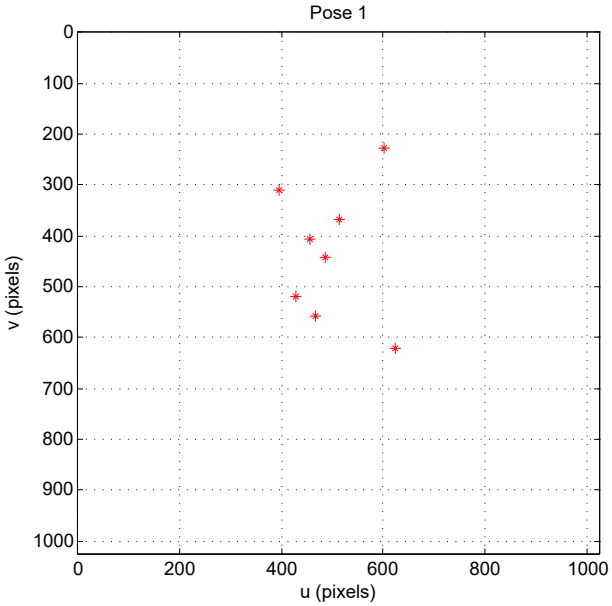
3.3 Visual Algorithms

In this section the visual algorithms related to the computer vision theory are provided and illustrated using the MATLAB RVC (Robotic Vision and Control) toolbox.

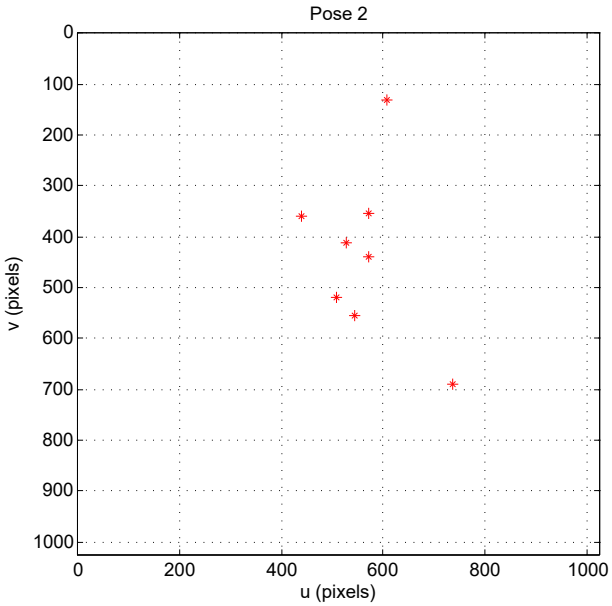
3.3.1 Pose Measurement from Vision Algorithms

Eight feature points are assumed with known 3D coordinates, located in the world frame N. The camera module is realized by a function of central perspective camera models. The intrinsic parameters are formed as inputs into the function with a user-defined camera name. We use the parameters from the

calibration of the Raspberry Pi camera module. Two adjacent image frames with eight points projected on the image is shown in Fig. 3.10.



(a) Image points from initial pose



(b) Image after camera pose transformation

Figure 3.10: Images from two camera poses

The epipolar geometry implementation is elaborated in Chapter 4. The actual transformation between two poses of the camera is set with a translation vector of $t^{\star} = [0.3821, 0, 0.1182]^T$ and a rotation with respect to the y-axis of

the camera frame with the rotation matrix of $R^\star = R_y(-0.6)$. The camera movement is shown in Fig. 3.11.

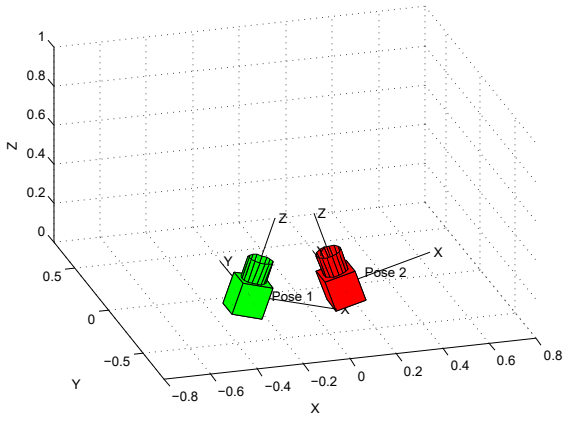


Figure 3.11: Camera movement between two poses

We collect the feature coordinate data from both images, form corresponding homogeneous coordinates and then store them into two matrices for fundamental matrix estimation. The exact transformation using the known parameters is shown numerically as follows

$$R^\star = \begin{bmatrix} 0.8253 & 0 & -0.5646 \\ 0 & 1.000 & 0 \\ 0.5646 & 0 & 0.8253 \end{bmatrix}, t^\star = \begin{bmatrix} 0.3821 \\ 0 \\ 0.1182 \end{bmatrix}$$

whereas the estimated pose pair using the eight-point algorithm presented in Chapter 2 is

$$\hat{R} = \begin{bmatrix} 0.8253 & 0.0000 & -0.5646 \\ -0.0000 & 1.0000 & 0.0000 \\ 0.5646 & -0.0000 & 0.8253 \end{bmatrix}, \hat{t} = \begin{bmatrix} 1.2724 \\ -0.0000 \\ 0.3936 \end{bmatrix}.$$

The estimated Euler angle in radian is $[0.0000, -0.6000, -0.0000]$, which is much closer to the true value. But the estimated translation vector is not close to the true value. As was explained in Chapter 2, the vision algorithms cannot recover the absolute scale. The estimated pose between two camera locations is up to a constant scale, i.e., $t^\star = 0.3 \cdot \hat{t}$. As a result, it becomes necessary for the filtering process to combine with the inertial sensor readings to obtain this absolute scale and estimate the quadrotor vehicle location in the real world coordinate frame N.

3.3.2 Optical Flow and Velocity Relationships

The motion of features in the image is defined as the term of optical flow (OF). The optical flow is a function of the camera's motion and also the 3D structure of the surrounding environment. The OF method is associated with image-based control and does not require the estimation of the vehicle states for robot control design.

In order to illustrate the OF method, first we assume that an image point of feature on the image plane is given as $p = [u,\ v]^T$. The motion of the camera is defined as both the linear velocity and the angular velocity in the camera frame $\nu = [v_C, \omega_C]^T$. Assume that the $3D$ point in the camera frame is given as $P = [X,\ Y,\ Z]^T$. The kinematics [5] of such a feature point are expressed as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{u}{Z} & uv & -(1+u^2) & v \\ 0 & -\frac{1}{Z} & \frac{v}{Z} & 1+v^2 & -uv & -u \end{bmatrix} \begin{bmatrix} v_{C,x} \\ v_{C,y} \\ v_{C,z} \\ \omega_{C,x} \\ \omega_{C,y} \\ \omega_{C,z} \end{bmatrix} \tag{3.10}$$

Equation (3.10) are the well-known kinematics for the classical image-based visual servoing (IBVS) method and it provides a relationship between the image feature velocity in the normalized image coordinates and the camera velocity. Equation (3.10) can be abbreviated as $\dot{p} = J\nu$ where the matrix J is defined as the image Jacobian and it can also be called as the interaction matrix L. In order to perform image based control design, the depth of a feature point must be known a priori, apart from its image coordinates.

Fig. 3.12 illustrates how the image feature can change in regards to the camera motion of a translational velocity v_C of $[0,\ 0,\ 3]^T$ and an angular velocity of $[0,\ 1,\ 0]^T$. It should be noted that the OF vision-based design is using the global feature, whereas the feature motion and feature dynamics used in IBVS (usually four points features) are in local feature forms. Although the OF method has a lower image feature motion accuracy, it does not have to deal with the issue of the singularity in the manipulator Jacobian and the local minima when doing the IBVS using the feature coordinates.

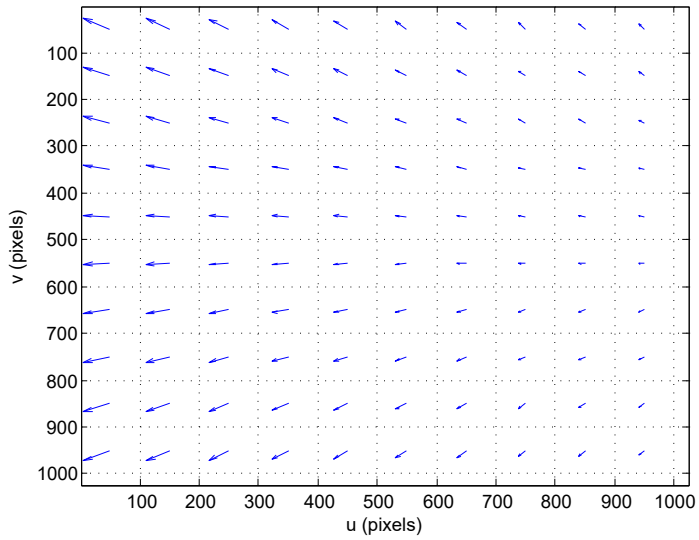


Figure 3.12: Optical flow: image plane velocities with camera motion

IBVS is the inverse manipulation problem of getting the image feature motions, which is expressed in (3.10). This kind of vision-based control is intended to minimize the errors between the desired image feature positions and the current image feature positions, i.e., $e(t) = s(t) - s_d$ where $s(t)$ is a function of a set of image feature coordinates and the camera parameters. Depending on the number of feature points, the robot vehicle can be under-actuated, fully-actuated or over-actuated. In order to fully actuate the robot to move in 6 DoF, at least three non-collinear feature points are required. As three points may lead to a square Jacobian and thus the singularity in Jacobian, we usually add one more redundant point and calculate the pseudo-inverse of the interaction Jacobian $J^{\dagger}$ in a visual servoing practice. This manipulation function is often defined as $\nu = J^{\dagger}\dot{p}$ and in error terms as $\nu = \lambda J^{\dagger}(p^{*} - p)$, where the pseudo-inverse is defined as $J^{\dagger} = (J^{T}J)^{-1}J^{T}$. The additional redundant point feature can attenuate the influence from the noise.

Chapter 4

EKF-Based Vision-Aided Navigation: Vehicle State Estimation

This chapter provides two results for vehicle state estimation based on the theory of the extended Kalman filter (EKF). One result demonstrates the map-based visual-inertial navigation, which assumes that the landmarks' 3D coordinates are known. The visual sensor, i.e., the camera, observes the coordinate pairs projected on the images and then those processed image coordinates are used as the visual measurements. This result does not have to estimate the absolute scale since the landmarks are known on a real-world scale. The other result is based on a computer vision algorithms and uses the scaled pose measurements after processing the image readings from the camera sensor. The scaled pose measurements are obtained independent of building a scaled map, which implies a different way of the simultaneous localization and mapping (SLAM) method and is computationally economical. The nonlinear observability analysis is also provided in this chapter.

In order to perform the control-related design for the unmanned aerial vehicles (UAVs) in the Applied Nonlinear Control Lab (ANCL), it is necessary to obtain accurate state variables for those tasks such as visual servoing and navigation. However, not all the state variables can be measured directly from onboard sensors. Even though a state variable can be measured through the related sensor, the result is normally affected by noise and is hardly reusable

unless more processing is done. The noise effect is largely due to sensors such as electronic instruments that are affected by electronic disturbance or biases, which exist in some inertial components. The Kalman filter has widely been proved to be effective in many areas, especially in navigation design. The Kalman filter can perform the task of an observer, namely estimation of the states that cannot be measured directly, and can also smooth the measurements and attenuate the noise effect on the measurements. One of the distinct advantages is that the Kalman filter gain can be derived rigourously using the Kalman gain formula.

The dynamics for UAVs such as quadrotors are nonlinear. Therefore, it is not possible to apply the Kalman filter directly to the quadrotor system as the Kalman filter handles linear systems only. We need to use the EKF, which is developed for nonlinear systems. The EKF method assumes the system dynamic is linearizable and the nonlinear system is linearized by the first order approximation in implementation.

4.1 Extended Kalman Filter Framework

The EKF adopts the idea that the nonlinear dynamic system is linearized around the state estimates generated from the EKF implementation and the state estimate is based on the linearized system. The proposed EKF design is based on work in [2]. The general form for the nonlinear system considered with noise input can be written as

$$\begin{aligned} \dot{x} &= f(x, u, w) \\ y &= h(x, v) \\ w &\sim (0, Q) \\ v &\sim (0, R) \end{aligned} \tag{4.1}$$

where x is the state, y is the output and u is the control input. The noise variable w represents the process noise and v represents the measurement noise. Both w and v are Gaussian white noise with zero means and known covariance. The nominal trajectory of the nonlinear system (4.1) can be written in the

noise-free form as

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}, u, 0) \\ \hat{y} &= h(\hat{x}, 0) \end{aligned} \tag{4.2}$$

where $\hat{y}$ is the estimated nominal output. Linearizing the nonlinear dynamic system (4.1) around the nominal system trajectory, we can get

$$\begin{aligned} \dot{x} &= f(\hat{x}, u, 0) + \left.\frac{\partial f}{\partial x}\right|_{\hat{x},u,0} (x - \hat{x}) + \left.\frac{\partial f}{\partial u}\right|_{\hat{x},u,0} (u - u) + \left.\frac{\partial f}{\partial w}\right|_{\hat{x},u,0} (w - 0) \\ y &= h(\hat{x}, 0) + \left.\frac{\partial h}{\partial x}\right|_{\hat{x},0} (x - \hat{x}) + \left.\frac{\partial h}{\partial v}\right|_{\hat{x},0} (v - 0) \end{aligned} \tag{4.3}$$

By defining the term of $\delta x = x - \hat{x}$, $\delta y = y - \hat{y}$ and the corresponding Jacobian matrices in the above linearized system such as $F = \left.\frac{\partial f}{\partial x}\right|_{\hat{x},u}$, $G = \left.\frac{\partial f}{\partial w}\right|_{\hat{x},u}$, $H = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}}$ and $L = \left.\frac{\partial h}{\partial v}\right|_{\hat{x}}$, we can get the linearized system of (4.1) in the following form (which will be used in this chapter later)

$$\begin{aligned} \delta\dot{x} &= F\delta x + Gw \\ \delta y &= H\delta x + Lv \end{aligned} \tag{4.4}$$

After that, the Kalman filter formula can be applied to (4.4).

The EKF method for non-vision navigation tasks uses aiding sensors such as the global positioning system (GPS) and the magnetometer to compensate for the slow drift of the inertial measurement unit (IMU). In this thesis, we eliminate the use of GPS and the magnetometer due to their ineffectiveness in the indoor environment. The visual sensor, namely the camera, is used alternatively as the aiding sensor for the inertial navigation system.

The vision-aided inertial navigation system in the nominal form (4.2) is first numerically integrated using the control input to produce the first stage state estimation $\hat{x}^-$ of the nonlinear system (4.1). When the aiding measurement is available, the nonlinear system is linearized at $\hat{x}^-$. Then, by using the Kalman filter formula, the state correction is rendered. The corrected state thus becomes $\hat{x}^+ = \hat{x}^- + \delta\hat{x}$ and is ready for use either for other control tasks or to propagate the $\hat{x}^-$ in the next EKF loop. If the aiding measurement is not available at the current loop, $\hat{x}^-$ is transmitted directly as the initial condition for the next integration of the nominal dynamics (4.2).

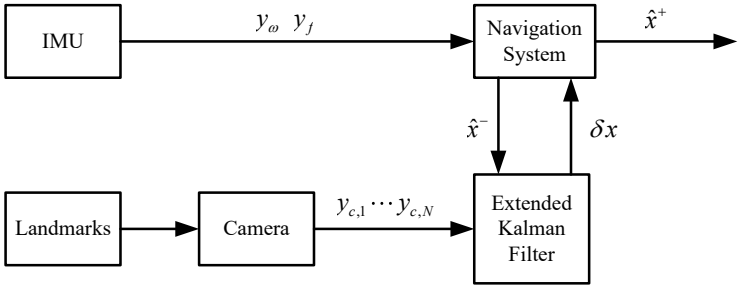


Figure 4.1: Block diagram of map-based visual-inertial navigation

The block diagram for the map-based visual-inertial navigation is shown in Fig. 4.1. The visual-inertial navigation system in this scheme gives the estimates of the vehicle, which include its positions, velocities and orientations with respect to a fixed world frame, namely the navigation frame N defined in Chapter 2.

The block diagram for the map-less visual-inertial navigation scheme is shown in Fig. 4.2.

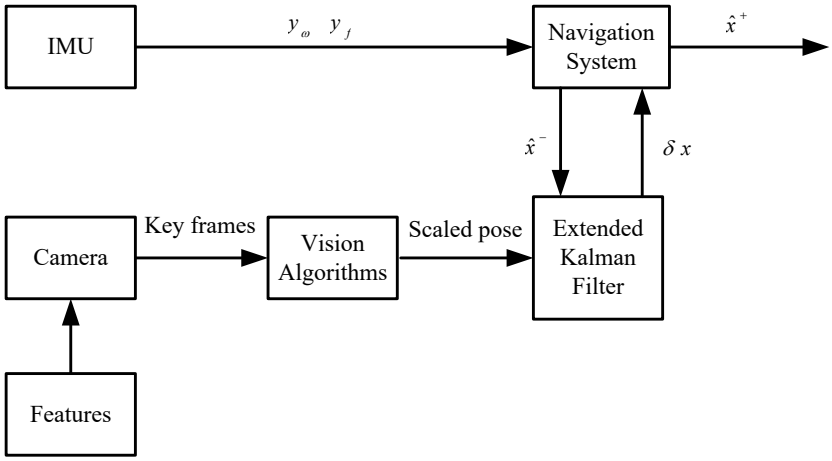


Figure 4.2: Block diagram of map-less visual-inertial navigation

The visual-inertial navigation system design in Fig. 4.2 renders the estimates of the vehicle states such as positions, velocities, orientations, as well as an absolute scale factor for the scaled pose generated from the visual algorithm. The position, velocity and orientation are in the absolute scale and are represented with respect to the navigation frame N.

4.2 Sensor Models

The sensors used in this thesis involve the IMU and the camera. The modeling of the sensors is covered in Chapter 3. As modeled in Chapter 3, the equation set (3.1) describes the inertial sensor model for the IMU, specifically both the gyroscope and accelerometer. The equation set (3.3) describes the bias model for those two sensors. The camera model assumed is the perspective model. The perspective model of central-projection is based on the pinhole model (3.4). For the visual measurement design, the measurement output equation is not dealing directly with the pixel. Particularly in the case of the map-based visual-inertial navigation, the projective equation is defined as

$$y_{C,i} = \pi(X_{C,i}) \tag{4.5}$$

where the projective function is $\pi = \{\mathbb{R}^3 \rightarrow \mathbb{R}^2 | X_{C,i} = [X\ Y\ Z]^T \rightarrow y_{C,i} = [X/Z\ Y/Z]^T\}$. $X_{C,i}$ in (4.5) is the absolute coordinates in the camera frame C. The lower right note i implies the index of the known feature points.

4.3 Navigation Dynamics and Output Equations

In this section, the dynamics for the integrated navigation system are provided for the map-based visual-inertial navigation system and the map-less visual-inertial navigation system. The main difference between the two systems is in their output equations. The map-based system, which depends on known maps or known landmarks, observes and reads these fixed features, whereas the map-less method uses the vision algorithm to detect particular types of features such as corners or edges and then matches them to estimate the vehicle motion through scaled relative pose measurements.

4.3.1 Map-Based Visual-Inertial Navigation Systems

The known features are denoted as $X_{N,i}$. Because we do not intend to build a map for the current indoor environment, the feature dynamics $\dot{X}_{N,i} = 0$ are omitted. No matter what kind of sensor combination method is taken, either

map-based or map-less, we assume that the IMU unit PX4FMU is placed in the center of mass of the quadrotor vehicle so that the IMU module is providing the inertial measurement of the vehicle. We denote the inertial measurement from the IMU module as the input of the system $u = [\omega_B^T \ f_B^T]^T$ where f_B is the specific force. Thus, the dynamics of the quadrotor vehicle are modeled in terms of (4.1)

$$\begin{bmatrix} \dot{t}_N \\ \dot{v}_N \\ \dot{R} \\ \dot{b}_\omega \\ \dot{b}_f \end{bmatrix} = \begin{bmatrix} v_N \\ Rf_B + g_N \\ R[\omega_B]_\times \\ n_{b_\omega} \\ n_{b_f} \end{bmatrix} \tag{4.6}$$

In the dynamic equations (4.6), the state variables t_N and v_n represent the absolute position and velocity with respect to the navigation frame N in the real world. The state variables b_ω and b_f are biases for the rate gyro and the accelerometer that are modeled in (3.3) as a random walk process. As the rotation dynamic is nonlinear, we cannot use the linear Kalman filter to estimate the vehicle states directly.

The calibration information, namely the transformation between the camera frame C and the body frame B, is assumed to have been calibrated before the maneuver process. The relative orientation and translation between B and C, i.e., R_B^C and t_C, are easy to obtain and remain unchanged during any maneuver process.

By using the normalized image coordinates output equation (4.5), the visual sensor output equation that depends on system states is given as

$$y_{C,i} = \pi(R_C^B R^T(X_{N,i} - t_N) - R_C^B t_C) + n_{C,i} \tag{4.7}$$

where i denotes the feature index $i = 1, \cdots, n_X$ and n_X represents the number of feature points being processed in the current Kalman filter update loop. $n_{C,i}$ is the visual sensor noise for the image coordinate $y_{C,i}$ and it is assumed to be white Gaussian.

4.3.2 Map-Less Visual-Inertial Navigation System

In the map-less visual-inertial navigation system, the camera acts as a real-time pose sensor and provides aiding measurements in a sensor fusion scheme

up to a translational scale. Since we assume that the features of the indoor environment are significant and not hard to detect and match, we do not need to add any feature dynamics to the navigation system equation. The scale factor λ is included in the system states because we want to recover the exact location of the quadrotor vehicle in the absolute scale. The navigation dynamics of the quadrotor vehicle in the map-less scheme are

$$\begin{bmatrix} \dot{t}_N \\ \dot{v}_N \\ \dot{R} \\ \dot{b}_\omega \\ \dot{b}_f \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} v_N \\ Rf_B + g_N \\ R[\omega_B]_\times \\ n_{b_\omega} \\ n_{b_f} \\ 0 \end{bmatrix} \tag{4.8}$$

The state variable λ represents the scaling between the vehicle's absolute position t_N in the navigation frame N and the corresponding scaled position measurement. The relative up-to-scale position is obtained from a monocular camera and computer vision algorithms. Because the camera's intrinsic parameters are not changed during the maneuver, the scale factor is modeled as a constant number. The calibration between the camera frame C and the vehicle body frame B is done before the maneuver process, which is the same as in the map-based design.

The output equations of the map-less visual-inertial navigation system use the up-to-scale pose measurement as the measurement variables. The scaled position measurement from the vision algorithm is the incremental amount added to the camera's previous scaled position. The scale position of the camera center is given as

$$y_{C,p} = (t_N + Rt_c)\lambda + n_{C,p} \tag{4.9}$$

This is the visual measurement output equation for the relative position. The other visual measurement equations for the relative orientation are given in terms of the rotation matrix of the camera center, namely the camera frame C with respect to the navigation frame N. Thus, the rotational visual measurement is given as

$$y_{C,R} = N_R R R_B^C \tag{4.10}$$

where $N_R \in \mathbf{SO}(3)$ is the multiplicative noise matrix. This adds to the non-linearity of the whole navigation system. The noise matrix is parameterized as a rotational noise vector $N_R = \exp(-[n_r]_\times)$ where n_r is a Gaussian white noise vector.

4.4 Numerical Integration

Although there are two visual-inertial navigation schemes proposed in this thesis, the integrations of their dynamics (4.6) and (4.8) use the same method at a sampling rate of 120 Hz. The states $\hat{x}^-$ are propagated using the input from the gyroscope in the first phase of the EKF loop. Next, the uncorrected states are updated periodically, using the aiding measurement in the second phase of the EKF loop.

The nominal (noise-free) dynamic equations of (4.6) are provided as follows

$$\begin{bmatrix} \dot{\hat{t}}_N \\ \dot{\hat{v}}_N \\ \dot{\hat{R}} \\ \dot{\hat{b}}_\omega \\ \dot{\hat{b}}_f \end{bmatrix} = \begin{bmatrix} \hat{v}_N \\ \hat{R}\hat{f}_B + g_N \\ \hat{R}[\hat{\omega}_B]_\times \\ 0 \\ 0 \end{bmatrix} \tag{4.11}$$

where $\hat{\omega} = y_\omega - \hat{b}_\omega$ and $\hat{f} = y_f - \hat{b}_f$. The input measurements y_ω and y_f are modeled in (3.1). The nominal system dynamics for the map-less method simply add the noise-free form of the absolute scale factor dynamic equation to (4.11), which is given as

$$\begin{bmatrix} \dot{\hat{t}}_N \\ \dot{\hat{v}}_N \\ \dot{\hat{R}} \\ \dot{\hat{b}}_\omega \\ \dot{\hat{b}}_f \\ \dot{\hat{\lambda}} \end{bmatrix} = \begin{bmatrix} \hat{v}_N \\ \hat{R}\hat{f}_B + g_N \\ \hat{R}[\hat{\omega}_B]_\times \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.12}$$

The nominal system is integrated between the time intervals t_0 and t_1. All the states are assumed constant during the integration procedure.

In order to obtain the integration of the rotational dynamic, we post-multiply it by the integrator $e^{-\int_{t_0}^{t}[\hat{\omega}_B(s)]_\times ds}$ and we have

$$\begin{aligned} \dot{\hat{R}}e^{-\int_{t_0}^{t}[\hat{\omega}_B(s)]_\times ds} &= \hat{R}[\hat{\omega}_B(s)]_\times e^{-\int_{t_0}^{t}[\hat{\omega}_B(s)]_\times ds} \\ \Rightarrow \frac{d}{dt}(\hat{R})e^{-\int_{t_0}^{t}[\hat{\omega}_B(s)]_\times ds} &= 0 \end{aligned} \tag{4.13}$$

Integrating the above equation from the time t_0 to t_1 in an EKF loop yields the analytic solution of

$$\hat{R}(t_1) = \hat{R}(t_0)e^{-\int_{t_0}^{t}[\hat{\omega}_B(s)]_\times ds} \tag{4.14}$$

If the above expression is evaluated using the trapezoidal rule

$$\int_a^b f(x)dx \approx (b-a)\left[\frac{f(a)+f(b)}{2}\right] \tag{4.15}$$

we obtain the update equation of the rotation dynamics in the form of

$$\hat{R}(t_1) = \hat{R}(t_0)\exp\left(\frac{t_1-t_0}{2}[y_\omega(t_0) - \hat{b}_\omega(t_0) + y_\omega(t_1) - \hat{b}_\omega(t_1)]_\times\right) \tag{4.16}$$

The integration of other dynamics is straightforward. For example by using (4.15), the integration of dynamics for $\hat{v}_N$, $\hat{t}_N$, $\hat{b}_\omega$, $\hat{b}_f$ and $\hat{\lambda}$ is given from the timestamp t_0 to t_1 as follows

$$\begin{aligned} \hat{v}_N(t_1) &= \hat{v}_N(t_0) + \frac{t_1-t_0}{2}(\hat{R}(t_0)\hat{f}(t_0) + \hat{R}(t_1)\hat{f}(t_1) + 2g_N) \\ \hat{t}_N(t_1) &= \hat{t}_N(t_0) + \frac{t_1-t_0}{2}(\hat{v}(t_0) + \hat{v}(t_1)) \\ \hat{b}_\omega(t_1) &= \hat{b}_\omega(t_0) \\ \hat{b}_f(t_1) &= \hat{b}_f(t_0) \\ \hat{\lambda}(t_1) &= \hat{\lambda}(t_0) \end{aligned} \tag{4.17}$$

4.5 Linearized Error Dynamics

The EKF theory requires the linearization of the current visual-inertial navigation system at the current Kalman filter estimate. After that, the state estimation is implemented based on the linearized system. In this section the linearized error dynamics are provided as

$$\begin{aligned} \delta\dot{x} &= F(t)\delta x + G(t)w \\ \delta y &= H(t)\delta x + v \end{aligned} \tag{4.18}$$

4.5.1 Linearization for Map-Based Navigation System

The error state of the linearized system is defined as $\delta x = x - \hat{x}$. Due to nonlinearity in the rotational dynamics, a new variable is introduced which can convert the matrix state variable R to a vector error state variable. In order to deal with the term $\delta R = R - \hat{R}$, we define $\omega_N := d\gamma/dt$ where $\gamma \in \mathbb{R}^3$ is a vector that represents the incremental attitude error. By taking the approximation $\delta R \approx R - \hat{R}$ and $d\gamma = \gamma - \hat{\gamma} = \delta\gamma$, we can derive from $\dot{R} = (\omega_N)_\times R$ and obtain the following approximation:

$$R \approx \hat{R} + [\delta\gamma]_\times \hat{R} \tag{4.19}$$

The term $\delta\gamma$ is the error state variable that takes the place of δR; the other states are updated using the normal form of $x = \hat{x} + \delta x$ since they have linear error dynamics.

While using the approximation (4.19) to update the rotation matrix $\hat{R}$, the numerical integration of the $\hat{R}$ matrix may deprive the orthogonal property of $\hat{R}$. To solve this issue, we use the method of SVD in [14] to re-orthogonalize the updated result of $\hat{R}$. By using the definition of $\delta\gamma$, one of the solutions that can preserve the orthogonality of the rotation matrix is given in [16]. This solution uses the integration factor $e^{-\int_{t_0}^{t}[\omega_N(s)]_\times ds}$ and the resulting rotational kinematics $R(t_1) = e^{\int_{t_0}^{t_1}[\omega_N]_\times ds} R(t_0)$. With the denotation of $R(t_0) = \hat{R}$, $R(t_1) = R$, the update of the rotation matrix takes the form of

$$R = e^{\int_{t_0}^{t_1}[d\gamma/ds]_\times ds]} \hat{R} = e^{[\delta\gamma]_\times} \hat{R} \tag{4.20}$$

In order to obtain the linearized rotational dynamic, recall the approximation (4.19) and take the derivative of both sides of the equation. We derive that

$$\dot{R} = \dot{\hat{R}} + [\delta\dot{\gamma}]_\times \hat{R} + [\delta\gamma]_\times \dot{\hat{R}} \tag{4.21}$$

Substituting both the rotational dynamic and the nominal rotational dynamic yields

$$R[\omega_B]_\times = \hat{R}[\hat{\omega}_B]_\times + [\delta\dot{\gamma}]_\times \hat{R} + [\delta\gamma]_\times \hat{R}[\hat{\omega}_B]_\times \tag{4.22}$$

Using the approximation (4.19) again, we have the derivations as follows:

$$\begin{aligned}
(\hat{R} + [\delta\gamma]_\times \hat{R})[\omega_B]_\times &= (\hat{R} + [\delta\gamma]_\times \hat{R})[\hat{\omega}_B]_\times + [\delta\dot{\gamma}]_\times \hat{R} \\
(\hat{R} + [\delta\gamma]_\times \hat{R})[\hat{\omega}_B - \delta b_\omega - n_\omega]_\times &= (\hat{R} + [\delta\gamma]_\times \hat{R})[\hat{\omega}_B]_\times + [\delta\dot{\gamma}]_\times \hat{R} \\
\hat{R}[-\delta b_\omega - n_\omega]_\times + \underbrace{[\delta\gamma]_\times \hat{R}[-\delta b_\omega - n_\omega]_\times}_{\approx 0} &= [\delta\dot{\gamma}]_\times \hat{R} \\
[-\delta b_\omega - n_\omega]_\times &= \hat{R}^T[\delta\dot{\gamma}]_\times \hat{R} = [\hat{R}^T\delta\dot{\gamma}]_\times \\
-\delta b_\omega - n_\omega &= \hat{R}^T\delta\dot{\gamma}
\end{aligned}$$

The last equation above gives the error dynamic for the rotational part

$$\delta\dot{\gamma} = -\hat{R}\delta b_\omega - \hat{R}n_\omega \tag{4.23}$$

The linearized error dynamics for the bias term $\delta b = b - \hat{b}$ are obtained in a straightforward way. Take the derivative and we have the dynamic for the bias term as follows

$$\delta\dot{b} = \dot{b} - \dot{\hat{b}} = n_b \tag{4.24}$$

In the same way we can get the error dynamic for the position state variable

$$\delta\dot{t}_N = \dot{v}_N - \dot{\hat{v}}_N = \delta v_N \tag{4.25}$$

and the error dynamic for the velocity term $\delta\dot{v}$ is given in the following steps

$$\begin{aligned}
\delta\dot{v}_N &= \dot{v}_N - \dot{\hat{v}}_N \\
&= Rf_B + g_N - (\hat{R}\hat{f}_B + g_N) \\
&= Rf_B - R\hat{f}_B + R\hat{f}_B - \hat{R}\hat{f}_B \\
&= -R(y_B - b_f - n_f - y_B + \hat{b}_f) + (R - \hat{R})\hat{f}_B \\
&= -(\hat{R} + [\delta\gamma]_\times \hat{R})(\delta b_f + n_f) + [\delta\gamma]_\times \hat{R}\hat{f}_B \\
&= -\hat{R}\delta b_f - \underbrace{[\delta\gamma]_\times \hat{R}\delta b_f}_{\approx 0} - \hat{R}n_f - \underbrace{[\delta\gamma]_\times \hat{R}n_f}_{\approx 0} + [\delta\gamma]_\times \hat{R}\hat{f}_B \\
&= -\hat{R}\delta b_f - \hat{R}n_f + \delta\gamma \times (\hat{R}\hat{f}_B) \\
&= -\hat{R}\delta b_f - \hat{R}n_f - (\hat{R}\hat{f}_B) \times \delta\gamma
\end{aligned}$$

Rearranging the last equation renders the error dynamics for the velocity term

$$\delta\dot{v}_N = [\hat{R}\hat{f}_B]_\times \delta\gamma - \hat{R}\delta b_f - \hat{R}n_f \tag{4.26}$$

The linearized error term for the output equation is obtained by calculate the difference between the visual measurements and their estimates. This linearized error term generates the correction update for the navigation system through the sensor fusion scheme of EKF. The linearization process starts with stacking the image feature measurements into a column vector $y = [\ y_{C,1}^T \quad \cdots \quad y_{C,n_X}^T\ ]^T$. These visual measurements are corrupted by the white Gaussian noise randomly present in the images. The noise can be generated from varying illuminations, which results in errors is visual measurements. The feature coordinates processed in the stacked vector y are in normalized image coordinates rather than in pixels. The normalized feature coordinates then enter into the Kalman filter loop. Based on these assumptions, the error term is derived in the following steps:

$$\begin{aligned} \delta y_{C,i} &= y_{C,i} - \pi(\hat{X}_{C,i}) \\ &= H_i(\hat{x})\delta x + n_{C,i} \end{aligned} \tag{4.27}$$

where $H_i(\cdot)$ is the Jacobian matrix of the nonlinear function $\pi(\cdot)$ with respect to the current state estimates. In order to calculate this Jacobian matrix, we denote the ith feature point in the camera frame as $X_{C,i} = R_C^B R^T(X_{N,i} - t_N) - R_C^B t_c$, which translates the output equation into

$$\begin{aligned} y_{C,i} &= \pi(X_{C,i}) + n_{C,i} \\ &= \begin{bmatrix} X_{C,i}(1)/X_{C,i}(3) \\ X_{C,i}(2)/X_{C,i}(3) \end{bmatrix} + n_{C,i} \end{aligned} \tag{4.28}$$

The partial derivative with respect to the states in the map-based method is given as

$$\frac{\partial X_{C,i}}{\partial x} = \begin{bmatrix} -R_C^B & 0_{3\times3} & [X_{C,i}]_\times & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{4.29}$$

This leads to the calculation of the Jacobian matrix for the ith visual measurement as follows

$$\begin{aligned} H_i(x) &= H_{X_{C,i}} \cdot \frac{\partial X_{C,i}}{\partial x} \\ &= \frac{1}{X_{C,i}^2(3)} \begin{bmatrix} X_{C,i}(3) & 0 & -X_{C,i}(1) \\ 0 & X_{C,i}(3) & -X_{C,i}(2) \end{bmatrix} \frac{\partial X_{C,i}}{\partial x} \end{aligned} \tag{4.30}$$

where $H_i(x) \in \mathbb{R}^{2\times15}$. As we stack the feature measurement vector into a united vector, the error terms of the visual output measurements are also put

into a joint vector and so are the Jacobian matrices for the output linearized terms. Thus, the linearized output error dynamic for the map-based visual-inertial navigation system is given as

$$\begin{aligned}\delta y_C &= H_C(\hat{x})\delta x + n_C \\ &= \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_{n_X} \end{bmatrix} \delta x + \begin{bmatrix} n_{C,1} \\ n_{C,2} \\ \vdots \\ n_{C,n_X} \end{bmatrix}\end{aligned} \tag{4.31}$$

The linearized error equations are given as

$$\begin{bmatrix} \delta\dot{t}_N \\ \delta\dot{v}_N \\ \delta\dot{\gamma} \\ \delta\dot{b}_\omega \\ \delta\dot{b}_f \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -[\hat{R}\hat{f}]_\times & 0 & -\hat{R} \\ 0 & 0 & 0 & -\hat{R} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta t_N \\ \delta v_N \\ \delta\gamma \\ \delta b_\omega \\ \delta b_f \end{bmatrix} \tag{4.32}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\hat{R} & 0 & 0 \\ -\hat{R} & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} n_\omega \\ n_f \\ n_{b_\omega} \\ n_{b_f} \end{bmatrix} \tag{4.33}$$

which can be abbreviated as

$$\delta\dot{x} = F\delta x + Gw$$

with $w = [n_\omega^T \; n_f^T \; n_{b_\omega}^T \; n_{b_f}^T]^T$. The noise vector w is characterized by the covariance matrix $Q = E < ww^T >= \text{diag}(\sigma_\omega^2, \sigma_f^2, \sigma_{b_\omega}^2, \sigma_{b_f}^2)$.

4.5.2 Linearization for Map-Less Navigation System

The difference between the linearized map-based and map-less systems is in the linearization procedure of the absolute scale term and the linearized output equation. For the map-less visual-inertial navigation system, the error dynamic with the absolute scale term is given as

$$\delta\dot{\lambda} = 0 \tag{4.34}$$

In the map-less method, the scaled pose aiding measurement is a vector of fixed length and the resulting error dynamics consist of two parts: one with

the scaled position measurement and the other with the relative attitude measurement. For the scaled position term, the linearized error dynamic is

$$\begin{aligned}\delta y_{C,p} &= y_{C,p} - \hat{y}_{C,p} \\ &= (t_N + Rt_C)\lambda + n_{C,p} - (\hat{t}_N + \hat{R}t_C)\hat{\lambda} \\ &= H'_p \delta x + n_{C,p}\end{aligned} \tag{4.35}$$

where $H'_p = [\ \hat{\lambda}I \quad 0_{3\times3} \quad -\hat{R}[t_C]_\times\hat{\lambda} \quad 0_{3\times3} \quad 0_{3\times3} \quad -(\hat{t}_N + \hat{R}t_C)\ ]$ that represents the output Jacobian matrix for the map-less navigation system. Based on the matrix property (2.7) introduced in Chapter 2, a rotation matrix perturbation, namely the associated error term, can be approximated as

$$\delta R = 1 - \delta\theta_\times \tag{4.36}$$

Since a rotation matrix can also be termed as $R = \exp(-[\alpha]_\times)$. The rotational output measurement $y_{C,R} = N_R R R_B^C$ from the visual sensor can be approximated by

$$(1 - \delta[\phi_m]_\times)\hat{y}_{C,R} = (1 - \delta[\rho]_\times)\hat{N}_R(1 - \delta[\gamma]_\times)\hat{R}R_B^C \tag{4.37}$$

Because ρ is a zero mean Gaussian white noise vector, it can be concluded that $\hat{N} = I$. For the nominal system, we have that $y_{C,R} = \hat{R}R_B^C$, where the rotation matrix R_B^C is assumed to have been calibrated beforehand. We then obtain the expression of

$$1 - \delta[\phi_m]_\times = (1 - \delta[\rho]_\times)(1 - \delta[\gamma]_\times) \tag{4.38}$$

By expanding (4.38) and omitting the higher order term, we have

$$\delta[\phi_m]_\times = \delta[\rho]_\times + \delta[\gamma]_\times \tag{4.39}$$

which is equivalent to

$$\begin{aligned}\delta\phi_m &= \delta\rho + \delta\gamma \\ &= H'_R\delta x + n_{C,R}\end{aligned} \tag{4.40}$$

where $H'_R = [\ I_{3\times3} \quad 0_{3\times3} \quad 0_{3\times3} \quad 0_{3\times3} \quad 0_{3\times3} \quad 0_{3\times1}\ ]$ and $\delta\phi_m$ is the incremental relative measurement computed from $y_{C,R}$. Note that the term $\delta\rho$ is rewritten as $n_{C,R}$ which is assumed as a white Gaussian noise vector. By combining

(4.35) and (4.40) we can obtain the linearized error dynamics for the map-less visual-inertial navigation system

$$\begin{bmatrix} \delta y_{C,p} \\ \delta \phi_m \end{bmatrix} = \begin{bmatrix} H'_p \\ H'_R \end{bmatrix} \delta x + \begin{bmatrix} n_{C,p} \\ n_{C,R} \end{bmatrix} \tag{4.41}$$

which is abbreviated as

$$\delta y_C = H'_C \delta x + n'_C \tag{4.42}$$

where H'_C is a combination matrix of the Jacobian H'_p and H'_R. So the linearized error equations for the map-less visual-inertial system are given as follows

$$\begin{bmatrix} \delta \dot{t}_N \\ \delta \dot{v}_N \\ \delta \dot{\gamma} \\ \delta \dot{b}_\omega \\ \delta \dot{b}_f \\ \delta \dot{\lambda} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & -[\hat{R}\hat{f}]_\times & 0 & -\hat{R} & 0 \\ 0 & 0 & 0 & -\hat{R} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta t_N \\ \delta v_N \\ \delta \gamma \\ \delta b_\omega \\ \delta b_f \\ \delta \lambda \end{bmatrix} \tag{4.43}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\hat{R} & 0 & 0 \\ -\hat{R} & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} n_\omega \\ n_f \\ n_{b_w} \\ n_{b_f} \end{bmatrix} \tag{4.44}$$

where the covariance matrix of the noise vector is defined in the same way as in the map-based design.

4.6 Observability Analysis

In order to verify that the EKF-based methods in this thesis are able to generate an unbiased estimate of the true system state, the observability analysis process is performed. Different observability analysis of either a linear or a nonlinear model for visual-inertial navigation design is reviewed in the paper of [25]. If the navigation systems model is nonlinear and in the input affine form, the observability analysis can be conducted using differential geometry. The nonlinear system from with affine form presented in [27, 52] is expressed as

$$\begin{cases} \dot{x} = f_0(x) + \sum_{i=1}^{n_u} f_i(x) u_i \\ y = h(x) \end{cases} \tag{4.45}$$

where $u \in \mathbb{R}^{n_u}$ is the input vector and $u = [\ u_1 \quad \cdots \quad u_{n_u}\ ]$. (4.45) is used for nonlinear observability analysis.

If the system model is linear time-invariant, the observability of the system can be determined directly using the rank test of the observability matrix. In regards to the linear time-varying (LTV) system, we need to numerically evaluate the corresponding observability Gramian matrix or to regard states of the LTV system as piecewise constant scalars during the sampling interval and then use the stripped observability matrix. Directly examining the observability is difficult because the navigation system in this thesis is not in the control affine form unless quaternion is used for rotation dynamics.

4.6.1 Definitions

The references [25, 27, 52] examine the observability of the navigation system based on distinguishable trajectories. We consider the system of (4.1) with $x \in X$, $y \in Y$, $u \in U$ where X, Y and U are open subsets of $\mathbb{R}^n$, $\mathbb{R}^m$ and $\mathbb{R}^p$ respectively. In the nonlinear theory, the two states $x_a, x_b \in X$ are *indistinguishable* and are denoted as $x_a \vee x_b$ if the output maps are equal, i.e.,

$$y_{x_a,u}(t) = y_{x_b,u}(t) \tag{4.46}$$

for any input $u \in U$ and any time stamp in the vehicle maneuver process. Thus, we define that the system (4.1) is observable for any two states $x_a, x_b \in X$ such that

$$x_a \vee x_b \Longleftrightarrow x_a = x_b \tag{4.47}$$

which means that for an admissible control $u \in U$ and a time $t \geq 0$ we have

$$y_{x_a,u}(t) \neq y_{x_b,u}(t) \tag{4.48}$$

indicating that the x_a and x_b are indistinguishable. Therefore, the observability for the navigation system (4.1) is defined in the following way: (4.1)

is described as *locally observable* at $x_a \in X$ if there exists a neighborhood V of x_a and $V \subseteq X$ such that for any $x_b \in V$, the states x_a and x_b are distinguishable. If the system states are restricted in an open set $V \subseteq X$ and this state-restricted system is locally observable, we say the system is *strongly locally observable*. In contrast, the general concept of locally observable is often defined as *weakly locally observable*. The visual-inertial navigation system design in this thesis is designed to show weak local observability of the proposed VI navigation system.

4.6.2 Observability Analysis for Map-based Navigation System

Rather than directly examining the observability of the nonlinear system, we verify the observability of the linearized system, based on the same method in [2]. Due to approximation in linearization is to omit the second order and higher order terms, the linearization process can result in information loss. The error propagation here may add rank to the observability matrix. The unobservable direction analysis is thus provided after the rank test process.

The theory in [6], which addresses the observability of the LTV system, is briefly introduced here. The input and output measurements enter into the Jacobian matrices F, G and H_C (omit the subscript C) in the EKF method. We assume the system is n-dimension. The pair $(F(t), H(t))$ is observable at t_0 if there exists a finite $t_1 > t_0$ such that the observability matrix is of full rank

$$\text{rank}(\mathcal{O}) = \text{rank} \begin{bmatrix} N_0(t_1) \\ N_1(t_1) \\ \vdots \\ N_{n-1}(t_1) \end{bmatrix} = n \tag{4.49}$$

where $N_0(t) = H_C(t)$ and the following matrix is defined as

$$N_{m+1}(t) = N_m(t)F(t) + \frac{d}{dt}N_m(t), \quad m = 0, 1, \cdots, n-1 \tag{4.50}$$

Because the linearized output equations in the map-based schemes and mapless schemes are different, the observability analyses are conducted separately. We denote $\mathcal{O}_{mb}$ and $\mathcal{O}_{ml}$ as the corresponding observability matrices for the

two design methods. As for the map-based navigation design, the system order is $n = 15$. Due to each feature's output Jacobin has the same structure (see (4.29) (4.30)), only the output Jacobian for the ith feature is considered here. Starting from N_0, we have

$$N_0 = H_i = \begin{bmatrix} -a_i R_C^B & 0 & a_i[X_{C,i}]_\times & 0 & 0 \\ -b_i R_C^B & 0 & b_i[X_{C,i}]_\times & 0 & 0 \end{bmatrix} \tag{4.51}$$

where a_i represents the first row of the matrix $H_{X_{C,i}}$ and b_i represents the second row. The second row block matrix is computed as

$$\begin{aligned} N_1 &= N_0 F(t) + \dot{N}_0(t) \\ &= \begin{bmatrix} 0 & a_i R_C^B & 0 & -a_i[X_{C,i}]_\times & 0 \\ 0 & b_i R_C^B & 0 & -b_i[X_{C,i}]_\times & 0 \end{bmatrix} \end{aligned} \tag{4.52}$$

Through the nominal system dynamic $\dot{\hat{R}} = \hat{R}[\hat{\omega}_B]_\times$, the third row block matrix is computed as

$$\begin{aligned} N_2 &= N_1 F(t) + \dot{N}_1(t) \\ &= \begin{bmatrix} 0 & 0 & a_i R_C^B[\hat{R}\hat{f}]_\times & -a_i[X_{C,i}]_\times \hat{R}[\hat{\omega}_B]_\times & -a_i R_C^B \hat{R} \\ 0 & 0 & b_i R_C^B[\hat{R}\hat{f}]_\times & -b_i[X_{C,i}]_\times \hat{R}[\hat{\omega}_B]_\times & -b_i R_C^B \hat{R} \end{bmatrix} \end{aligned} \tag{4.53}$$

N_3 is thus computed as

$$\begin{aligned} N_3 &= N_2 F(t) + \dot{N}_2(t) \\ &= \begin{bmatrix} 0 & 0 & a_i R_C^B([\dot{\hat{R}}\hat{f}]_\times) & -a_i(R_C^B[\hat{R}\hat{f}]_\times \hat{R} + [X_{C,i}]_\times(\dot{\hat{R}}[\hat{\omega}_B]_\times)) & -a_i R_C^B[\hat{\omega}_B]_\times \\ 0 & 0 & b_i R_C^B([\dot{\hat{R}}\hat{f}]_\times) & -b_i(R_C^B[\hat{R}\hat{f}]_\times \hat{R} + [X_{C,i}]_\times(\dot{\hat{R}}[\hat{\omega}_B]_\times)) & -b_i R_C^B[\hat{\omega}_B]_\times \end{bmatrix} \end{aligned} \tag{4.54}$$

Since the form of the next few row blocks requires more complex matrix derivation, the observability matrix for the map-based matrix is given as

$$\mathcal{O}_{mb} = \begin{bmatrix} -a_i R_C^B & 0 & a_i[X_{C,i}]_\times & 0 & 0 \\ -b_i R_C^B & 0 & b_i[X_{C,i}]_\times & 0 & 0 \\ 0 & a_i R_C^B & 0 & -a_i[X_{C,i}]_\times & 0 \\ 0 & b_i R_C^B & 0 & -b_i[X_{C,i}]_\times & 0 \\ 0 & 0 & a_i R_C^B[\hat{R}\hat{f}]_\times & -a_i[X_{C,i}]_\times \hat{R}[\hat{\omega}_B]_\times & -a_i R_C^B \hat{R} \\ 0 & 0 & b_i R_C^B[\hat{R}\hat{f}]_\times & -b_i[X_{C,i}]_\times \hat{R}[\hat{\omega}_B]_\times & -b_i R_C^B \hat{R} \\ * & * & * & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.55}$$

As can be seen from the observability matrix $\mathcal{O}_{mb}$, the rotation matrix R exists in almost every non-zero term of the observability matrix. A rotation matrix

R implies $R \in SO(3)$ and R is invertible. But analytically determining the rank of the observability is not straightforward.

In order to analyze the rank of the observability matrix, we can recheck of the problem investigated. The VI navigation system should generate estimation of the robot vehicle pose states in a global way. But in fact, the observability matrix $\mathcal{O}_{mb}$ of the linearized error system is rank deficient in one column. Intuitively, this unobservable direction is the global yaw, namely the absolute heading of the navigation system, because we cannot obtain any aiding information about magnetic North. However, a practical consideration is that since the quadrotor is maneuvering in the indoor environment, it is not as important to get the absolute yaw aiding as it would be in the outdoor environment.

Although the yaw angle ψ may slowly diverge, according to Claim 4 in [25], we use the enforced initial condition that the initial pose pair is set as $(I, 0_3)$. By saturating the filter along three visible directions and thus fixing the global reference in the meantime, the navigation system is locally observable as a result.

An alternative way of making the system of full observability is to add the aiding sensor which can provide the yaw, i.e., using the magnetometer. The magnetometer yaw aiding is proposed in [2] with the magnetometer measurement $y = R^T m_N + n_m$, where m_N is the magnetometer measurement reading and is corrupted with white Gaussian noise n_m. By using the approximation (4.19), the yaw-only magnetometer output equation is given as

$$\delta y_m = \hat{R}[m_N]_\times [0\ 0\ E_3\gamma]_\times + n_m \tag{4.56}$$

4.6.3 Observability Analysis for Map-less Navigation System

The observability analysis for the map-less navigation system follows the same principle as the map-based navigation system. We compute the observability matrix for the LTV system (4.43). The difference in the analysis of the map-less navigation system design includes the system dimension and the linearized

output equations H'_C. The observability matrix no longer depends on the constant landmark coordinates. All the output equations are used to evaluate the rank of the observability matrix. The first matrix block term N_0 is simply obtained from the linearized output equation.

$$N_0 = H'_C = \begin{bmatrix} \hat{\lambda}I & 0 & 0 & -\hat{R}[t_C]_\times\hat{\lambda} & 0 & -(\hat{t}_N + \hat{R}t_C) \\ I & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.57}$$

Then the following matrix blocks are computed as

$$\begin{aligned} N_1 &= N_0F(t) + \dot{N}_0(t) \\ &= \begin{bmatrix} 0 & \hat{\lambda}I & 0 & -\hat{R}[t_C]_\times\hat{\lambda} & 0 & -(\hat{v}_N + \dot{\hat{R}}t_C) \\ 0 & I & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \hat{\lambda}I & 0 & -\hat{R}[\omega_B]_\times[t_C]_\times\hat{\lambda} & 0 & -(\hat{v}_N + \dot{\hat{R}}t_C) \\ 0 & I & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.58}$$

$$\begin{aligned} N_2 &= N_1F(t) + \dot{N}_1(t) \\ &= \begin{bmatrix} 0 & 0 & -\hat{\lambda}([\hat{R}\hat{f}]_\times + \ddot{\hat{R}}[t_C]_\times) & 0 & -\hat{\lambda}\hat{R} & -(\dot{\hat{v}}_N + \ddot{\hat{R}}t_C) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.59}$$

$$\begin{aligned} N_3 &= N_2F(t) + \dot{N}_2(t) \\ &= \begin{bmatrix} 0 & 0 & -\hat{\lambda}(\frac{d}{dt}([\hat{R}\hat{f}]_\times) + \hat{R}^{(3)}[t_C]_\times) & \hat{\lambda}([\hat{R}\hat{f}]_\times + \ddot{\hat{R}}[t_C]_\times)\hat{R} & -\hat{\lambda}\dot{\hat{R}} & -(\ddot{\hat{v}}_N + \hat{R}^{(3)}t_C) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.60}$$

Then we provide the expression of the observability matrix for the linearized error dynamics of the map-less navigation system as

$$\mathcal{O}_{ml} = \begin{bmatrix} \hat{\lambda}I & 0 & 0 & -\hat{R}[t_C]_\times\hat{\lambda} & 0 & -(\hat{t}_N + \hat{R}t_C) \\ I & 0 & 0 & 0 & 0 & 0 \\ 0 & \hat{\lambda}I & 0 & -\hat{R}[\omega_B]_\times[t_C]_\times\hat{\lambda} & 0 & -(\hat{v}_N + \dot{\hat{R}}t_C) \\ 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & -\hat{\lambda}([\hat{R}\hat{f}]_\times + \ddot{\hat{R}}[t_C]_\times) & 0 & -\hat{\lambda}\hat{R} & -(\dot{\hat{v}}_N + \ddot{\hat{R}}t_C) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\hat{\lambda}(\frac{d}{dt}([\hat{R}\hat{f}]_\times) + \hat{R}^{(3)}[t_C]_\times) & \hat{\lambda}([\hat{R}\hat{f}]_\times + \ddot{\hat{R}}[t_C]_\times)\hat{R} & -\hat{\lambda}\dot{\hat{R}} & -(\ddot{\hat{v}}_N + \hat{R}^{(3)}t_C) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * \end{bmatrix} \tag{4.61}$$

The observability matrix (4.61) has the block matrix terms up to N_{15} and here we omit the block matrices N_i that have higher order derivatives. The rank analysis cannot determine the rank of the observability matrix directly from (4.61), because rotation matrices are commonly distributed and a rotation matrix is of full rank.

As we discussed in 4.6.2, the global yaw, i.e., the absolute heading, is not located in the fully observable direction. By enforcing the initial pose pair to

$(I, 0_3)$ as in 4.6.2 and fixing the global reference, the global yaw direction is made locally observable. In fact, there is another unobservable state in the map-less design, which is the absolute scale. As the visual aiding measurement are scaled, the absolute scale can only be updated in the state propagation process. In addition, this absolute scale factor can be jointly observable as in [52], with the vehicle velocities and the vehicle absolute positions.

4.7 Discretization

Implementing the EKF-based navigation system requires discrete time expression of the linearized error dynamic system. This section gives the expression and calculation of the discretized error dynamics

$$\begin{aligned} \delta x_{k+1} &= \Phi_k \delta x_k + w_k \\ \delta y_k &= H_k \delta x_k + v_k \end{aligned} \tag{4.62}$$

where the noise vectors w_k and v_k are described by the covariance matrix $Q_k := E < w_k w_k^T >$ and $R_k := E < v_k v_k^T >$.

We denote consecutive sampling time steps as t_k and t_{k+1} and the sampling time interval is thus $\tau = t_{k+1} - t_k$. The state estimate at the time stamp t_k is $\hat{x}_k$. By evaluating the matrices of the error dynamics (4.18) at the timestamp t_k with the estimated states $\hat{x}_k$, we get $F_k = F(\hat{x}_k)$, $G_k = G(\hat{x}_k)$ and $H_k = H(\hat{x}_k)$ which are assumed constant during the time interval $[t_k, t_{k+1}]$. Therefore the matrix expressions in (4.62), namely the discrete-time transition matrix and the process noise vector, are formed as

$$\begin{aligned} \Phi &= e^{F_k \tau} \\ w_k &= \int_{t_k}^{t_{k+1}} e^{F_k (t_{k+1} - t)} G_k w(t) dt \end{aligned} \tag{4.63}$$

The covariance matrix of white noise vector w_k is derived as

$$\begin{aligned} Q_k &= E < w_k w_k^T > \\ &= E < \int_{t_k}^{t_{k+1}} e^{F_k (t_{k+1} - t)} G_k w(t) dt \int_{t_k}^{t_{k+1}} w^T(s) G_k^T e^{F_k^T (t_{k+1} - s)} ds > \\ &= E < \int_{t_k}^{t_{k+1}} \int_{t_k}^{t_{k+1}} e^{F_k (t_{k+1} - t)} G_k w(t) w^T(s) G_k^T e^{F_k^T (t_{k+1} - s)} dt ds > \end{aligned}$$

Moving the expectation inside yields

$$Q_k = \int_{t_k}^{t_{k+1}} \int_{t_k}^{t_{k+1}} e^{F_k(t_{k+1}-t)} G_k \underbrace{E < w(t)w^T(s) >}_{Q\delta(s-t)} G_k^T dt e^{F_k^T(t_{k+1}-s)} ds$$
$$= \int_{t_k}^{t_{k+1}} e^{F_k^T(t_{k+1}-s)} G_k Q G_k^T e^{F_k^T(t_{k+1}-s)} ds$$

where δ is the Dirac delta function (also referred to as the unit pulse function) and the continuous white noise vector is described as a stationary random process. The autocorrelation of the noise vector is $E < w(t)w(t+s) >= \sigma^2\delta(s)$. Since the covariance matrix Q_k involves the integration of the multiplication of the matrix terms between any two successive sample times t_{k+1} and t_k, we reevaluate it by using the method introduced in [14]. We start with forming the matrix of

$$\Gamma = \begin{bmatrix} -F_k & G_k Q G_k^T \\ 0 & F_k^T \end{bmatrix} \tag{4.64}$$

The matrix exponential $e^{\Gamma\tau}$ is calculated as

$$e^{\Gamma\tau} = \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ 0 & \Lambda_3 \end{bmatrix} = \begin{pmatrix} e^{-F_k\tau} & \int_0^t e^{-F_k(\tau-s)} G_k Q G_k^T e^{F_k^T\tau} ds \\ 0 & e^{F_k^T\tau} \end{pmatrix} \tag{4.65}$$

By using the variable substitution $s = t_{k+1} - t$, we have

$$\begin{aligned} \Lambda_2 &= e^{-F_k\tau} \int_0^\tau e^{F_k s} G_k Q G_k^T e^{F_k^T s} ds \\ &= (e^{F_k\tau})^{-1} \int_{t_k}^{t_{k+1}} e^{F_k(t_{k+1}-t)} G_k Q G_k^T e^{F_k^T(t_{k+1}-t)} dt \end{aligned} \tag{4.66}$$

Thus, we have the calculation expression of the covariance matrix for the noise vector w_k as

$$Q_k = (\Lambda_3)^T \Lambda_2 = \Phi_k \Lambda_2 \tag{4.67}$$

As in the linearized error dynamics, vector v is used to denote the visual measurement noise n_C and the discrete version of v is v_k. This white noise vector v is characterized by covariance matrix $R = E < v(t)v(t+s) >= \sigma^2\delta(s)$ and v_k is carved by $R_d = E < v_k v_{k+\varepsilon} >= \sigma_d^2\delta_\varepsilon$. In order to calculate v_k we explore the power spectral density of v which is given as

$$S_c(\omega) = \mathcal{F}_c(R) = \int_{-\infty}^{\infty} \sigma^2\delta(\tau) e^{-j\omega\tau} d\tau = \sigma^2 \tag{4.68}$$

The power spectral density of the discrete noise vector v_k is given as

$$S_d(\Omega) = \mathcal{F}_d(R_d) = \sum_{\varepsilon=-\infty}^{\varepsilon=\infty} \sigma_d^2 \delta_\varepsilon e^{-j\Omega\varepsilon} = \sigma_d^2 \tag{4.69}$$

It can be derived that

$$S_d(\Omega) = \frac{1}{T} S_c(\frac{\Omega}{T}) = \frac{\sigma^2}{T} \tag{4.70}$$

Therefore, the covariance of the discrete white noise vector v_k is given as

$$\sigma_d^2 = \frac{\sigma^2}{T} \tag{4.71}$$

The covariance matrix for v_k is derived as

$$R_k = \frac{1}{\tau} R \tag{4.72}$$

The covariance matrix Q_k and R_k are used for algorithm implementations.

4.8 Extended Kalman Filter

What actually enter into the filter loop are the states in the discrete-time linearized error dynamics (4.62). Based on the EKF theory, as the linear discrete-time Kalman filter is applied to the error dynamics, the error state is updated, prepared for the update of the navigation system states in the next filter loop. The linearized error dynamics and the navigation system dynamics share the noise vectors of the same stochastic process. The noise vectors w_k and v_k are assumed to be white zero-mean noise vectors with known covariance matrices Q_k and R_k. They are assumed to be uncorrelated, which is summarized as

$$\begin{aligned} w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \\ E < w_k v_j^T > &= 0 \end{aligned}$$

The last assumption is easy to understand because w_k and v_k are process noise and measurement noise coming from different sensors.

The Kalman filter commonly contains two steps: the state propagation and the state correction. At the time stamp k, the updated linearized error system state is denoted as $\delta\hat{x}_k^-$ and the estimated state, corrected by the

aiding measurements, is denoted as $\delta\hat{x}_k^+$. In order to simplify the notation, we omit the superscript + here. The prior updated estimation error is given as $e_k^- = \delta x_k - \delta\hat{x}_k^-$ and the covariance of this error is $P_k^- = E < e_k^- \cdot (e_k^-)^T >$. The Kalman filter uses the following observer form to perform the post state update

$$\delta\hat{x}_k = \delta\hat{x}_k^- + K_k(\delta y_k - H_k\delta\hat{x}_k^-) \tag{4.73}$$

where K_k is the Kalman filter gain. The gain K_k is determined by minimizing the trace of the covariance matrix $P_k = E < e_k \cdot e_k^T >$ of the post state update error $e_k = \delta x_k - \delta\hat{x}_k$. By substituting the above equation (4.73) into P_k, we can derive the expression of P_k in terms of corresponding matrices

$$\begin{aligned} P_k &= E\left\langle(\delta x_k - \delta\hat{x}_k^- - K_k(\delta y_k - H_k\delta\hat{x}_k^-)(\delta x_k - \delta\hat{x}_k^- - K_k(\delta y_k - H_k\delta\hat{x}_k^-)^T\right\rangle \\ &= E\left\langle(e_k^- - K_k(H_k e_k^- + v_k))(e_k^- - K_k(H_k e_k^- + v_k))^T\right\rangle \\ &= (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k R_k K_k^T \end{aligned} \tag{4.74}$$

which is in quadratic terms of K_k. Through the matrix derivative formula, the derivative of $\mathrm{Tr}(P_k)$ with respect to K_k is set to zero, which is

$$\frac{d(\mathrm{Tr}(P_k))}{dK_k} = -2(H_k P_k^-)^T + 2K_k H_k P_k^- H_k^T + 2K_k R_k = 0 \tag{4.75}$$

We can obtain the Kalman filter calculation term

$$K_k = P^- H_k^T(H_k P_k^- H_k^T + R_k)^{-1} = P_k H_k^T R_k^{-1} \tag{4.76}$$

The state update (4.73) is performed using this calculated Kalman gain. We prepare the next state propagation for the time stamp $k+1$ using

$$\delta\hat{x}_{k+1}^- = \Phi_k\delta\hat{x}_k \tag{4.77}$$

And the new error covariance is

$$\begin{aligned} P_{k+1}^- &= E\left\langle(\delta x_{k+1} - \delta\hat{x}_{k+1}^-)(\delta x_{k+1} - \delta\hat{x}_{k+1}^-)^T\right\rangle \\ &= \Phi_k P_k \Phi_k^T + Q_k \end{aligned} \tag{4.78}$$

In the next Kalman filter loop, when the measurement update information is ready, we calculated the new Kalman gain and perform the state update correction.

The system actually updates the state estimations $\hat{x}$ and $\hat{y}$ during the EKF design. In the same way that the error state is defined with superscripts $+$ and $-$, we denote the propagated state at time k as $\hat{x}_k^-$ and the estimated output as $\hat{y}_k^-$. Then, in connection with the error state definition $\delta x_k = x_k - \hat{x}_k^-$ and $\delta y_k = y_k - \hat{y}_k^-$, the estimated error state is given as $\delta\hat{x}_k = \hat{x}_k - \hat{x}_k^-$ as well as $\delta\hat{x}_k^- = \hat{x}_k^- - \hat{x}_k^- = 0$. As a consequence, the navigation state update is given as

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - \hat{y}_k^-) \tag{4.79}$$

where K_k and P_k in this navigation state update process are calculated as given in (4.76) and (4.74), respectively. After the state estimations are updated, the linearization is executed at the estimated state trajectory $\hat{x}_k$. At this moment the error state and its estimation are $\delta x_k = x_k - \hat{x}_k$ and $\delta\hat{x}_k = \hat{x}_k - \hat{x}_k = 0$. We can see the linearized error state propagation becomes trivial and is omitted, while the error covariance matrix is being updated through the updated $\hat{x}_k$.

The EKF state update in the correction stage may not be periodic because the filter loop expectation is done with or without the measurement. In an aperiodic update case, we record the instant time as t_a and compute the elapsed time as $\tau = t - t_a$ which is used to update Φ_k, Q_k and the error covariance P_k as well.

So we conclude that we start the EKF loop with the initialization of the state $\hat{x}_0 = E < x_0 >$ and the covariance matrix of the state error of $P_0 = E < (x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T >$. It is common to set the matrix P_0 as a zero matrix and the matrix R_0 as an identity matrix in practice but they can be tuned. In order to make the EKF execution more stable, we assign extremely small value rather than a pure zero matrix to P_0. When the aiding variable is ready, we perform the following update of the state correction:

$$\begin{aligned} K_k &= P^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(y_k - \hat{y}_k^-) \\ P_k &= (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \end{aligned} \tag{4.80}$$

where P_k^- has been calculated in the state propagation process using (4.74).

A more practical consideration in implementation is to identify the false visual aiding information. Here, false means that the aiding measurement

is either generated by the feature outliers or there are not enough features. Because we know that in order to estimate the six degrees of freedom for the vehicle's full pose states, the minimum points needed are one fewer than the degree of freedom: in other words, five-point coordinates are required. The absent point requirement is the unobservable direction of the absolute scale. In this thesis, as we estimate the full pose of the quadrotor vehicle, we examine the number of effective points that are used for the EKF design. If the number of feature points passes the minimum points number requirement and the features are recognizable, the visual aiding measurements are used for the state update in the EKF loop. Since the absolute scale is jointly observable with position states, we calculate it through three position directions, average the result and weight the result with the initial value.

4.9 Results

This section provides results of state estimation in simulation based on work and scripts in [26, 2]. First we provide the map-based visual-inertial navigation results of the quadrotor vehicle states in two cases: taking off and hovering. After that, we show the results for the map-less visual-inertial navigation system with the quadrotor state estimation.

4.9.1 Map-Based Visual-Inertial Navigation Results

The map-based visual-inertial navigation design requires a known map with recognizable features. The map is built in the same way as the one in [21] which is shown in Fig. 4.3. We assume that the 3D world coordinates of these features in the map are known. In Fig. 4.3 the visual features are evenly distributed in the interior surface of a virtual cylinder with a radius of 2 m and a height of 2 m. There are thirty-six feature pillars in the figure. Each pillar has twenty-four features, which make a full visual feature environment for different kinds of tests. The camera is assumed mounted to the vehicle facing front, so we do not have to generate any features in the ceiling or on the ground.

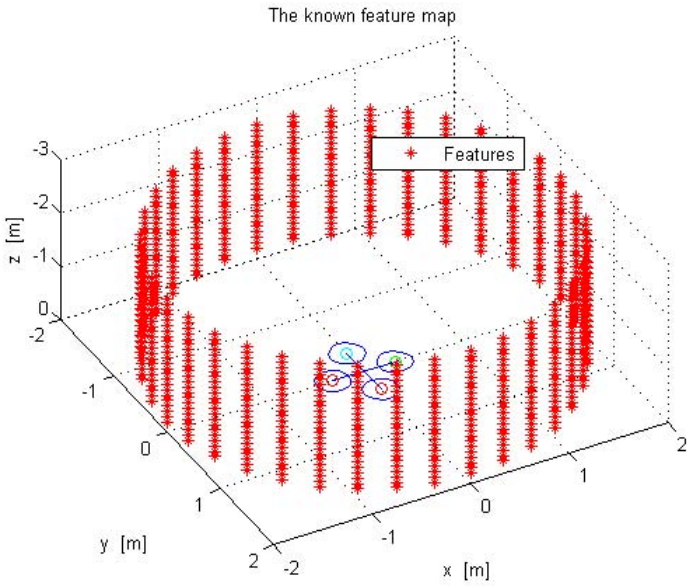


Figure 4.3: Generated known map for the map-based visual-inertial navigation

Case 1

We assume the initial position of the quadrotor vehicle is placed in the origin of the navigation frame N. We first simulate the case of the quadrotor taking off. The vehicle prepares statically for initialization on the ground surface. This takes 15 seconds. After that, the quadrotor vehicle takes 5 seconds to rise up off the ground surface to a height of 1.5 m. The final phase for the vehicle is to stay still in the air for 20 seconds. The quintic and cubic splines are used to interpolate values to the positions and attitudes during the transition. The calibration between the camera frame C and the body frame B is assumed with zero translation and a fixed rotation matrix in a way that the camera is mounted facing front. The reference bias is assumed to be constant and corrupted by additive white noise vectors. The constant parts of the accelerometer and gyro biases are given as $[0.1; 0.1; 0.1]$ and $[0.02; 0.02; 0.02]$, respectively. The state estimation results are shown in the following figures.

Case 2

The next estimation case study involves the estimation for quadrotor vehicle

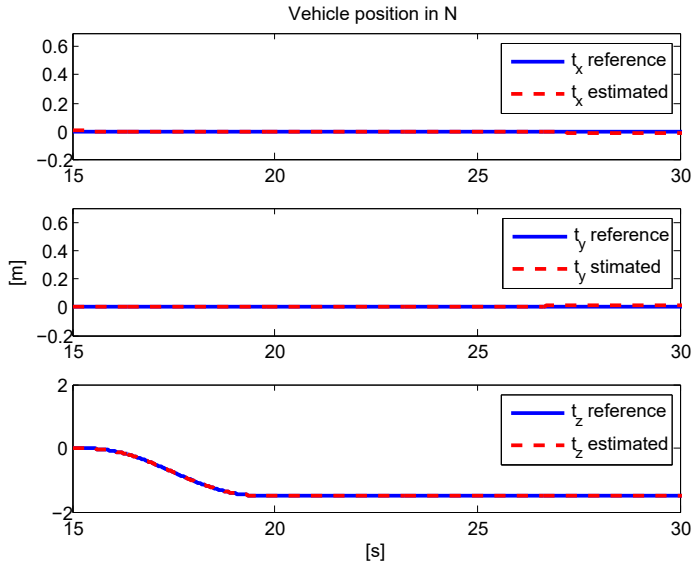


Figure 4.4: Reference and estimated positions for taking off

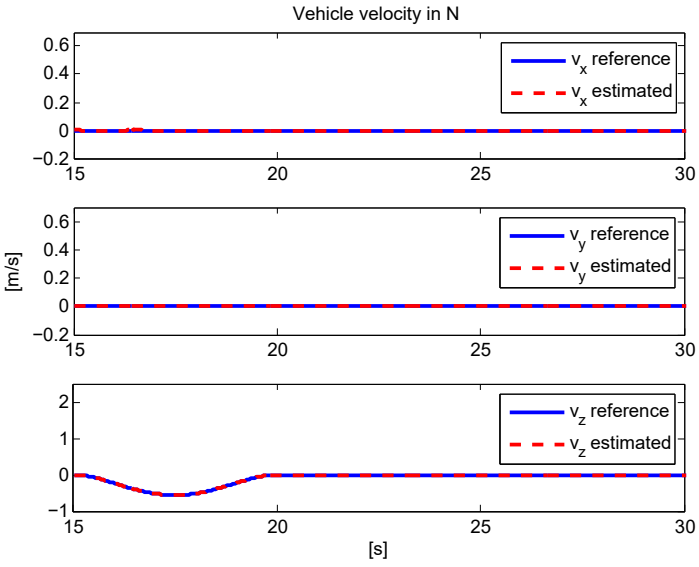


Figure 4.5: Reference and estimated velocities for taking off

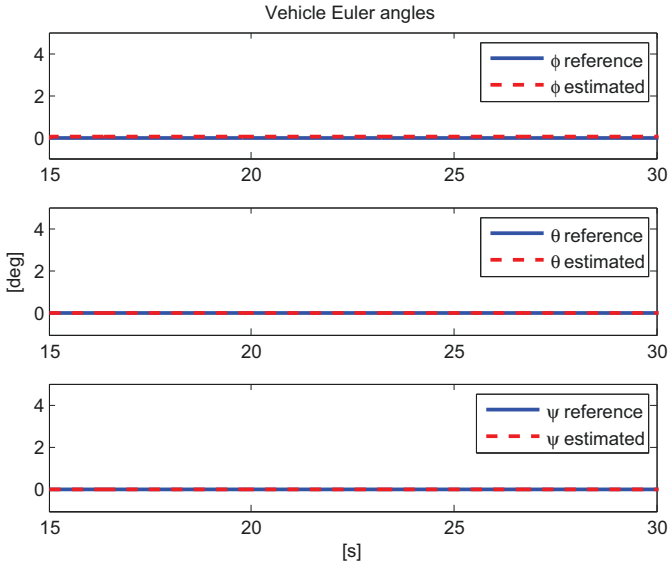


Figure 4.6: Reference and estimated Euler angles for taking off

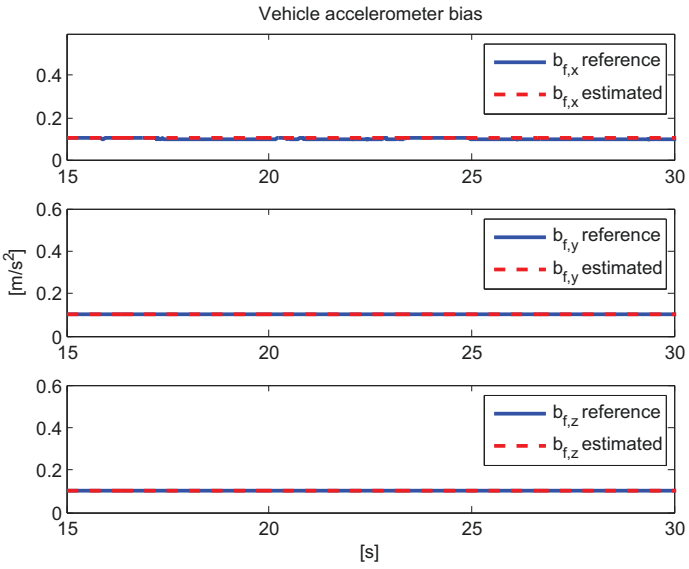


Figure 4.7: Reference and estimated accelerometer biases for taking off

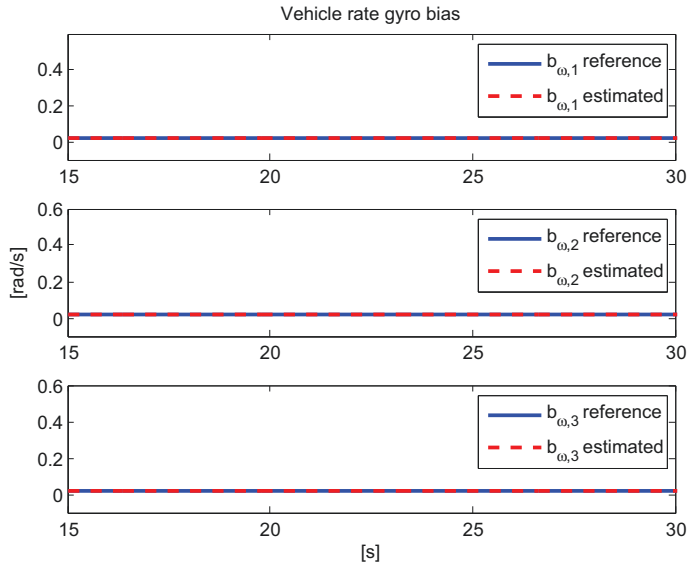


Figure 4.8: Reference and estimated gyro biases for taking off

hovering. The recognizable visual features are generated in the same way as the one for the quadrotor taking off. After it takes off at $t = 20$ seconds, the quadrotor vehicle turns counterclockwise 90 degrees, following the reference trajectory. In this transition phase, the quadrotor vehicle also shifts to a new point of $t_x = 0.5$ and $t_y = 0.5$ in N, while maintaining the height of $t_z = -1.5$. This transition lasts 10s and then the quadrotor vehicle is assumed to stay still for 5s at that final point. The camera configuration is the same as in the first case. The results of this case study are given in the following figures.

The map-based method can yield acceptable results, because the aiding features are known in 3D world. From the figures, we see all the bias terms are shown as small constant scalars. The hovering result of position estimation has offsets, but they are small which still shows the effectiveness of the position aiding .

4.9.2 Map-Less Visual-Inertial Navigation Results

In this section, we provide simulation results of the quadrotor vehicle states estimation for the map-less visual-inertial navigation system. As the vehicle takes off, the state is propagated from the input from the IMU element and

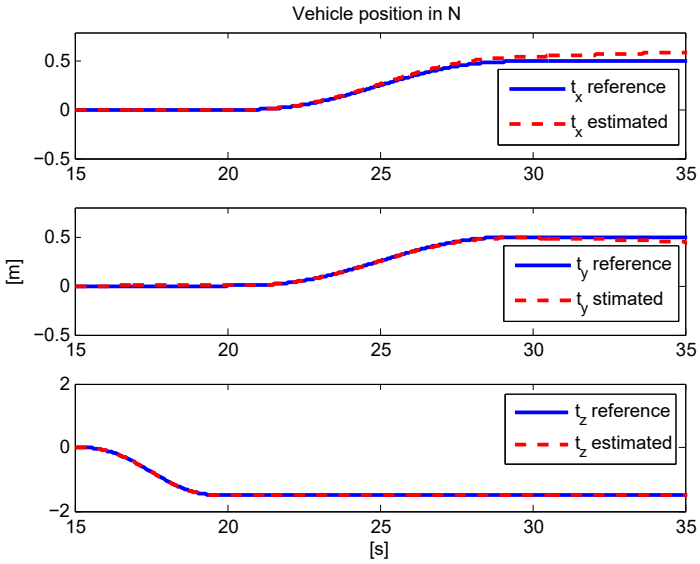


Figure 4.9: Reference and estimated positions for hovering

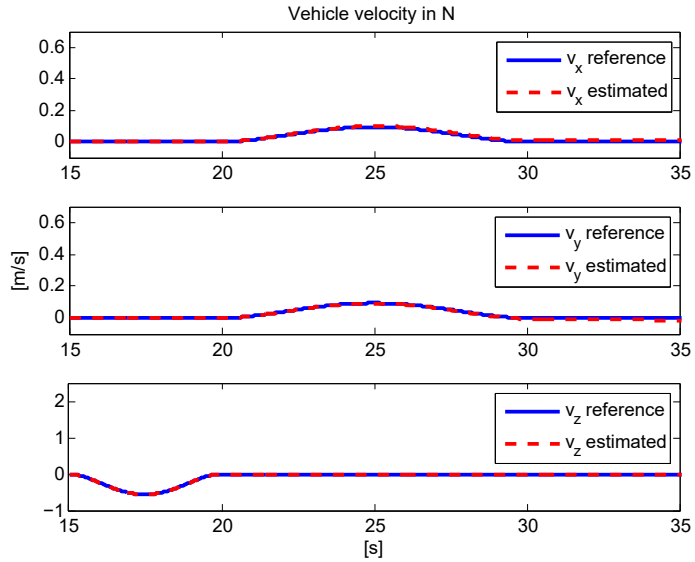


Figure 4.10: Reference and estimated velocities for hovering

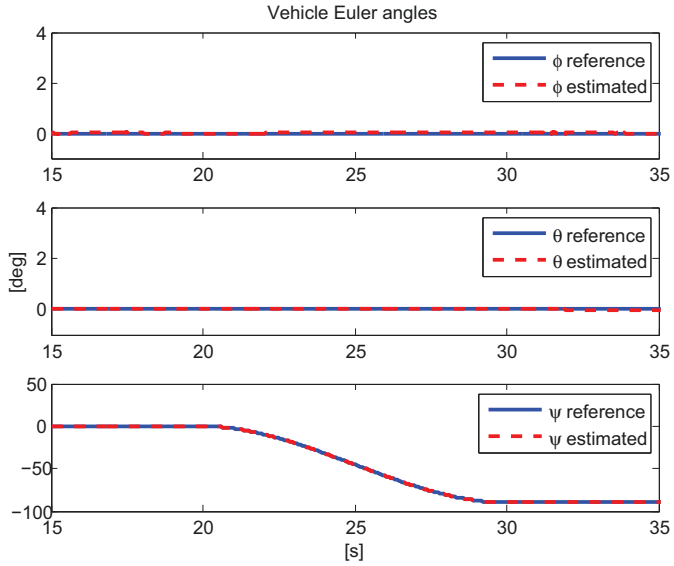


Figure 4.11: Reference and estimated Euler angles for hovering

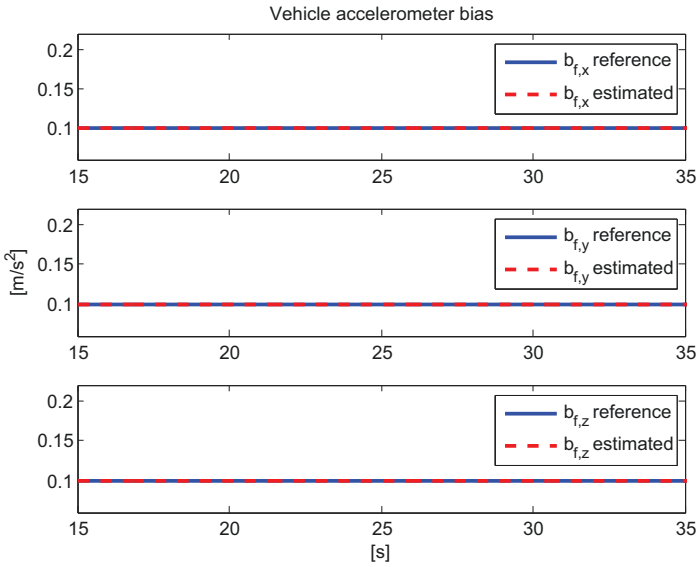


Figure 4.12: Reference and estimated accelerometer biases for hovering

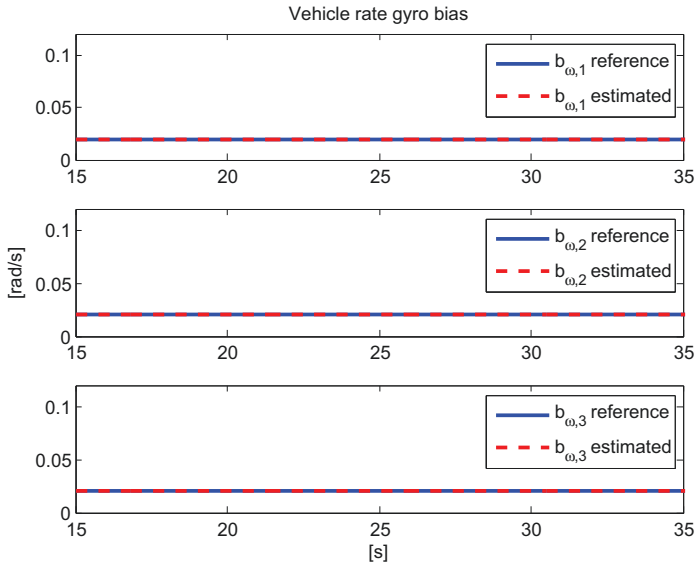


Figure 4.13: Reference and estimated gyro biases for hovering

the EKF uses the visual aiding information to correct the estimated state. The camera is mounted at the vehicle center of mass, i.e., the origin of B, and is facing front. The bias terms are assumed constant vectors. This navigation system design does not assume that we have a pre-built feature map with known feature coordinates. We assume that adequate random features are distributed in the indoor environment and it is feasible to recognize, extract and match them. As this design uses the vision algorithms to process visual information, it stores the feature coordinates from two adjacent frames and uses the epipolar geometry to generate the scaled pose aiding measurement. The initial pose pair is set to be $(I,\ 0)$.

Assume pseudorandom features are distributed in the surrounding environment. For example, features are assumed distributed as in Fig. 4.3, but we do not know their 3D coordinates in the navigation frame N. The vision algorithm such as the speeded up robust features (SURF) method is used to detect the features and keeps tracking them in every camera frame snapshot during the quadrotor taking off and hovering. The features from two consecutive image frames are then matched. The matched points are illustrated in Fig. 4.14, where the detected features are denoted as red circles. In Fig.

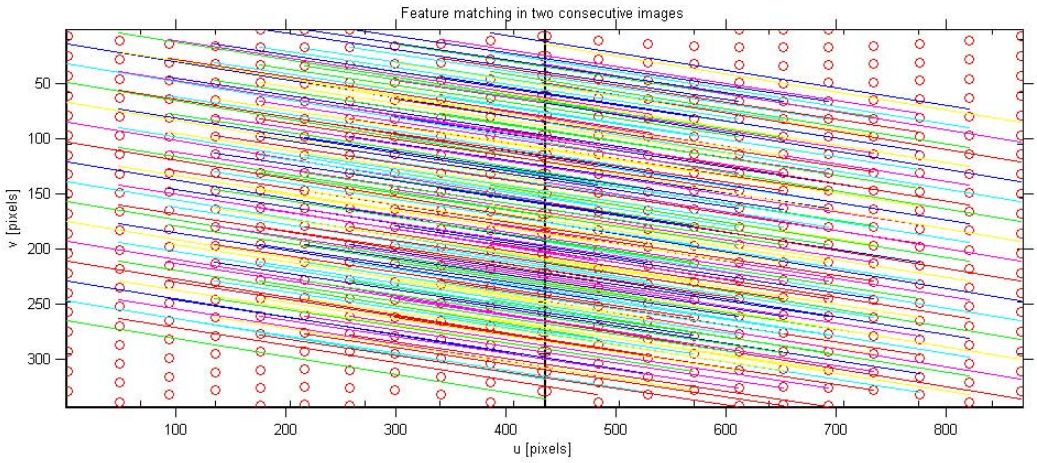


Figure 4.14: Feature matching between two frames taken from adjacent poses

4.14, two images are taken when the quadrotor reaches different heights in the case of taking off. Features are detected then. Both images catch more than 200 features. The matching algorithm returns 180 matched pairs of all the detected features. The coordinates of these matches pairs are recorded as $p_a = [x_1, y_1; x_2, y_2; \cdots]$ and $p_b = [x_1^{'}, y_1^{'}; x_2^{'}, y_2^{'}; \cdots]$. There can be mismatches of features in the two images. The vision algorithms eventually select the eight most significant features and their matches image coordinates pairs, because significant features renders fewer mismatches of their coordinates. Selecting fewer features also reduces the computational burden. By using the eight-point algorithm presented in Chapter 3, the fundamental matrix F_{ab} and the essential matrix E_{ab} are computed numerically, where the MATLAB robot vision control (RVC) toolbox [8] is used.

After the essential matrix is obtained, the scaled pose aiding measurement vector is generated from the scale relative pose pair (R_{ab}, t_{ab}), which is decomposed from the formula of $E_{ab} = [t_{ab}]_{\times} R_{ab}$. The false solution pair is eliminated using the method presented in Chapter 2.

Case 1

The initialization takes around 15 seconds when the initial conditions are de-

termined. Next, the vehicle starts taking off and it spends 10 seconds to reach the hover point at $[0\ 0\ -1.5]^T$ in N where the quadrotor vehicle hovers for 5 seconds. The constant accelerometer and gyroscope biases are set as in the map-based design method. The absolute scale is set as 0.5. The vehicle state estimation results are given in the following figures.

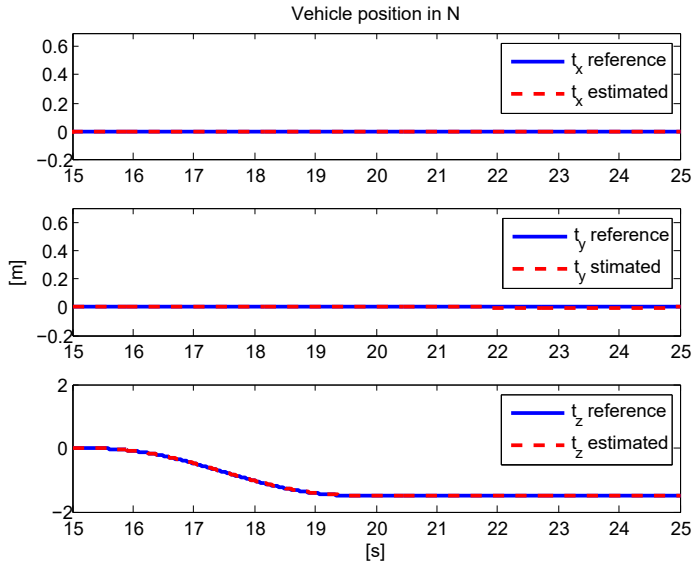


Figure 4.15: Reference and estimated positions for taking off

From the result figures, we can see that the state variables such as the positions, velocities and orientations can track the references. There is no end drift in the x and y positions. The bias values stay close to constants as expected. However, the absolute scale has an offset between the reference and its estimation. As the IMU provides the estimation in the real world scale, the fact that it is not possible to correct the absolute scale from the visual aiding measurement does not ruin the estimation results of other state estimations.

Case 2

The next estimation case study is the estimation for quadrotor vehicle hovering. The visual scaled pose measurements are obtained using the vision algorithms described in the first case in this map-less design. After the vehicle

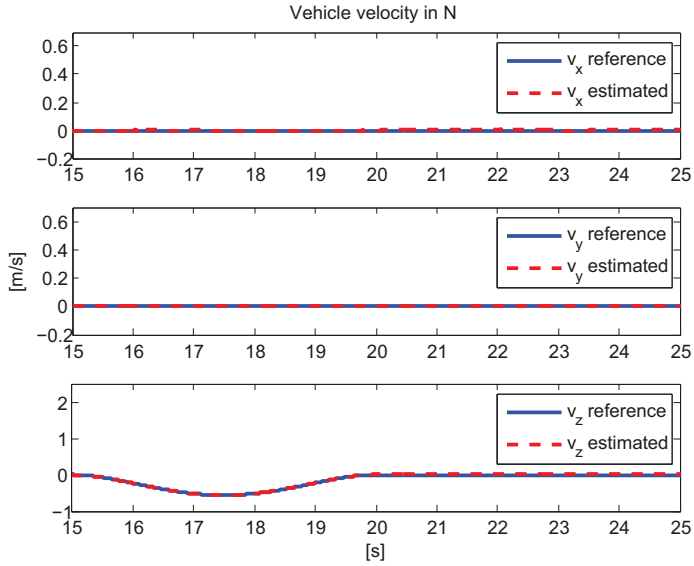


Figure 4.16: Reference and estimated velocities for taking off

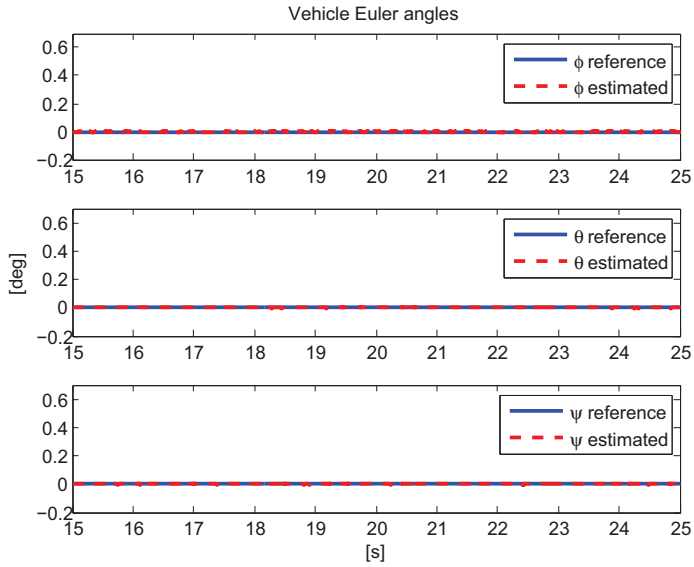


Figure 4.17: Reference and estimated Euler angles for taking off

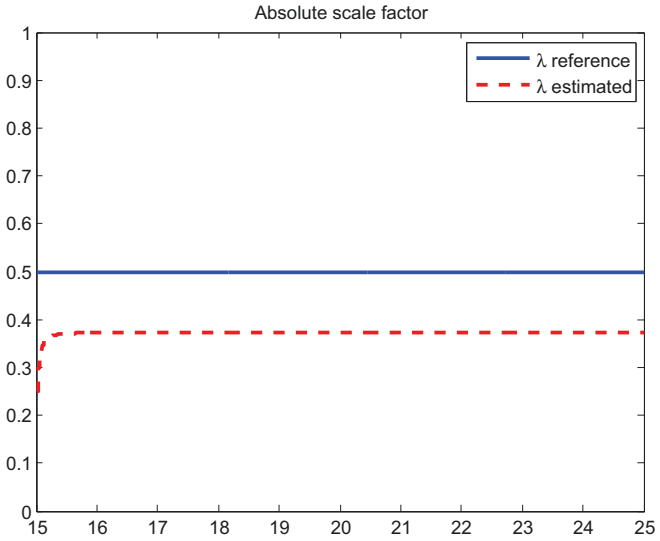


Figure 4.18: Reference and estimated absolute scale factor for taking off

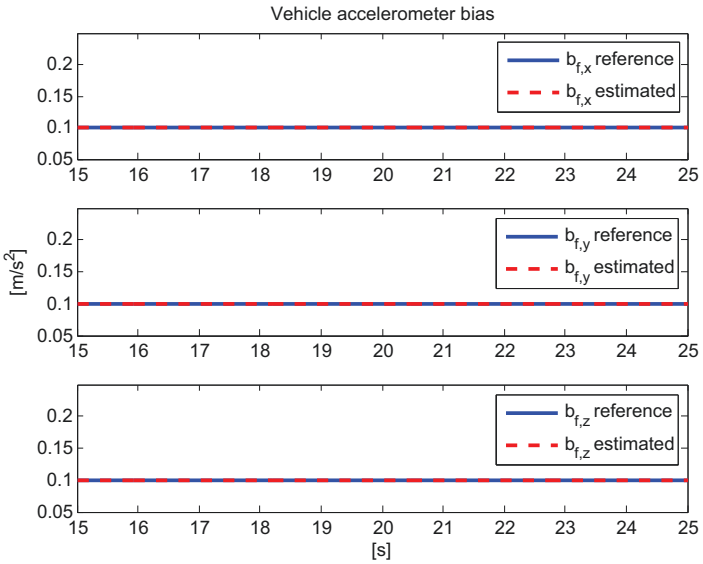


Figure 4.19: Reference and estimated accelerometer biases for taking off

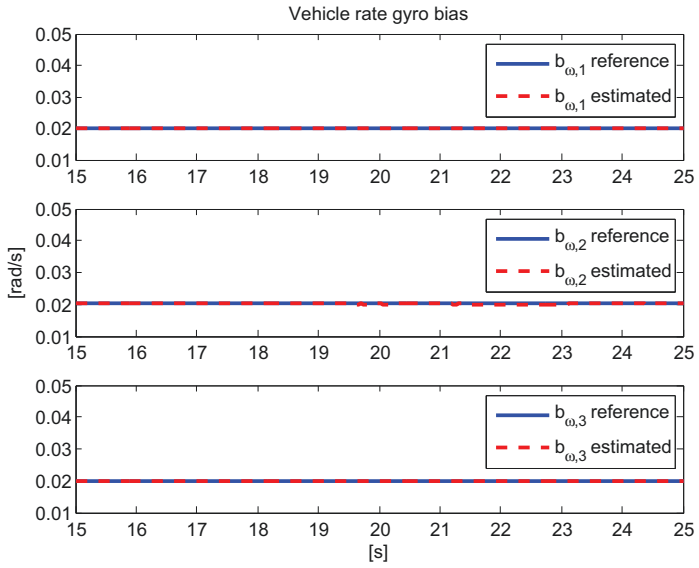


Figure 4.20: Reference and estimated gyro biases for taking off

takes off, the quadrotor vehicle turns counterclockwise 90 degrees, following the reference trajectory. In this transition phase, the quadrotor vehicle shifts to a new point of $t_x = 0.5$ and $t_y = 0.5$ in N, while maintaining a height of $t_z = -1.5$. The accelerometer bias is set as $[0.01; 0.01; 0.01]$. The rate gyroscope bias is set to be $[0.02; 0.02; 0.02]$. This transition spends 5 seconds and then the quadrotor vehicle is assumed to stay still for 5 seconds at that final point. The camera configuration is the same as in the first case. The vehicle state estimation results are given in the following figures.

If no aiding measurement is available, the position estimation integrated from the IMU measurements will drift significantly. There are offsets for the end values of the position estimates. This occurs because the scale estimation is not perfect. The initial guess of the absolute scale is 0.6. However, this value is not corrected much after the filter loop implementation. The inherent drift of the position estimation is rooted in the double integration to obtain the value of it. The results show that the velocity estimation remain stable while the quadrotor is hovering from one point to another. This result shows the effectiveness of the proposed VI navigation system.

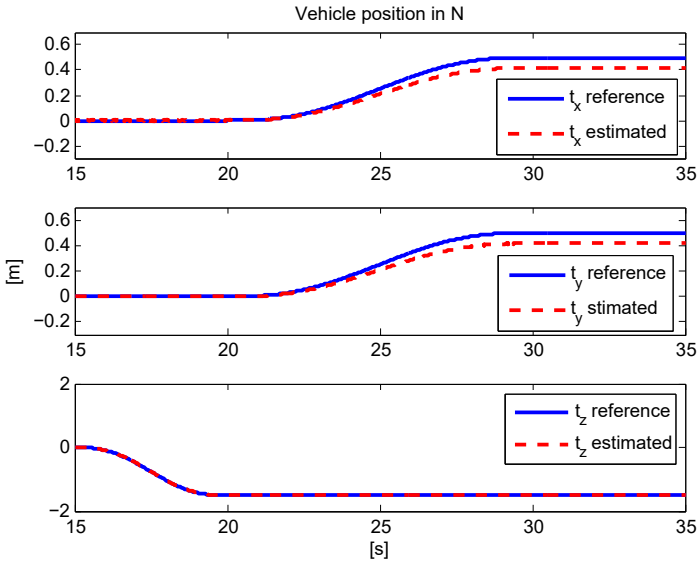


Figure 4.21: Reference and estimated positions for hovering

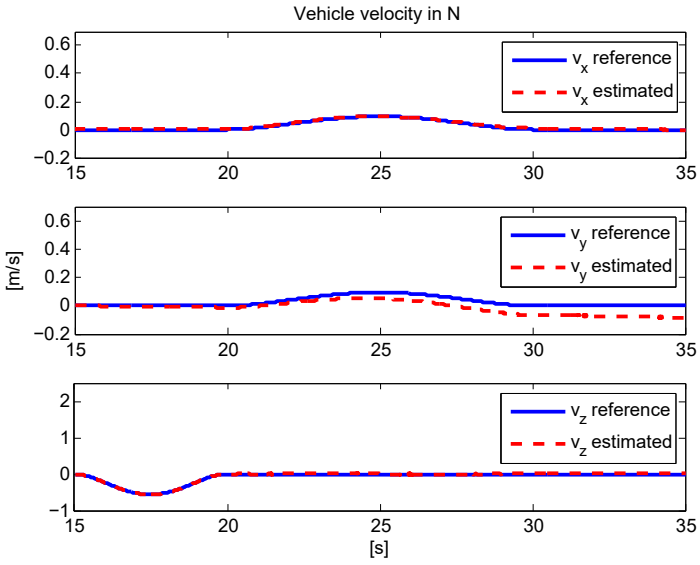


Figure 4.22: Reference and estimated velocities for hovering

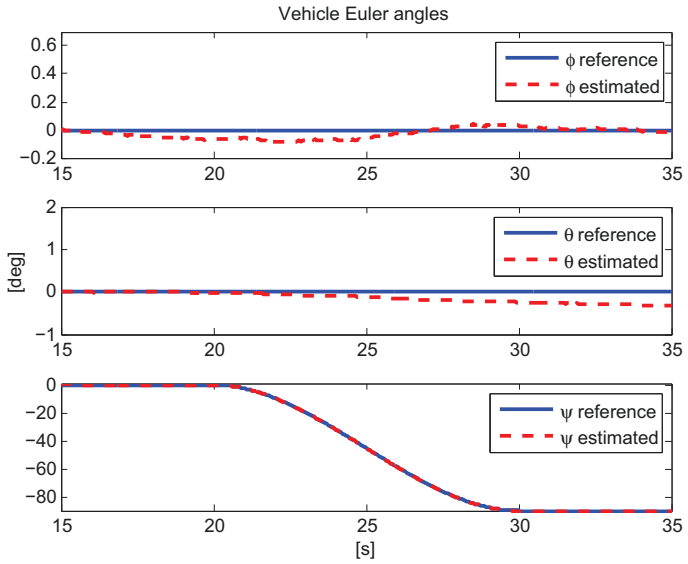


Figure 4.23: Reference and estimated Euler angles for hovering

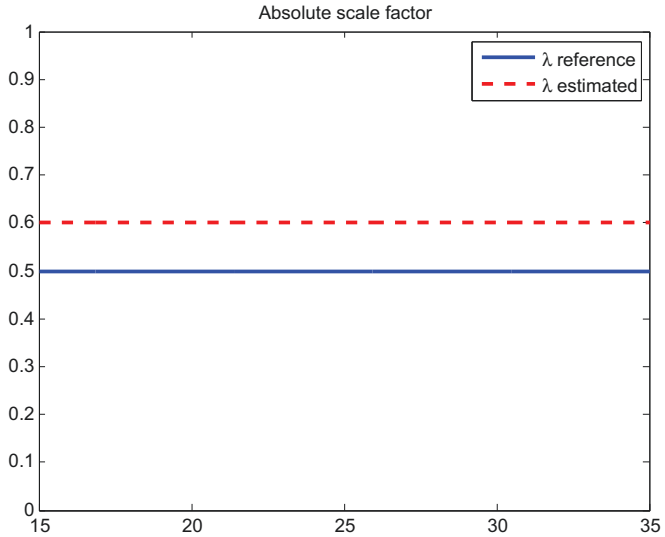


Figure 4.24: Reference and estimated absolute scale factor for hovering

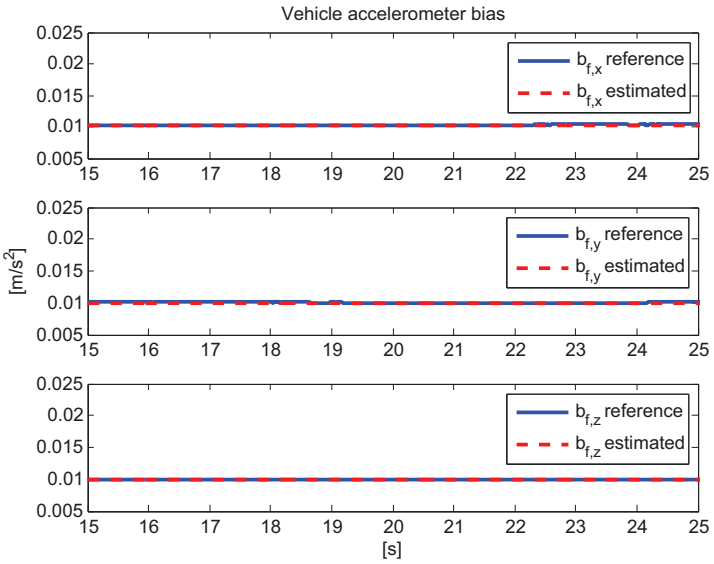


Figure 4.25: Reference and estimated accelerometer biases for hovering

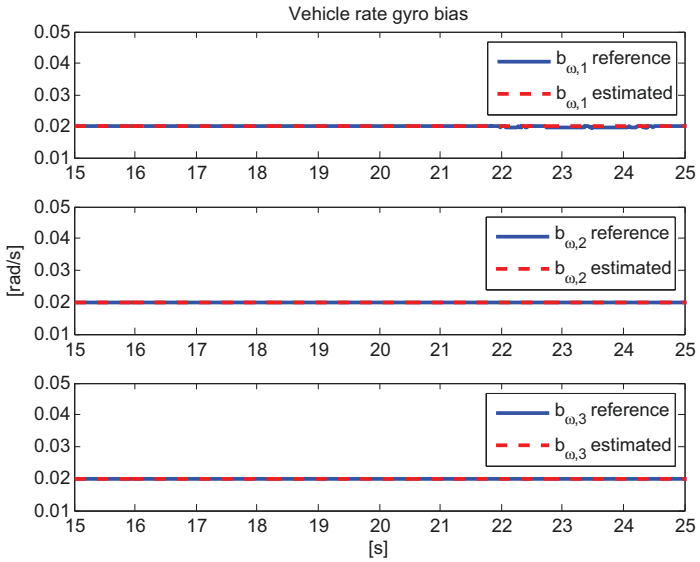


Figure 4.26: Reference and estimated gyro biases for hovering

Chapter 5

Conclusion

This chapter summarizes the thesis and provides direction for possible future work.

5.1 Summary of Thesis Work

The work presented in this thesis focuses on developing a vision-aided navigation system for a quadrotor in an indoor environment. From a practical point of view, we focus on the design using monocular vision. The sensors of the perspective camera platform, Raspberry Pi and the inertial measurement unit (IMU), PX4FMU are each modeled. The visual odometry (VO) algorithm based on epipolar geometry is proposed for the Raspberry Pi camera. The local window bundle adjustment algorithm is used to refine the visual aiding measurement.

In order to design a vehicle state estimation method with constraint onboard computation ability, the loosely coupled sensor fusion method is employed to combine the visual and inertial sensor blocks. The visual and inertial navigation blocks are running independently. The visual navigation block uses the VO design to store the scales incremental pose trajectories rather than to build any type of map, as in the simultaneous localization and mapping (SLAM) method.

There are two visual-inertial navigation systems designed in this thesis, which one will be used depends on whether a known feature map is given. The observability analysis is provided for both navigation systems. The ob-

servability analysis shows that the proposed navigation systems are locally observable. The map-based visual-inertial navigation system and the map-less visual-inertial navigation system are simulated. The results show that the designed navigation systems can effectively estimate the quadrotor vehicle states with the modeling parameters from the practical sensors.

5.2 Directions of Future Work

This thesis proposes an integrated navigation system for the quadrotor vehicle using the fusion of IMU and VO. Based on the current work and literature reviews, several possible directions for future research work and implementation refinements are given.

5.2.1 Optical Flow-Based VI Navigation System Design

The vehicle state estimation can be completed from the optical flow (OF) using decomposition of the homography matrix and fusion with the IMU sensor. Using the optical flow has an advantage in that it provides the camera motion information directly and can avoid drift effects in the incremental visual pose measurements.

The optical flow method is used in cases where the robot vehicle is going to travel a long distance and the camera frames can update at a high rate of speed. This is in contrast to incremental VI odometry state estimation where camera motion is assumed to be finite. Assume a world point as X and the scaled estimation of it as x. Introducing the scale factor λ gives the relationship $X = \lambda x$. We thus have the kinematic

$$\dot{X} = \dot{\lambda} x + \lambda v_x \tag{5.1}$$

where v_x is the point linear velocity measurement from the image. From the relationship involving the homography matrix H $\dot{X} = HX$ we have

$$v_x = Hx - \frac{\dot{\lambda}}{\lambda} x \tag{5.2}$$

Removing the depth dynamic yields

$$[x]_\times Hx = [x]_\times \tag{5.3}$$

which is the continuous homography constraint [34]. From (5.3) we can see by stacking the reconstructed point and the measured velocity, that the homography can be obtained through a numerical estimation process.

In [19], optical flow has been integrated with the IMU block for the quadrotors to perform a particular flight task of vertical landing on a textured target. The optical flow algorithm is not performed onboard. This system design involves measuring and holding a constant rate of image expansion as the surface target is approached. However, it works best only if the quadrotor approaches the ground vertically, which makes it easy to create optical flow.

5.2.2 Vehicle States Estimation Using Trifocal Tensor

The uncalibrated visual servoing using the three-view (i.e., initial, current and target views) is explored in [43]. The numerical constraint of the three-view geometry is the trifocal tensor, which is independent of the scene structure and depends only on the relationship between the cameras [18].

In [23], a vision-aided navigation system is proposed using three view geometry. The vision block develops a trifocal tensor constraint, which is fused with the inertial IMU block through an implicit extended Kalman filter (IEKF) loop to generate vehicle state estimation. The work presented in [49] and [13] uses three-view geometry constraints as well but in a unscented Kalman filter (UKF) version. The advantage of using three-view geometry is it needs less computation compared to SLAM and VO methods which use windowed bundle adjustments.

Bibliography

[1] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, Inc., 2002.

[2] Martin Barczyk. *Nonlinear State Estimation and Modeling of a Helicopter UAV*. PhD thesis, University of Alberta, 2012.

[3] M. Bergamasco and M. Lovera. Identification of linear models for the dynamics of a hovering quadrotor. *Control Systems Technology, IEEE Transactions on*, 22(5):1696–1707, Sept 2014.

[4] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[5] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13(4):82–90, Dec 2006.

[6] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, third edition edition, 1999.

[7] S.Y. Chen. Kalman filter for robot vision: A survey. *Industrial Electronics, IEEE Transactions on*, 59(11):4409–4420, Nov 2012.

[8] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 73 of *1610-7438*. Springer-Verlag Berlin Heidelberg, 2013.

[9] J.L. Crassidis. Sigma-point kalman filtering for integrated gps and inertial navigation. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(2):750–756, April 2006.

[10] AJ. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, Oct 2003.

[11] A.J. Davison and D.W. Murray. Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):865–880, Jul 2002.

[12] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, June 2001.

[13] Q. Fang and S.X. Huang. Ukf for integrated vision and inertial sensors based on three-view geometry. *Sensors Journal, IEEE*, 13(7):2711–2719, July 2013.

[14] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill Professional, 2008.

[15] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90, June 2012.

[16] Vladislav Gavrilets. *Autonomous Aerobatic Maneuvering of Miniature Helicopters*. PhD thesis, Massachusetts Institute of Technology, May 2003.

[17] Volker Grabe, Heinrich H. Blthoff, Davide Scaramuzza, and Paolo Robuffo Giordano. Nonlinear ego-motion estimation from optical flow for online control of a quadrotor uav. *The International Journal of Robotics Research*, 34(8):1114–1135, 2015.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[19] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. *Robotics, IEEE Transactions on*, 28(1):77–89, Feb 2012.

[20] J.A. Hesch, D.G. Kottas, S.L. Bowman, and S.I. Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. *Robotics, IEEE Transactions on*, 30(1):158–176, Feb 2014.

[21] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 33(1):182–201, 2014.

[22] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. Observability-based rules for designing consistent ekf slam estimators. *The International Journal of Robotics Research*, 29(5):502–528, 2010.

[23] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. Real-time vision-aided localization and navigation based on three-view geometry. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(3):2239–2259, JULY 2012.

[24] Gijeong Jang, Sungho Kim, and Inso Kweon. Single camera catadioptric stereo system. In *The 6th Workshop on Omnidirectional Vision, Camera Networks and Non-classical cameras*, 2005.

[25] Eagle S. Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, 2011.

[26] David R. Kastelan. Design and implementation of a gps-aided inertial navigation system for a helicopter uav. Master's thesis, University of Alberta, 2009.

[27] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[28] Jonghyuk Kim and Salah Sukkarieh. Real-time implementation of airborne inertial-slam. *Robotics and Autonomous Systems*, 55(1):62 – 71, 2007.

[29] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[30] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visualinertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[31] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visualinertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[32] Mingyang Li and Anastasios I. Mourikis. Vision-aided inertial navigation with rolling-shutter cameras. *The International Journal of Robotics Research*, 33(11):1490–1507, 2014.

[33] Mingyang Li, Hongsheng Yu, Xing Zheng, and A.I. Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 409–416, May 2014.

[34] Yi Ma, Stefano Soatto, Jana Koseck, and S. Shankar Sastry. *An Invitation to3-D Vision: From Images to Geometric Models*, volume 26. Springer-Verlag New York, 2004.

[35] A. Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *Robotics, IEEE Transactions on*, 28(1):44–60, Feb 2012.

[36] A.I. Mourikis and S.I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572, April 2007.

[37] Richard A. Newcombe. *Dense Visual SLAM*. PhD thesis, Imperial College London, n Computing of Imperial College London, 2012.

[38] D. Nister. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):756–770, June 2004.

[39] David Nistr, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[40] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92, Dec 2011.

[41] K. Schmid, F. Ruess, M. Suppa, and D. Burschka. State estimation for highly dynamic flying systems using key frame odometry with varying time delays. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2997–3004, Oct 2012.

[42] Korbinian Schmid, Philipp Lutz, Teodor Tomi, Elmar Mair, and Heiko Hirschmller. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, 31(4):537–570, 2014.

[43] Azad Shademan. *Uncalibrated Vision-Based Control and Motion Planning of Robotic Arms in Unstructured Environments*. PhD thesis, University of Alberta, Edmonton, Alberta, 2012.

[44] Malcolm D. Shuster. Constraint in attitude estimation part i: Constrained estimation. *The Journal of the Astronautical Sciences*, 51(1):51–74, 2003.

[45] Malcolm D. Shuster. Constraint in attitude estimation part ii: Unconstrained estimation. *The Journal of the Astronautical Sciences*, 51(1):75–101, 2003.

[46] Nikolas Trawny, Anastasios I. Mourikis, Stergios I. Roumeliotis, Andrew E. Johnson, and James F. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, 2007.

[47] Bill Triggs, Philip F, McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. *Vision Algorithms: Theory and Practice*, volume 1883, chapter Bundle Adjustment A Modern Synthesis, pages pp 298–372. Springer Berlin Heidelberg, 2000.

[48] E.A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[49] Sen Wang, Ling Chen, Dongbing Gu, and Huosheng Hu. Vision-aided inertial navigation using three-view geometry. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pages 946–951, June 2014.

[50] S. Weiss, M.W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964, May 2012.

[51] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Michael C. Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, 30(5):803–831, 2013.

[52] Stephan M. Weiss. *Vision Based Navigation for Micro Helicopters*. PhD thesis, ETH ZURICH, 2012.