Camera-Based Integrated Indoor Positioning

Ylva Stokke Vintervold

Master of Science in Engineering Cybernetics
Submission date: June 2013
Supervisor: Lars Imsland, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

PROJECT DESCRIPTION SHEET

Name of the candidate:	Ylva Stokke Vintervold
Thesis title (English):	Camera-based integrated indoor positioning

Background

As a possible testbed for UAV guidance and estimation algorithms (for instance for testing algorithms for estimation of ice properties in Arctic regions), it is proposed to use the Parrot AR.Drone 2.0 quadcopter in an indoor lab setup. This will require an accurate positioning algorithm for this UAV, which is the topic of this project. The algorithm should be based on a camera-based positioning system, integrated with inertial sensor measurements on-board the quadcopter.

Work description

1. Give a brief overview of methods for indoor positioning, and their relative strengths and weaknesses.
2. Describe camera-based indoor positioning in general, and the specific solution chosen.
3. Implement a Kalman filter for position, orientation and velocity estimation based on measurements from the camera-based system.
4. Implement an integrated solution using also the quadcopter's IMU measurements.
5. Discuss merits, challenges and limitations of the chosen approach, and make proposals for future work.

Start date: 14 January, 2013 **Due date:** 10 June, 2013

Supervisor: Lars Imsland
Co-advisor(s):

Trondheim, __24. January 2013__________

[illegible]

Lars Imsland
Supervisor

Address	**Org.no.** 974 767 880	**Location**	**Phone**
Sem Sælandsvei 5	E-mail:	O.S. Bragstads plass 2D	+ 47 73 59 43 76
NO-7491 Trondheim	postmottak@itk.ntnu.no	NO-7034 Trondheim	**Fax**
	http://www.itk.ntnu.no		+ 47 73 59 45 99

Phone: + 47 47 23 19 49

Preface

This thesis concludes my (long and winding) path towards becoming a Master of Science in the field of Engineering Cybernetics at the Norwegian University of Science and Technology.

First and foremost, I would like to express my gratitude to my supervisor Lars Imsland for providing helpful discussions and advice. His inputs, both theoretical, practical and in the writing of this thesis, have been greatly appreciated. I am also grateful to Joakim Haugen for the help and guidance he has provided throughout this thesis.

Furthermore, Christian Holden and Pål Liljebäck deserve a thank you for allowing me to use the new lab, and helping me with practical problems.

The friends I've made while studying deserve a thank you as well. My life in Trondheim would not have been the same without Hedvik, Hilde, Marianne, Signe, Kristian and Øystein, who have filled the past five years with good times. You guys are great!

Finally, Johan deserves my gratitude for his unconditional support and understanding, and for always believing in me.

Abstract

This thesis is motivated by the use of unmanned aerial vehicles for obtaining measurements of the scene in a system for estimation of ice properties. Such systems are needed to ensure safety when conducting marine operations in Arctic seas. As a possible testbed for UAV guidance and estimation algorithms, the use of the Parrot AR.Drone 2.0 quadcopter has been proposed. As a consequence, a positioning algorithm for the UAV is necessary, which is the topic of this thesis.

The aim of this thesis is to implement a positioning algorithm applicable to the proposed lab setup. As a consequence, challenges such as the time aspect as well as measurement loss and receiving outlier measurements for unknown periods of time are directed.

A camera-based positioning system serves as the main measurement source in this thesis. The system delivers position and orientation measurements based on marker tracking through the use of cameras placed along the ceiling in a lab setup. The measurements are subsequently used by two implemented positioning algorithms.

The Camera Measurement Algorithm uses measurements from the camera system to estimate the position, orientation, linear and angular velocities of the Parrot AR.Drone 2.0, while the Integrated Camera System/INS Algorithm additionally exploits inertial measurements from the UAV to estimate its position, orientation and linear velocity as well as inertial sensor biases.

The algorithms both utilize extended Kalman filters to perform state estimation, while the integrated algorithm also makes use of the sensor fusion feature of the state estimator. Both algorithms are tested online in the lab setup, and their applicabilities are, to some extent, validated. That is, no ground truths are available in the online tests, and simulations are performed to validate the accuracies of the state estimates. The algorithms have not yet been used as part of a larger motion control system, and their performances cannot be completely verified. With the goal of this thesis in mind, the Camera Measurement Algorithm obtained the best results. However, further development of the Integrated Camera System/INS Algorithm may lead to a different conclusion.

Sammendrag

Den overordnede bakgrunnen for denne hovedoppgaven er den økende bruken av ubemannede fly. Slike farkoster kan benyttes for å innhente nødvendige målinger i et system for beregning av isegenskaper, som er en nødvendig del av marine operasjoner i arktiske strøk for å kunne ivareta sikkerheten på en god måte. Kvadrokopteret Parrot AR.Drone 2.0 har blitt foreslått som del av et innendørs testsystem for ulike algoritmer i forbindelse med slike systemer. En innendørs posisjoneringsalgoritme er i så måte nødvendig for å kunne styre kvadrokopteret til ønskede deler av rommet. Denne posisjoneringsalgoritmen er temaet for denne oppgaven.

Målet med oppgaven er å implementere en posisjoneringsalgoritme som fungerer godt i et overordnet system for styring av kvadrokopteret. Fordi algoritmen skal brukes i et virkelig system, må det tas hensyn til forskjellige feil som kan inntreffe, f.eks. kan målinger utebli, eller de kan inneholde feil. I tillegg er det viktig at tilstandsestimatene oppdateres kontinuerlig.

Et kamerabasert posisjoneringssystem benyttes som kilde til målinger i denne oppgaven. Systemet leverer posisjons- og orienteringsmålinger basert på tracking av markører, og disse benyttes av to implementerte posisjoneringsalgoritmer.

Den første algoritmen benytter data fra det kamerabaserte posisjoneringssystemet til å estimere posisjon og orientering samt lineære- og vinkelhastigheter, mens den andre i tillegg benytter målinger fra IMUen ombord på UAVen til å estimere posisjon, orientering og lineær hastighet samt sensorbiaser.

Begge algoritmene bruker et extended Kalman filter for tilstandsestimering, mens den integrerte algoritmen også utnytter sensor fusion-funksjonen til denne tilstandsestimatoren. Begge algoritmene er implementert for bruk i testsystemet, men det kan ikke fastslås med sikkerhet at de fungerer optimalt fordi de ikke er blitt testet som del av et større system ennå. Algoritmene er i tillegg simulert for å undersøke nøyaktigheten i tilstandsestimatene, og de er funnet tilfredsstillende. Den første algoritmen oppnådde de beste resultatene i denne hovedoppgaven, men ved videre utvikling er det ikke unaturlig om den andre oppnår forbedret funksjonalitet.

Contents

Nomenclature

AP	Access Point
API	Application Programming Interface
DLL	Dynamic-Link Library
DOF	Degrees of Freedom
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
EKF	Extended Kalman Filter
FOV	Field of View
FPS	Frames Per Second
GNC	Guidance, Navigation and Control
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
IPS	Indoor Positioning System
IR	Infrared
ISA	Inertial Sensor Assembly
LED	Light-Emitting Diode
NED	North-East-Down
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
WLAN	Wireless Local Area Network

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Testbed for Estimation of Ice Properties in Arctic Areas

The motivation for the assignment given in this thesis is the entering of off-shore oil and gas production into Arctic seas, and the challenges that arise when conducting marine operations in such areas. For safe operation in ice-infested environments, knowledge of the nature of the surroundings is crucial. As a consequence, a system for detection of ice and estimation of its properties is needed.

In a system for estimation of ice properties, the use of *Unmanned Aerial Vehicles* (UAVs) for obtaining the required measurements of the scene has been proposed. Such crafts can be remotely operated or autonomous, and are thus suitable for mapping the area of interest. It follows that navigation as well as guidance and control systems need be part of an overall system for directing the UAV towards interesting areas.

As a possible testbed for UAV guidance and estimation algorithms, an indoor lab setup consisting in part of the Parrot AR.Drone 2.0 quadcopter has been proposed. In particular, the testing of algorithms for estimation of ice properties in Arctic environments using measurements obtained by UAVs is intended. The testbed will require a motion control system for commanding the quadcopter in the lab setup, similarly to the full-scale setup outlined above. In [4], motion control systems for large-area operation are divided into three separate blocks denoted as the guidance, navigation, and control (GNC) systems. A GNC system is depicted in Figure 1.1. The topic of this thesis is the development of an *Indoor Positioning System* (IPS) for use in the proposed lab setup. Such a system constitutes the Navigation block of Figure 1.1 when indoor operation is intended.

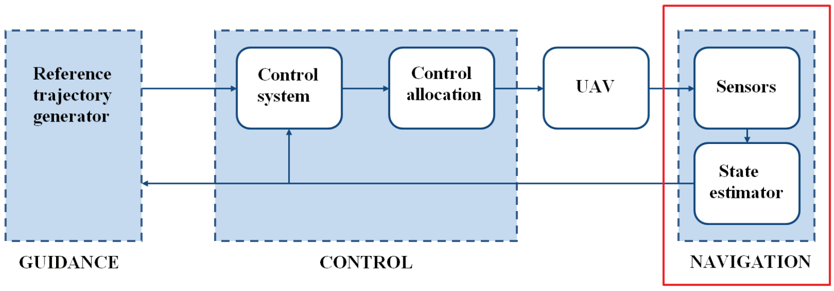

Figure 1.1: *Figure reproduced from [4]. A motion control system divided into guidance, navigation, and control subsystems is depicted. The right-most, framed block, Navigation, for an indoor lab setup is the topic of this thesis.*

1.2 Indoor Positioning System for Unmanned Quadcopter

The implementation of an accurate system for determining the position, orientation, and velocity of a small quadcopter is the assignment given in this thesis. In the full-scale system described above, GPS would be used for this purpose, possibly in combination with an *Inertial Navigation System* (INS), which utilizes inertial sensors to obtain the required measurements for estimation of the variables in question. An *indoor* positioning system is used to determine the pose of an object inside buildings (where GPS signals become unreliable), and there exist several different methods utilizing various sensor types which solve the indoor positioning problem.

Previous work within the field include vision-based methods, the use of wireless networks, and inertial sensor systems, to mention a few types of setups. Of special importance in this thesis is the integrated system proposed in [6], which combines the use of inertial sensors and a video system to produce position, orientation, and velocity estimates in a hand-tracking scenario. The measurement fusion is performed using an *Extended Kalman Filter* (EKF), and the obtained results include minimization of errors resulting from camera outages of short duration. A selection of methods for indoor positioning is presented and comparisons are made in Chapter 2, thus a further literature review is not given here.

The task of choosing a suitable indoor positioning system for the Parrot AR.Drone 2.0 was first investigated in the project work performed in the fall of 2012 [17]. A candidate method to solve the problem was chosen based on a literature review, and prototype implementation was performed. The chosen method used cameras as sensors, recognizing markers attached to the UAV based on color. Furthermore, the computer vision technique *stereo vision* was used to estimate the 3D pose of the UAV from the image positions of the color markers. Finally, an extended Kalman filter would use the world position of the quadcopter as well as its orientation as the required measurement input, and output position, orientation, and velocity estimates.

The prototype implementation in the project work was performed using simulated measurements, and for planar motion only. Thus, the project work was a simplified and simulated version of an indoor positioning system. In this thesis, a positioning algorithm based on measurements obtained by marker tracking from a camera-based positioning system is implemented. An extended Kalman filter estimates position, orientation, and velocity in all six Degrees of Freedom (DOF), as opposed to the system in the project work. Furthermore, the algorithm in this thesis is extended to include inertial measurements from the quadcopter, resulting in an integrated solution. Hence, this thesis intends to implement a system similar to the one that was simulated in the fall of 2012, and subsequently develop the positioning algorithm further.

1.3 Contribution of Thesis

The goal of this thesis is to implement an accurate positioning algorithm for the Parrot AR.Drone 2.0. The algorithm should be suitable for online use in the lab setup scenario described above. As part of the development of a sufficiently accurate solution for the intended testbed, two positioning algorithms are implemented for use with real sensors in the lab; The **Camera Measurement Algorithm**, which uses data from a camera-based positioning system as measurements, and the **Integrated Camera System/INS Algorithm**, which augments the first algorithm by exploiting the inertial measurements obtained by the *Inertial Measurement Unit* (IMU) attached to the UAV.

Challenges connected to implementing a system for use with real data in a real-life scenario as opposed to simulations (such as the time aspect as well as measurement loss and receiving outlier measurements for unknown periods of time) are directed. Although time did not allow complete real-time implementations, a "near real-time" approach using MATLAB is chosen, and its applicability for both algorithms in the lab setup is evaluated. Additionally, both positioning algorithms are tested using simulated as well as recorded (real) measurements to verify or reject the achieved performance against known ground truths, and compare the algorithms to each other using the exact same scenario.

The positioning results achieved for the two implemented algorithms are analyzed and compared in order to come to a conclusion regarding the applicability of each algorithm to the intended lab setup. That is, the differences between the two system setups and resulting pose estimates are compared with the lab setup in mind, discussing advantages and disadvantages of each solution.

1.4 Outline of Thesis

As explained earlier, this thesis is divided into two separate indoor positioning algorithms, but both algorithms utilize the same camera-based measurement system and share the use of an extended Kalman filter. As a consequence, the coinciding aspects are elaborated in preceding chapters as well as in the presentation of the first positioning algorithm, and subsequently updated according to a new system structure in the Integrated Camera System/INS Algorithm presentation.

The thesis is organized as follows:

The next chapter provides a brief overview of a selection of methods for indoor positioning, including a more detailed presentation of camera-based indoor positioning in general. Furthermore, Chapter 3 is dedicated to a presentation of the particular camera-based positioning system used as the main measurement source in this thesis. Before the two proposed positioning algorithms are presented, Chapter 4 provides an overview of the terminology and notation used throughout the thesis. Important vectors and definitions for the utilized system models are emphasized in this chapter.

The measurement vector from the camera-based positioning system is connected to the rest of the Camera Measurement Algorithm in Chapter 5, and the functionality, real-time challenges, mathematical modelling, and state estimation of the algorithm are presented. Chapter 6 provides a corresponding presentation of the Integrated Camera System/INS Algorithm including theoretical aspects connected to inertial measurements and measurement integration.

The results achieved by both positioning algorithms are presented in Chapter 7 and discussed, along with challenges and limitations, in Chapter 8, while conclusions with regards to the goal of the thesis are drawn in Chapter 9. Furthermore, proposals for future work are made in this chapter.

Chapter 2

Methods for Indoor Positioning

The topic of this thesis is the development of an accurate positioning algorithm suitable for use in an indoor lab setup. The task of the algorithm is to determine the position, orientation, and velocity of the Parrot AR.Drone 2.0 online. To provide an overview of the subject of indoor positioning, a short introduction is given in the first section of this chapter, followed by a brief description and comparison of a selection of indoor positioning systems.

The presentation of camera-based indoor positioning is given in more detail because both positioning algorithms implemented in this thesis are based on such a system. The field of camera-based positioning was investigated in the project work performed during the fall of 2012, a project in which Bluetooth-based methods were also considered, before a vision-based system using cameras as sensors was simulated. The sections in this chapter covering these two methods as well as the introduction to indoor positioning are based on the literature review performed in the project work [17]. The second proposed positioning algorithm in this thesis utilizes inertial sensors in addition to the camera system. Thus, inertial sensors in positioning algorithms are briefly presented here and elaborated in Chapter 6.

2.1 Introduction to Indoor Positioning

The task of an indoor positioning system is to provide accurate position information for moving objects in real-time - inside buildings. GPS is the most common positioning system for outdoor use, but the signals become unreliable when passing through walls and other obstacles. This necessitates another system more appropriate for indoor use.

Position information is essential when designing guidance and control algorithms, and the data must be accurate and up-to-date in order to obtain acceptable performance in the overall system. An IPS should therefore deliver highly accurate information while maintaining efficient data processing, a problem which has been addressed in the literature using several different types of sensors as well as data processing algorithms. However, a positioning system may be divided into the following necessary steps regardless of choice of sensors and algorithms:

· Measurements of the scene

· Processing of the measurements

· Real-time update of position estimates

2.2 Indoor Positioning Based on Wireless Technologies

2.2.1 Bluetooth

Indoor positioning systems using Bluetooth technology were, as previously mentioned, investigated in the project work [17]. Several of the reviewed approaches utilize *Received Signal Strength Indicator* (RSSI), which is a measure of the strength of a signal received by a Bluetooth device [8]. An RSSI value of zero indicates a signal power within an optimal range with regards to battery consumption during transmission. The RSSI value is converted to an estimate of the distance between a Bluetooth transmitter and mobile device through a propagation model, utilizing that the received power should decrease proportionally to the square of the distance from the transmitter [3], [8]. Furthermore, the estimated distance can be used to calculate the position of the mobile device through triangulation.

Using Bluetooth in an IPS is a relatively inexpensive solution because the technology is available in most ordinary, handheld devices today. However, Bluetooth devices are designed to minimize power consumption, which means that the transmission power is continuously adjusted in order to reach an RSSI level of zero [3]. Hence, the relationship between this value and distance is uncertain. The results obtained in the literature are quite inaccurate because of this uncertainty as well as other sources of error [3], [8]. Bluetooth-based methods which do not rely on RSSI values have also been investigated in [3] as well as in other reviewed papers, none of which reported sufficient accuracy for the application in this thesis. Thus, Bluetooth-based positioning systems are low-cost and fairly straight-forward, but also inaccurate.

2.2.2 Wireless Local Area Network

Another approach based on signal strength uses a *Wireless Local Area Network* (WLAN) for data transfer. In [18], the beacon frames emitted by (stationary) wireless *Access Points* (APs) are received by a mobile device, and subsequently used to determine the signal strength of all visible APs in its neighborhood. By storing these signal strengths and the locations where they were detected, a radio map of signal strength patterns is created in an offline phase. The assumption that each location in the environment is associated with a unique collection of signal strength values is made. An illustration of the offline phase is shown in Figure 2.1.

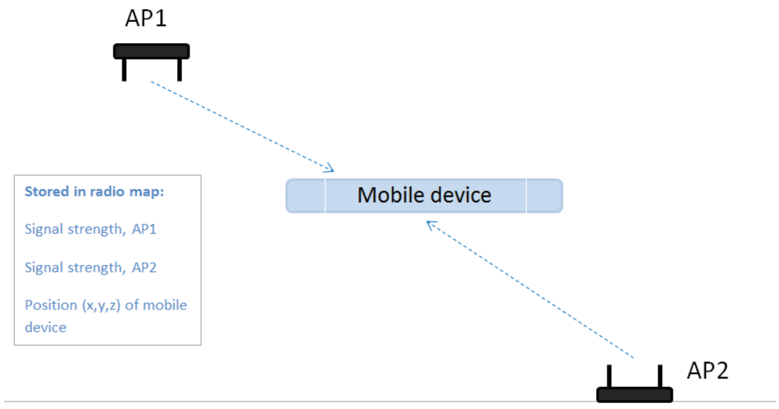

Figure 2.1: *Illustration of the offline phase in a WLAN indoor positioning system. Beacon frames from two APs are received by a mobile device, and the necessary information is stored.*

In the online phase, signal strength readings are compared to the radio map and the closest match is determined through a pattern recognition algorithm. Thus, an estimate of the current position of the mobile device is found. This method is quite straightforward and does not require additional hardware that is not commonly found in indoor environments, but it is also inaccurate due to multipath and other errors. Additionally, forming the radio map can be a time-consuming process because a large number of locations need be accounted for in order to provide acceptable position estimates in the online phase [18]. Another IPS utilizing WLAN technology in which positioning is modelled as a state estimation problem is proposed in [18], but the accuracy remains low with reported errors of > 1 m. Thus, indoor positioning based on WLAN technology does not improve the results reported for Bluetooth-based IPS, and would not be a better choice for the application in this thesis.

2.2.3 Radio Frequency Identification

A similar approach using *Radio Frequency Identification* (RFID) tags and readers is described in [9]. Tags are placed in an environment that has been divided into discrete locations. Reading patterns, i.e. patterns showing which tags are read by an RFID reader at the respective discrete locations, are collected in an offline phase. The patterns are then used in combination with a pattern recognition algorithm to determine the location of the RFID reader in the online phase. The method achieves errors of about 1 m, which although not sufficiently accurate for the use intended in this thesis, is better than the described WLAN approach. Knowledge of which tags are within reading range at a given time step is the only online information required for position estimation [9], making the RFID approach a quite simple method. However, additional hardware is required as compared to the Bluetooth and WLAN approaches, resulting in a higher cost and a more complex setup.

2.3 Inertial Sensors in Indoor Positioning

An inertial navigation system is a system consisting of an IMU and software for computing position, velocity, and attitude from the obtained measurements [16]. The sensor assembly of an IMU consists of three gyroscopes and three accelerometers for measuring angular velocities and linear accelerations, respectively. A *strapdown* inertial system is attached to the object of interest, and it has a relatively low weight. Hence, such a system may be used with the small vehicles capable of operating in indoor environments. In the field of navigation, inertial navigation systems are often combined with GPS to exploit the complementary advantages of the systems, and suppress their individual shortcomings. In an indoor environment, GPS may be replaced by another positioning system or external aid more suitable for indoor use. For instance, the use of an IMU combined with a map of the area of interest is proposed in [5]. Inertial sensors are robust and accurate for short-term applications, and external disturbances do not have much effect on such a system. Also, the system provides frequent measurements [16]. However, the estimates from an INS tend to drift, a problem which Glanzer et al. proposed to solve using characteristic building information, as mentioned above [5]. The use of IMUs in navigation systems has become quite common, and the cost of such devices has decreased significantly. The drift of the estimates is the main reason for combining an IMU with another indoor positioning scheme. The use of inertial sensors in indoor positioning will be further discussed in Chapter 6.

2.4 Camera-Based Indoor Positioning

Yet another type of sensors is used by camera-based indoor positioning systems, which is a wide notion covering several different image processing and computer vision techniques as well as physical system setups. However, it was discovered in the project work that the different approaches follow similar overall procedures in order to obtain the intended functionality [17]. There are two possible problem formulations when using cameras as sensors; to locate moving objects in images captured by one or several cameras, or to estimate the position and orientation of the camera itself. The first option is called a *system with static sensors*, while the latter approach is referred to as an *ego-motion system* [10]. Regardless of chosen system setup, an algorithm for object detection as well as a method for converting 2D (image) position into a 3D pose estimate is needed. In addition, a state estimator is often used for noise suppression, velocity estimation and tracking.

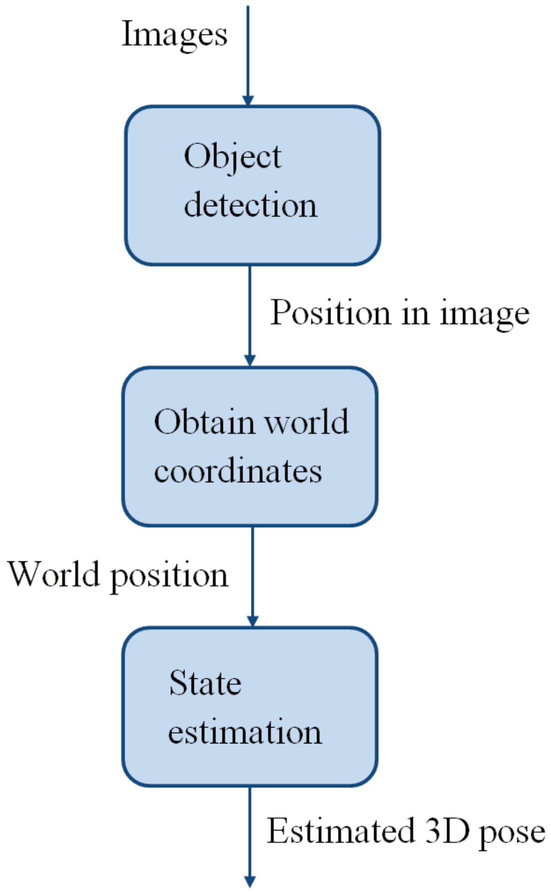

Figure 2.2: *Schematic of the overall procedure of camera-based positioning systems.*

This overall procedure is illustrated in Figure 2.2 and presented in the remainder of this section, which is based on [17].

2.4.1 Object Detection

Using cameras as sensors in a system for obtaining position necessitates an algorithm for detecting the object of interest in the image stream. However, the first step is deciding whether to search for a predefined moving object, e.g. a quadcopter, or some feature in the environment of operation of the IPS. The first option is, as mentioned earlier, referred to as a system with static sensors, and it enables tracking of several objects of interest. In such systems, the object for which position information is desired appears in the image stream of the cameras. Several papers have reported satisfactory results using markers attached to the object in the object detection procedure, i.e. [19]. When utilizing this approach, the object detection algorithm detects these predefined markers in the images, and calculates the 2D position of the object as the center (or some other geometric combination) of the detected markers [17]. The markers may be recognized by the algorithm through different characteristics such as color, shape, or reflected light. Regardless of chosen type of marker, placing them in a non-symmetrical pattern allows for calculation of the orientation of the object as well as its world position in a later step.

The latter option, searching for features in the environment of operation, involves placing the camera *on* the object to be localized, and is thus called an ego-motion system. To obtain the position of the camera through processing of the images captured by it, the use of *feature detection* has been proposed in the literature [17]. A feature is often a static, physical object in the environment of operation, and it should differ sufficiently from its surroundings to allow for detection in the images. In some approaches, the physical object is characterized by a special image feature, allowing for 2D detection through stated mathematical criteria [7]. Edges, corners, ridges, and blobs are examples of such image features, e.g. a ridge is used to describe elongated objects. The world position of the recognized object must be known in advance. From this information, an estimate of the 3D position as well as the orientation of the camera can be obtained. For details on object detection methods for both system setups, see [17].

2.4.2 Obtain World Coordinates

When the images from the camera(s) have been processed to detect an interesting region or object, this image position must be transformed into an estimate of the corresponding 3D position with respect to a world coordinate system. There are multiple algorithms for solving this 2D to 3D position problem, based on type of system setup. For systems with static sensors, *stereo vision* is an appropriate method. It requires two cameras in order to estimate the world position of a point appearing in the images from both cameras [13]. *Epipolar geometry*, i.e. the projective geometry of stereo vision, as well as the *intrinsic* and *extrinsic* parameters of the cameras are used in the intermediate calculations performed to obtain the world position of the point. The mentioned parameters are obtained through a process called *camera calibration*, and include the focal length and other parameters related to the inside of the camera (intrinsic parameters) as well as the position and orientation of the camera (extrinsic parameters). Thus, accurate knowledge of the system setup combined with projective geometry provides an estimate of the world position of a point. The method is easily combined with a marker detection algorithm, as discussed earlier. Accuracies of ± 1 cm have been reported in the literature, and the method can be augmented to calculate the orientation of the object of interest as well.

Space resection is a method suitable for use in ego-motion systems. The method requires knowledge of both the image and world coordinates of some control points appearing in the image stream in order to calculate the world coordinates and orientation of the camera capturing the images [1]. The intrinsic parameters of the camera are needed as well. An ego-motion system using space resection and a system with static sensors using stereo vision both constitute accurate camera-based positioning systems, although the two approaches may be thought of as solving opposite problems. For details on 2D to 3D position algorithms, see [17].

2.4.3 State Estimation

The world position estimates are, along with the orientation of the object of interest, among the desired results of the positioning system. However, the estimates may be noise-infested or even lost for some time steps, a problem which can be solved using a state estimator. Furthermore, one may not be able to estimate all of the desired states of the object of interest through the camera-based system alone. Many of the papers reviewed in [17] use a Kalman filter for these purposes. The position and orientation estimates obtained by the camera-based system are used as measurement inputs to the Kalman filter, which performs noise filtering, prediction, and also reconstruction of unmeasured states from the measurements [4]. Additionally, a Kalman filter may be used to fuse measurements from several sensors in order to improve the accuracy of the state estimates and provide redundancy [16]. The extended Kalman filter, which is used with nonlinear systems, will be presented in Chapter 5 and updated to utilize the sensor fusion feature in Chapter 6. Further details on both linear and nonlinear Kalman filtering were given in [17].

2.5 Summary, Comparison, and the IPS of this Thesis

Indoor positioning based on three different wireless technologies as well as inertial sensors and cameras have been presented. The systems based on wireless technologies are quite similar, resulting in the following coinciding strength and weakness; The methods are relatively straight-forward, but do not obtain high accuracies. The Bluetooth and WLAN approaches are low-cost because the necessary hardware is found in most handheld devices, while the RFID method requires specialized equipment. A camera-based approach also requires extra hardware, but high accuracies have been reported in the literature using ordinary, low-cost web-cameras [17]. However, the utilized computer vision and image processing techniques can be quite advanced, resulting in a need for efficient implementation. Inertial sensors are low-cost and provide frequent and accurate measurements for short-term applications, but the estimates tend to drift. Thus, an additional system for resetting the procedure and providing long-term accuracy is advised, resulting in an integrated solution.

From the review given above, it may be noted that indoor positioning systems differ with respect to the sensors used to obtain the required measurements of the scene. The techniques for transforming the measurements into pose estimates vary accordingly. A choice must be made by the developer as to what is more important when designing an indoor positioning system for a specific application - low cost, high accuracy, or moderate complexity.

In this thesis, a system combining the use of an IMU and a camera-based positioning system is the ultimate goal. This combination is expected to produce accurate and frequent estimate updates, although the cost will be higher than that of using web-cameras as the vision-based sensors. The reason is that a specialized camera system which delivers preprocessed position and orientation measurements based on marker tracking is used. Thus, some of the complexity connected to using a camera-based IPS is removed, while the cost of the system increases. The following chapter presents the various components of the camera system responsible for delivering position and orientation measurements in this thesis.

Chapter 3

The Camera-Based Positioning System OptiTrack

The two positioning algorithms implemented in this thesis both rely on position and orientation measurements from a camera-based system. The chosen camera system is OptiTrack from NaturalPoint, which is an overall *optical motion capture system* provider. OptiTrack is the subgroup of NaturalPoint's systems which provides optical tracking of humans and objects capable of leaving the immediate proximity of the sensors, and it includes several different options of both hardware and software for this purpose. In this thesis, equipment from OptiTrack is used with the aim of tracking a quadcopter in an indoor lab setup and streaming the obtained information to MATLAB for use in an extended Kalman filter. The provided measurements consist of position and orientation data in six degrees of freedom, i.e. a vector of three position and three orientation variables.

A brief presentation of motion capture systems in general is given in the first section of this chapter, while the chosen components from OptiTrack as well as other aspects related to the utilized system are presented in subsequent sections (based on available material from the manufacturer). Finally, a constrained test of the camera system accuracy as well as an investigation of the data transfer between OptiTrack and MATLAB have been performed. A presentation of the findings concludes this chapter.

3.1 Motion Capture Systems

In [11], motion capture is defined as "The creation of a 3D representation of a live performance". That is, a motion capture system observes an object moving around a scene through sensors, and recreates the performed motion in three dimensions and in real-time. There are several types of motion capture systems, e.g. mechanical, electromagnetic, and optical, of which optical is the most commonly used technology. Such systems often require special markers (which can be tracked by the cameras) attached to the body or object, enabling the recreation of the motion of the object of interest [11]. The markers may be reflective, i.e. the motion capture cameras emit light which is reflected by the markers, or the markers themselves can emit light. The most commonly known use of optical motion capture is perhaps related to the film industry, where actors wearing special suits perform the movements intended for an animated character. The suits are equipped with markers, allowing the motion of the actor to be captured and transferred to the animated character.

OptiTrack is, as mentioned earlier, an optical motion capture system. In this thesis, the system is used to obtain position and orientation measurements of the Parrot AR.Drone 2.0 for use in positioning algorithms.

3.2 Sensors - The Flex 13

As mentioned above, a camera-based positioning system is in this thesis used to obtain measurement inputs for a state estimator. Together, the measurement acquisition and subsequent processing constitute a positioning algorithm for an indoor quadcopter, which is a type of UAV utilizing four rotors to generate lift and motion. Thus, the object of interest is capable of moving in the entire test environment, and the sensors must cover as much of the area as possible. That is, if the cameras fail to cover any part of the lab setup, the measurements from OptiTrack will be lost if and when the quadcopter enters that particular area.

The chosen sensor is the **Flex 13** camera, which has the technical specifications listed in Table 3.1[1]. The specifications of Table 3.1 are found in the Flex 13 Data Sheet [2] and on the company website (see footnote). The Flex 13 is a

Table 3.1: *Technical Specifications, Flex 13*

Resolution	1280×1024 pixels
Maximum Frame Rate	120 FPS
Horizontal FOV	56°
Vertical FOV	46°
Accuracy	Sub-millimeter marker precision
Latency	8.3 ms
Maximum Range	12.2 m

medium volume motion capture camera, which means that the camera is capable of tracking objects at a maximum distance of approximately 12 m (see Table 3.1).

[1]Source: www.naturalpoint.com/optitrack/products/flex-13

The camera is equipped with 28 *Light-Emitting Diodes* (LEDs) which emit IR light (see Figure 3.1). The reason for attaching the LEDs to the camera is that the Flex 13 detects points which reflect its emitted light. As a consequence, light-reflective markers need be attached to the object of interest. From this marker information, position and orientation is calculated. Some of the required image processing is performed in the camera before the data is sent to a PC. This will be further discussed below.

3.2.1 Accuracy and Frame Rate

From Table 3.1, it can be seen that the promised accuracy of the Flex 13 is in the sub-millimeter range for individual markers. Furthermore, the camera is capable of delivering frames at a frequency of 120 Frames Per Second (FPS), which is more than sufficient for NaturalPoint's object tracking software Tracking Tools to deliver timely pose measurements (Tracking Tools will be presented in Section 3.3). The combination of a high accuracy and a high output rate in the sensors is desired for the application intended in this thesis.

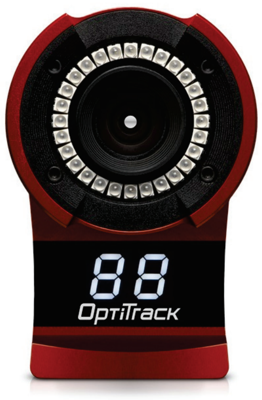

Figure 3.1: *Photo from NaturalPoint.*[2] *The Flex 13 camera from NaturalPoint. LEDs form a circular pattern around the lens.*

3.2.2 System Setup and On-Camera Processing

The camera system setup used in this thesis consists of 16 Flex 13 cameras placed with the aim of covering as much of the room as possible, i.e. to maximize the *capture volume*. The capture volume is the area in which multiple OptiTrack cameras have overlapping Fields of View (FOV). Tracking of the reflective markers can only occur within this area, i.e. a marker must be visible to at least two cameras (ideally more) for it to be tracked [14]. In Figure 3.2, a capture

[2]Source: www.naturalpoint.com/optitrack/products/flex-13/

volume resulting from three overlapping fields of view is depicted for illustration of this concept. The capture volume is shown as the gray area in the figure. The fields of view of the Flex 13 are stated to be 56° and 46° for the horizontal and vertical directions, respectively. In comparison, humans are typically capable of observing about 180° horizontally and 120° vertically. Although the Flex 13 has considerably smaller FOVs than humans, a collection of 16 cameras arranged cleverly in the environment should produce a sufficiently large capture volume for the limited Parrot AR.Drone 2.0 lab setup intended in this thesis.

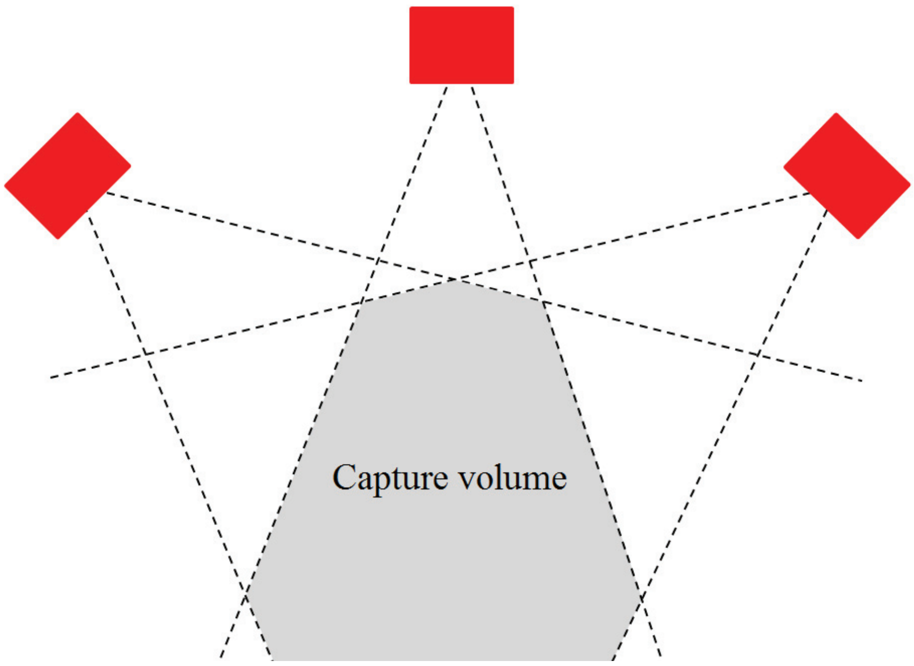

Figure 3.2: *Figure reproduced from [15]. The resulting capture volume from three (red) cameras. The scene is depicted from above.*

Four of the cameras in the system setup are shown in Figure 3.3. As mentioned earlier, the cameras are placed with the intention of including as much of the room as possible in the capture volume. This is obtained by placing all 16 cameras along the ceiling in a similar manner as the four cameras depicted in Figure 3.3. The overall reason for using the cameras is to send precise quadcopter position and orientation data to MATLAB for use in a positioning algorithm. This data is obtained through the use of the OptiTrack tracking software Tracking Tools. However, some preprocessing of the captured frames is performed in each camera. The user may choose between several video types, a choice which influences the type and amount of image processing performed on-board, and thus the processing required of the PC. In this thesis, the *Object Mode*, in which the location and size of the reflective markers is detected on-board each camera, is chosen. Thus, the PC receives marker information from which rigid body position and orientation is calculated. The Object Mode is chosen because it requires less bandwidth than the Precision Mode while still delivering highly accurate data [14].

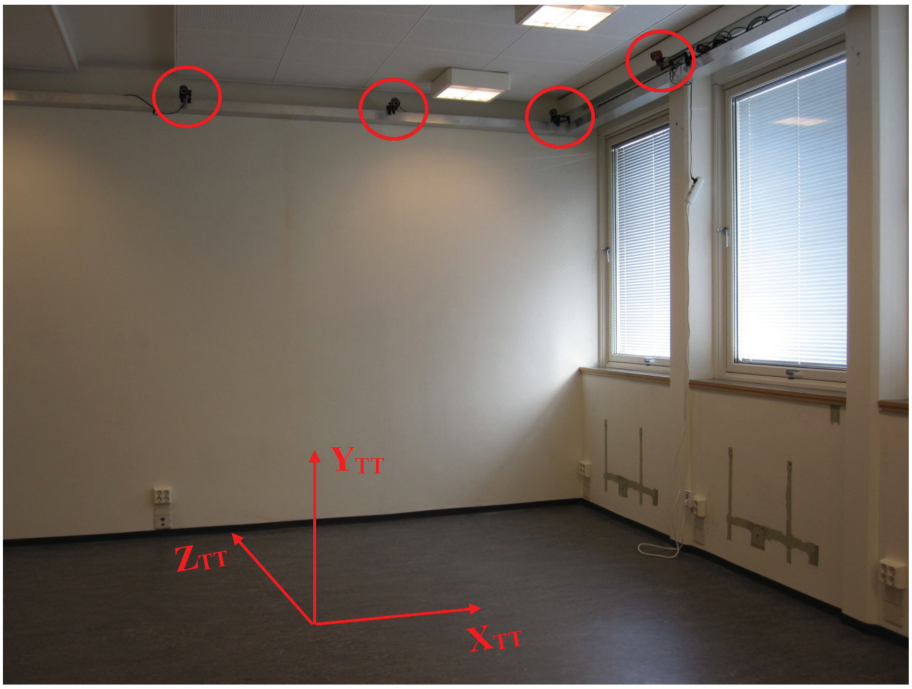

Figure 3.3: *Part of the OptiTrack system setup. Four Flex 13 cameras along the ceiling are highlighted by red circles while the left-handed coordinate system used by Tracking Tools is placed on the floor.*

The coordinate frame used by Tracking Tools is depicted in Figure 3.3. The x_{TT} axis points towards one of the long sides of the lab room, while the z_{TT} axis is directed towards one of the short edges. The y_{TT} axis is perpendicular to the $x - z$ plane, completing a *left-handed* coordinate frame.

3.3 Software - Tracking Tools

When the cameras have captured images of the scene and obtained the location and size of the detected markers, the information is sent to Tracking Tools, where a 6 DOF pose estimate for the rigid body is calculated from the received information. This section presents relevant aspects of Tracking Tools, while a presentation of the required communication between the utilized OptiTrack software and MATLAB is provided in Section 5.3.

3.3.1 Initialization of the System

Calibration When the cameras have been set up, a calibration must be performed. A wand with three reflective markers attached to it (identical to the markers attached to the UAV for detection) is waved through the capture volume while the cameras take samples, a process called "wanding" [14]. It is important to cover as much as possible of the capture volume to ensure sufficient sampling and proper performance by the tracking system. When the samples have been collected, they are used to solve the intrinsic and extrinsic parameters of the cameras (see Section 2.4). Thus, the somewhat complicated calibration calculations required in camera-based positioning systems are performed by Tracking Tools.

Ground Plane When the 16 cameras of the system are calibrated, a world coordinate frame to which the position and orientation measurements will be referenced is set by the user. The frame is left-handed by default, and in this thesis, it is set as shown in Figure 3.3. Thus, a conversion to a somewhat more common right-handed coordinate frame becomes a necessary part of the measurement preprocessing performed in a positioning algorithm using the measurements. This will be further discussed in subsequent chapters.

Create Rigid Body Once the ground plane is set, rigid bodies may be defined relative to it. To track a rigid body using this system, reflective markers must be attached to it, as explained earlier. The markers are placed on the object in a random pattern, although it is important that the pattern is asymmetrical to allow for orientation estimation. In this thesis, the markers are placed on the Parrot AR.Drone 2.0 as shown in Figure 3.4.

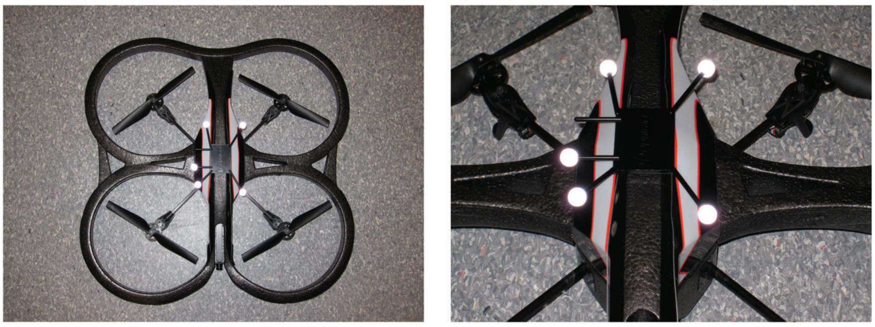

Figure 3.4: *The Parrot AR.Drone 2.0 with reflective markers attached to it. The quadcopter is shown to the left, while a close-up of the attached markers is depicted to the right. The collection of markers is placed on top of the quadcopter to exploit the placement of the cameras along the ceiling.*

The five reflective markers are attached with the intention of tracking the quadcopter in the entire lab room. As shown in Figure 3.3, the cameras in the lab setup are placed along the ceiling. Thus, placing the markers on top of the UAV should ensure its visibility in a sufficiently large area. The collection of five markers shown in Figure 3.4 is defined as a rigid body (called "trackable") in Tracking Tools, and the software tracks the center point of this trackable. Hence, the markers should be placed close to the center point of the quadcopter for accurate tracking through marker detection. The asymmetrical pattern ensures accurate orientation measurements (see Figure 3.4).

The process of defining an object as a rigid body mentioned above includes placing the object within the area covered by the cameras, and manually selecting its markers as part of the rigid body to be localized. Thus, a unique arrangement of markers define a unique rigid body, and its position and orientation may be tracked by the system. The group of markers on top of the Parrot AR.Drone 2.0 is the only defined trackable in this thesis, although an extension of the system to include several UAVs would be straightforward.

3.3.2 Marker Detection and Rigid Body Tracking

The system is ready for use when the initialization outlined above is performed, i.e. real-time position and orientation information for the defined trackable is available whenever at least three of its markers are visible to the cameras. However, limited information is revealed from the manufacturer regarding the algorithms used to obtain these estimates. It is assumed that a marker detection algorithm is running in each of the cameras, while a technique for transforming the image positions of the markers to a world position estimate for the trackable is performed by Tracking Tools. It *is* known that when several cameras are being used, as is the case in this thesis, triangulation is used in the process of obtaining the world position of a point of interest.

3.4 The Camera System used in Positioning Algorithms

The camera-based positioning system used in this thesis is classified as a system with static sensors, which implies that objects of interest are detected in the images captured by the cameras (see Section 2.4). As mentioned earlier, marker detection is performed using emitted and reflected IR light, and the world position of the object is calculated by Tracking Tools using triangulation and possibly computer vision techniques. Hence, the main measurement system in this thesis performs the first two steps of the overall procedure of camera-based positioning systems shown in Figure 2.2, and delivers measurements to the state estimation block of the figure. In the first positioning algorithm proposed in this thesis, an extended Kalman filter performs state estimation based solely on measurements from the OptiTrack system. That is, the Camera Measurement Algorithm represents a possible implementation of the schematic shown in Figure 2.2.

3.4.1 Accuracy of the OptiTrack System

The promised accuracy in the measurements from OptiTrack should be tested to ensure that no large errors are propagated through the algorithm. However, due to lack of redundancy in available sensors, there is a significant source of error in this test; the physical movement of the trackable is quite inaccurate when measured manually. This is the reason why only linear motion in the $x-z$

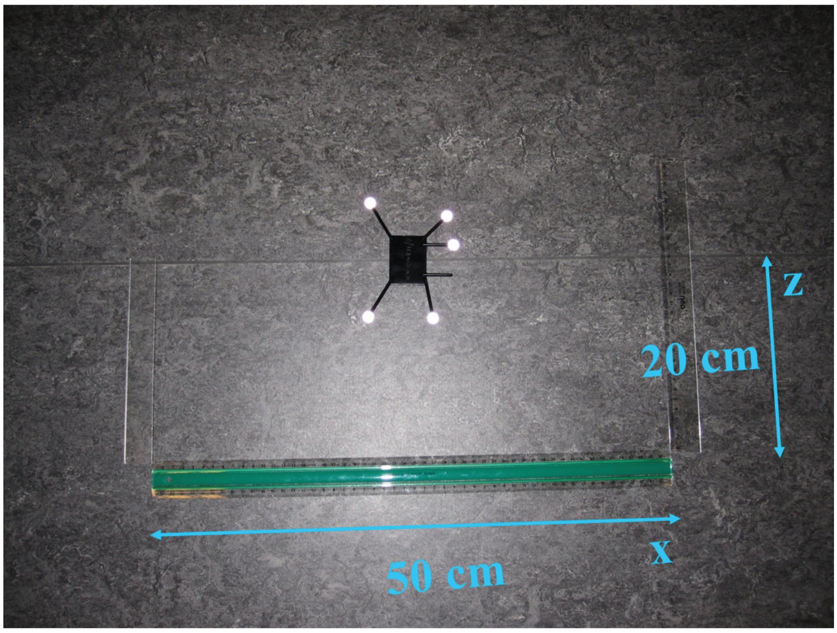

Figure 3.5: *Setup for the OptiTrack accuracy test. Accuracy in the measurement of planar, linear motion is tested by moving the trackable a predefined distance along the rulers and observing the corresponding change in position in Tracking Tools.*

plane of the Tracking Tools coordinate system is considered in this test - accurate test measurements of angular motion and movement along the vertical axis proved too difficult to obtain. Thus, the test is merely an investigation of whether the camera system delivers measurements with accuracies within an acceptable range, and it provides preliminary insight into the main measurement source of the implemented positioning algorithms.

The test setup consisted of three rulers and the defined trackable as shown in Figure 3.5. The trackable was moved according to the rulers and the corresponding change in position was observed in Tracking Tools. The results of the accuracy test are shown in Table 3.2. From the table, it can be seen that the physically moved distances ("Distance moved along ruler") correspond to the resulting observations displayed in Tracking Tools with less than 1 cm error in all three tests and for both the x and z directions.

Table 3.2: *Test Results, OptiTrack Accuracy Test*

	Test 1	**Test 2**	**Test 3**
Start position x, TT	-1.93 cm	-2.27 cm	-1.93 cm
Start position z, TT	0.012 cm	0.16 cm	20.17 cm
Distance moved along ruler, x	50 cm	0 cm	25 cm
Distance moved along ruler, z	20 cm	20 cm	0 cm
Stop position x, TT	49.01 cm	-1.93 cm	23.75 cm
Stop position z, TT	19.34 cm	20.17 cm	19.58 cm

However, the source of error mentioned above is significant, and produces limitations in the results due to the difficulty of knowing with certainty whether the trackable was moved erroneously, or if the position calculated by Tracking Tools is less accurate than promised. The promised accuracy is in the sub-millimeter range for *individual marker positions*, and the calculation of the position of the trackable from the marker information is not available to the user. Thus, it is difficult to inspect.

Another aspect of this test is that the user of an OptiTrack system is responsible for performing a satisfactory calibration of the cameras prior to use. As a consequence, the user may be at fault if the provided tracking accuracy is less than optimal. However, a calibration result within the limits for optimal use (set by the OptiTrack developers) was obtained in this thesis, making another source of error more likely. No conclusion can be drawn with certainty with regards to OptiTrack's delivered accuracy using the equipment at hand. However, the performed test in combination with observed changes in orientation and height does show a promising accuracy for the main measurement source in this thesis.

3.4.2 Measurement Acquisition and Measurement Noise

The use of the OptiTrack *Application Programming Interface* (API) for obtaining position and orientation measurements is tested by comparing stationary measurements observed in the user interface of Tracking Tools to the same stationary measurements obtained by MATLAB (the API will be presented in Ch. 5). The test is performed to ensure proper transfer of data, and to test the amount of time spent on the start-up procedures of the OptiTrack system. Furthermore, performing this test while the trackable is stationary provides an opportunity to investigate the amount of noise in the measurements.

Table 3.3: *Test of OptiTrack to* MATLAB *communication - Location displayed in Tracking Tools*

x^{TT}	y^{TT}	z^{TT}	ϕ^{TT}	θ^{TT}	ψ^{TT}
0.190 ± 10^{-5} m	0.011 ± 10^{-5} m	$-0.080 \pm 3 \cdot 10^{-5}$ m	$-1.01 \pm 0.08°$	$0.71 \pm 0.3°$	$5.73 \pm 0.01°$

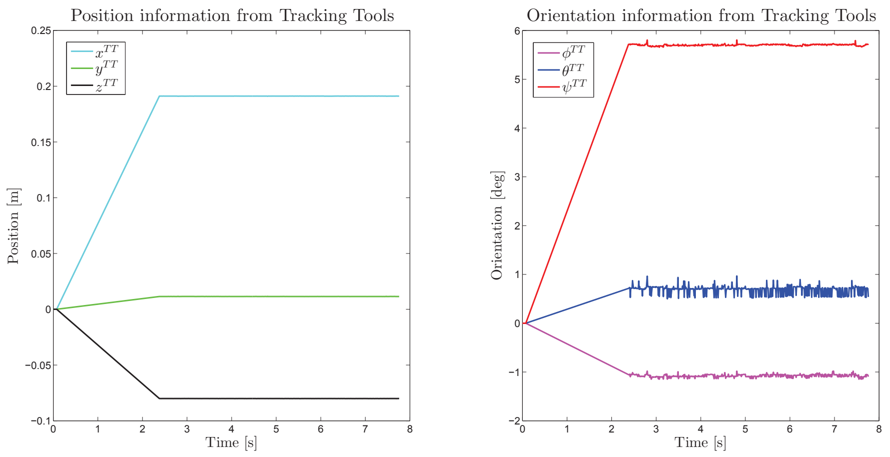

Figure 3.6: *Test of OptiTrack to* MATLAB *communication - Location received by* MATLAB*. The position measurements from OptiTrack are shown to the left, while the orientation measurements are depicted to the right.*

The initialization of the system is visible as the transients in Figure 3.6, i.e. the period before the signals stabilize around their static values. This is a result of performing necessary start-up procedures for the cameras and API, and it can be seen that valid position and orientation estimates are received after approximately 2.5 seconds. Figure 3.6 also shows measurement noise in the orientation measurements, corresponding to the uncertainty observed in Tracking Tools (Table 3.3). However, the units of the two parts of Figure 3.6 differ, and a comparison of the amount of noise in the position and orientation measurements is difficult to perform based on this test alone.

Chapter 4

Terminology and Notation

This chapter presents the notation used throughout this thesis as well as important vectors used in the two positioning algorithms. The utilized coordinate frames are presented because of their importance with regards to mathematical modelling of the two systems.

4.1 Coordinate Systems

Three coordinate systems are used in this thesis: The *North-East-Down* (NED) frame, the BODY frame, and the left-handed frame to which the measurements obtained by the OptiTrack system are referenced. The three coordinate systems are shown in Figure 4.1 (the left-handed OptiTrack coordinate system is also depicted in Figure 3.3).

In Figure 4.1, the BODY frame is depicted in green. It is firmly attached to the quadcopter, and moves with it. The BODY frame has the following axes [4]:

· x_b directed from the back towards the front of the UAV
· y_b directed towards the right side of the UAV
· z_b directed from top to bottom

The NED frame (depicted in light blue in Figure 4.1) is attached to the ground, and assumed inertial in this thesis. Consequently, it has a fixed relation to the lab room, and it is the frame to which the position estimates from the two implemented algorithms are referenced.

The OptiTrack coordinate system (depicted in dark blue in Figure 4.1) is included in this presentation because it is left-handed by default, and cannot be changed. It is worth noting that the following convention is used by the camera system: ϕ^{TT} is the angle about the x_{TT} axis, θ^{TT} is the angle about the z_{TT} axis, and ψ^{TT} is the angle about the y_{TT} axis. Furthermore, because the OptiTrack frame is left-handed, the positive rotation direction is clockwise.

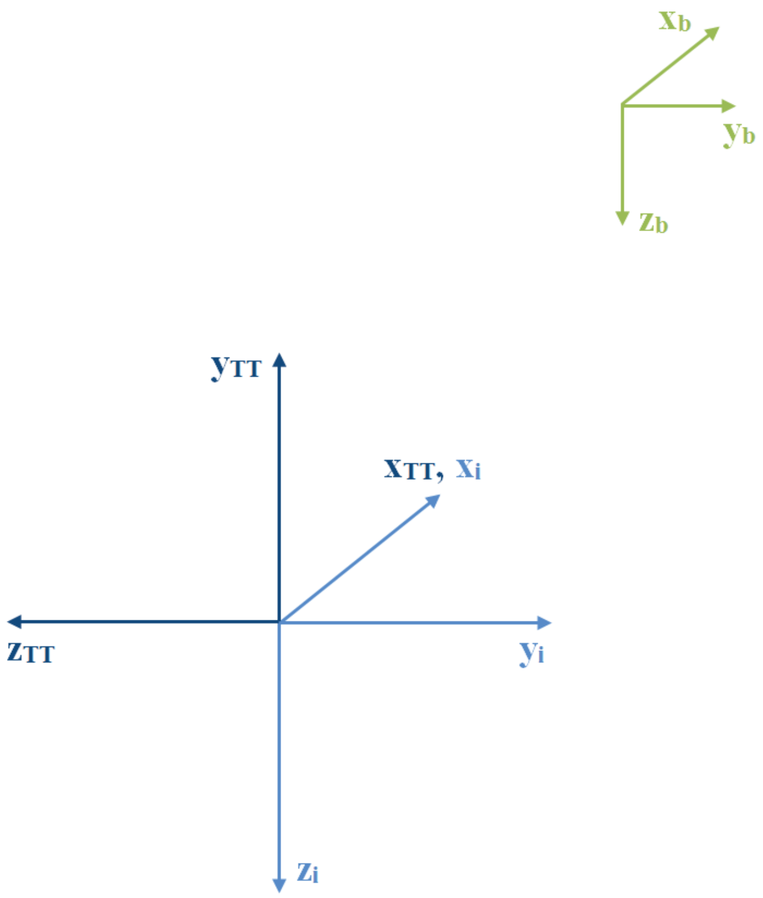

Figure 4.1: *The (inertial) NED frame (x_i, y_i, z_i) and the left-handed Tracking Tools frame (x_{TT}, y_{TT}, z_{TT}) are shown in blue, while the BODY frame (x_b, y_b, z_b) is depicted in green.*

4.2 Notation

The three coordinate systems are abbreviated as follows:

- (Inertial) NED frame: i
- BODY frame: b
- Left-handed Tracking Tools frame: TT

As can be seen in Figure 4.1, the *axes* of a coordinate frame are written with *subscripts* showing to which frame they belong, i.e. x_i is the x axis of the inertial frame.

Vectors are written in bold letters, and the following vectorial notation is adopted from [4]. To show which frame a vector is decomposed in, *superscripts* are used. As an example, if the vector $\mathbf{a}$ is expressed in the BODY frame, it is denoted $\mathbf{a}^{\mathrm{b}}$. To describe movement of a frame with respect to another, subscripts are used. For instance, $\mathbf{c}^{\mathrm{TT}}_{\mathrm{b/i}}$ means that the vector $\mathbf{c}$ describes motion of the body-fixed frame b with respect to i, expressed in TT (superscripts are used as explained above). The Euler angles between the inertial and body-fixed frames are denoted $\mathbf{\Theta}_{\mathrm{ib}}$.

For vectors and variables with subscripts not on the form $\mathbf{c}^{\mathrm{TT}}_{\mathrm{b/i}}$, the subscripts simply denote an explanation or numbering of the variable in question (e.g. $\mathbf{b}^{\mathrm{b}}_{\mathrm{acc}}$ or $\mathbf{x_1}$), or it is part of the variable name (v_{x}).

4.3 Important Vectors

State Vector 1 The state vector used in the Camera Measurement Algorithm for quadcopter motion in all six degrees of freedom is given by

$$\mathbf{x_1} = \left[x, y, z, \phi, \theta, \psi, v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{z}}, \omega_{\mathrm{x}}, \omega_{\mathrm{y}}, \omega_{\mathrm{z}}\right]^{\mathsf{T}}, \tag{4.1}$$

where the various elements refer to the following:

· x - position in the x direction
· y - position in the y direction
· z - position in the z direction
· ϕ - rotation about the x axis (Euler angle)
· θ - rotation about the y axis (Euler angle)
· ψ - rotation about the z axis (Euler angle)
· v_{x} - linear velocity in the x direction
· v_{y} - linear velocity in the y direction
· v_{z} - linear velocity in the z direction
· ω_{x} - angular velocity about the x axis
· ω_{y} - angular velocity about the y axis
· ω_{z} - angular velocity about the z axis

State Vector 2 The state vector used in the Integrated Camera System/INS Algorithm is given by

$$\mathbf{x_2} = \left[x, y, z, \phi, \theta, \psi, v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{z}}, b_{\mathrm{acc,x}}, b_{\mathrm{acc,y}}, b_{\mathrm{acc,z}}, b_{\mathrm{gyro,x}}, b_{\mathrm{gyro,y}}, b_{\mathrm{gyro,z}}\right]^{\mathsf{T}}, \tag{4.2}$$

where the first nine elements are equal to the nine first states of $\mathbf{x_1}$ while the latter six refer to:

· $b_{\mathrm{acc,x}}$ - accelerometer bias, motion in the x direction
· $b_{\mathrm{acc,y}}$ - accelerometer bias, motion in the y direction

- $b_{\mathrm{acc,z}}$ - accelerometer bias, motion in the z direction
- $b_{\mathrm{gyro,x}}$ - gyroscope bias, motion about the x axis
- $b_{\mathrm{gyro,y}}$ - gyroscope bias, motion about the y axis
- $b_{\mathrm{gyro,z}}$ - gyroscope bias, motion about the z axis

The elements of $\mathbf{x_1}$ and $\mathbf{x_2}$ are given with respect to different coordinate systems as follows:

$$\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \boldsymbol{\Theta}_{\mathrm{ib}} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}} = \begin{bmatrix} v_{\mathrm{x}} \\ v_{\mathrm{y}} \\ v_{\mathrm{z}} \end{bmatrix}, \boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} = \begin{bmatrix} \omega_{\mathrm{x}} \\ \omega_{\mathrm{y}} \\ \omega_{\mathrm{z}} \end{bmatrix} \tag{4.3}$$

$$\mathbf{b}^{\mathrm{b}}_{\mathrm{acc}} = \begin{bmatrix} b_{\mathrm{acc,x}} \\ b_{\mathrm{acc,y}} \\ b_{\mathrm{acc,z}} \end{bmatrix}, \mathbf{b}^{\mathrm{b}}_{\mathrm{gyro}} = \begin{bmatrix} b_{\mathrm{gyro,x}} \\ b_{\mathrm{gyro,y}} \\ b_{\mathrm{gyro,z}} \end{bmatrix} \tag{4.4}$$

Measurement Vector 1 The measurement vector from OptiTrack is modelled as

$$\mathbf{h}(\mathbf{x_1}) = \left[x, y, z, \phi, \theta, \psi\right]^{\mathsf{T}} \tag{4.5}$$
$$\mathbf{z} = \mathbf{h}(\mathbf{x_1}) + \mathbf{v_1}, \tag{4.6}$$

i.e. position and orientation are measured. $\mathbf{v_1}$ is the measurement noise.

Because the TT and i frames are defined according to opposite conventions, no series of ordinary rotations will provide the transition between them. As can be seen in Figure 4.1, the x axes coincide while the y and z axes are interchanged and pointed in opposite directions. This transformation need be performed for the measurement vector from OptiTrack, $\mathbf{z}^{\mathrm{TT}}$, in order to use it in the positioning algorithms. That is, the linear elements of the measurement vector should be expressed in the inertial frame, implying that the angular elements are the Euler angles Θ_{ib}.

Measurement Vector 2 and Input Vector The measurement vector used in the Integrated Camera System/INS Algorithm is equal to the measurement vector from OptiTrack, i.e.

$$\mathbf{h}(\mathbf{x_2}) = \mathbf{h}(\mathbf{x_1}) = \left[x, y, z, \phi, \theta, \psi\right]^{\mathsf{T}} \tag{4.7}$$
$$\mathbf{z} = \mathbf{h}(\mathbf{x_2}) + \mathbf{v_2}, \tag{4.8}$$

where $\mathbf{v_2}$ is the measurement noise.

The input vector to the extended Kalman filter in the Integrated Camera System/INS Algorithm is modelled as

$$\mathbf{u} = \begin{bmatrix} a_{\text{meas,x}} + b_{\text{acc,x}} \\ a_{\text{meas,y}} + b_{\text{acc,y}} \\ a_{\text{meas,z}} + b_{\text{acc,z}} \\ \omega_{\text{x}} + b_{\text{gyro,x}} \\ \omega_{\text{y}} + b_{\text{gyro,y}} \\ \omega_{\text{z}} + b_{\text{gyro,z}} \end{bmatrix} = \begin{bmatrix} u_{\text{acc,x}} \\ u_{\text{acc,y}} \\ u_{\text{acc,z}} \\ u_{\text{gyro,x}} \\ u_{\text{gyro,y}} \\ u_{\text{gyro,z}} \end{bmatrix} \tag{4.9}$$

$$\mathbf{u}_{\text{IMU}} = \mathbf{u} + \mathbf{w}_{\text{IMU}}, \tag{4.10}$$

where $a_{\text{meas,x}}$, $a_{\text{meas,y}}$, $a_{\text{meas,z}}$ are the noise- and bias-free outputs from accelerometers, while ω_{x}, ω_{y}, ω_{z} are noise- and bias-free outputs from gyroscopes. $\mathbf{w}_{\text{IMU}}$ is the measurement noise from an IMU. Thus, IMU measurements are modelled as inputs to the EKF.

Chapter 5

Camera Measurement Algorithm: State Estimation using Position and Orientation Information

This chapter presents all components of the first implemented positioning algorithm, i.e. using the measurements from OptiTrack only. The camera-based positioning system was presented in Chapter 3, while this chapter connects the measurement vector to the rest of the algorithm (see Figure 5.1). The first section presents an overview of the main functionality of the algorithm, while the more important aspects as well as the handling of various deviations from the normal flow are elaborated in subsequent sections. Mathematical modelling of the system is performed in Section 5.5 before the extended Kalman filter used in the algorithm is presented. Finally, aspects related to implementation of the Camera Measurement Algorithm are explained.

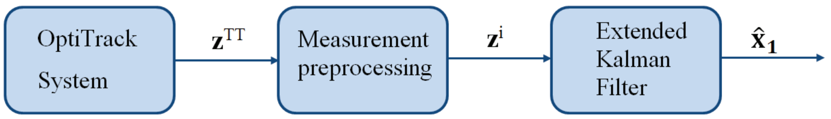
OptiTrack System
$\mathbf{z}^{\mathrm{TT}}$
Measurement preprocessing
$\mathbf{z}^{\mathrm{i}}$
Extended Kalman Filter
$\hat{\mathbf{x}}_1$

Figure 5.1: *High-level schematic of the Camera Measurement Algorithm. To the left, the OptiTrack subsystem as presented in Chapter 3 is depicted. The measurement vector z^{TT} is obtained from OptiTrack, preprocessed, and subsequently used as measurement input to the EKF subsystem, which is to the right in the figure. The outputs from the EKF are the desired variables - the position, orientation, and linear/angular velocity estimates.*

5.1 Main Functionality of the First Positioning Algorithm

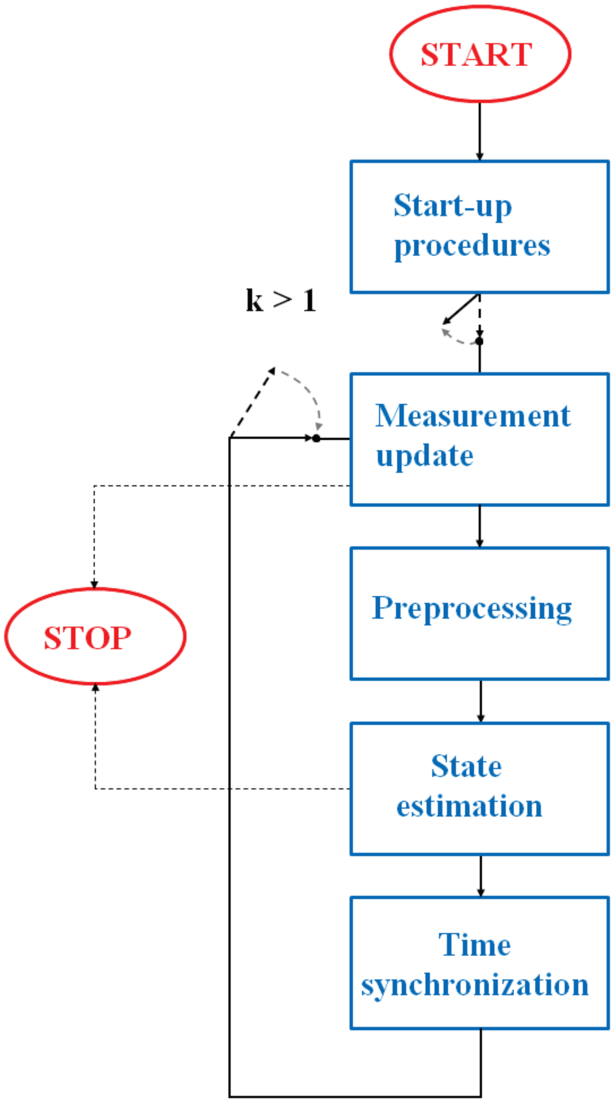

Figure 5.2: *Overall schematic of the main functionality of the Camera Measurement Algorithm. The red blocks indicate start-up and stop actions, while the blue blocks represent the various modules of the algorithm. Initially, the start-up procedures are performed, while only the four lower modules are active afterwards. The dotted lines pointing into "STOP" indicate that the algorithm may be aborted. Handling of possible errors is performed in the modules, and is therefore not visible.*

Figure 5.2 is a somewhat more detailed schematic of the Camera Measurement Algorithm, included with the intention of clarifying the flow of the algorithm. This section aims at presenting a successful running (without deviations from the normal flow due to system errors) of the Camera Measurement Algorithm in order to provide an overview of its main functionality (handling of various system errors will be presented in Section 5.4). The red "START" and "STOP" blocks in Figure 5.2 refer to starting and closing down the system, while the blue blocks represent the various subsystems of the algorithm which will now be presented.

Start-up Procedures The top-most module represents the necessary start-up procedures for the camera system and the utilized API, which is a protocol for communication between different software components. In this thesis, real-time Tracking Tools data must be accessed by Matlab, a topic which is discussed in Section 5.3 below. The start-up procedures include loading the Tracking Tools API, loading a performed calibration as well as defined trackables in Matlab, and initialization of the API as well as of several variables used by the algorithm.

Measurement Update When the start-up procedures have been performed without errors, the cameras are collecting frames and the OptiTrack system can be accessed from Matlab. For the current group of frames (one frame from each of the 16 cameras), a search for the trackable connected to the UAV is performed by OptiTrack, and a time stamp is stored. If the trackable is detected, its position and orientation data is obtained. This corresponds to obtaining the measurements $\mathbf{z}^{\mathrm{TT}}$ from the OptiTrack system as discussed in Chapter 3.

Preprocessing The OptiTrack system measures angles between $-180°$ and $180°$, i.e. the measurements obtained from the camera system contain leaps of $\pm 360°$ when an angular limit of either $-180°$ or $180°$ is crossed. In the non-planar angles ϕ and θ, such leaps are rarely experienced, but ψ may very well reach these limits. A non-continuous measurement will propagate through the system due to the system equations being interconnected. Thus, the OptiTrack measurements $\mathbf{z}^{\mathrm{TT}}$ are preprocessed to avoid such leaps by letting the angular limits become $\pm\infty$.

Furthermore, the measurements are transformed into the inertial coordinate frame before use in the extended Kalman filter. The resulting position and orientation is in accordance with the coordinate systems depicted in Figure 4.1 (and the definition of angles used by the camera system, which was presented in Section 4.1). An illustration of the various measured states expressed in the two frames is shown in Figure 5.3 for clarity.

State Estimation The state estimates $\hat{\mathbf{x}}_{\mathbf{1}}$ are calculated by an extended Kalman filter, which is further discussed in Section 5.6.

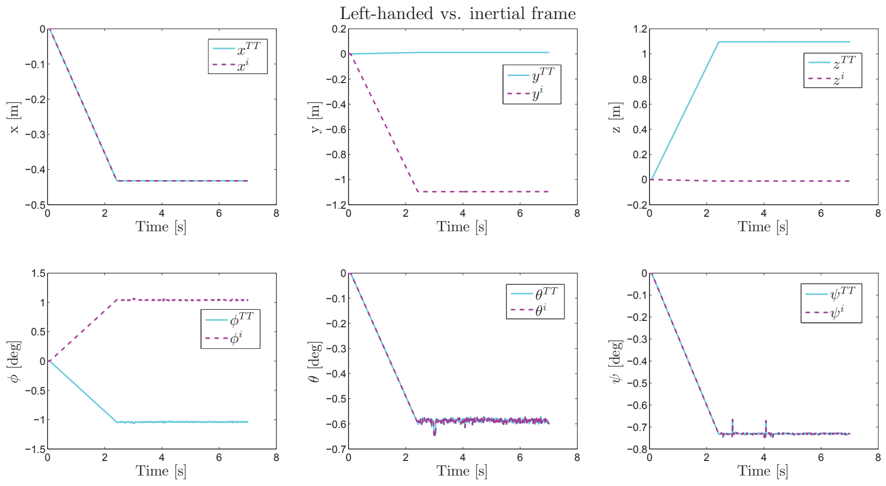

Figure 5.3: *Measurements expressed in the Tracking Tools as well as inertial frames. The blue lines represent position and orientation expressed in the TT frame, while the purple, dashed lines show the position and orientation with respect to the inertial frame.*

Time Synchronization The sampling period of the OptiTrack system as well as of the EKF is Δt, while the time spent processing the current OptiTrack measurement is denoted t_{proc}. To ensure that the next execution of the Measurement Update block begins after Δt seconds, when a new measurement is available from the camera system, the Time Synchronization block delays the system for $(\Delta t - t_{\text{proc}})$ seconds. It is assumed and tested that the processing time of a group of frames is considerably shorter than Δt. Time aspects are further discussed in the following section.

5.2 "Near Real-Time" Aspects

The estimates of the quadcopter states must be updated in a timely manner for the positioning algorithm to be useful in the proposed test setup. Delayed position estimates may cause the UAV to be erroneously directed towards obstacles due to an incorrect perception of its current position. Furthermore, the estimates must correspond to the correct sampled measurement from OptiTrack, i.e. the time stamps and sampling rates of the various components of the algorithm must be synchronized. The sampling period Δt of the camera system is the reciprocal of its sampling rate (also known as its frame rate). Hence, the sampling rate of the algorithm is determined by the chosen rate of acquiring frames in OptiTrack.

To avoid loss of data from OptiTrack, frames must be collected at least every Δt seconds. Furthermore, for each collected group of frames, the position and orientation data for the UAV must be extracted, preprocessed, and received as well as used by the EKF for state estimation. That is, the total time spent processing a group of frames cannot exceed the sampling period of the system. The processing time t_{proc} includes

$$t_{\text{proc}} = t_{\text{getMeas}} + t_{\text{getTime}} + t_{\text{UAVinFrame}} + t_{\text{getLoc,TT}} + t_{\text{getLoc,i}} + t_{\text{EKF}},$$

where t_{getMeas} is the total time required to deliver a group of frames to MATLAB, t_{getTime} is spent obtaining a time stamp for the current measurement, $t_{\text{UAVinFrame}}$ is the required time to check whether the UAV is in the current group of frames, $t_{\text{getLoc,TT}}$ is the time it takes to obtain the location of the UAV in the TT frame while $t_{\text{getLoc,i}}$ seconds is spent performing measurement preprocessing. t_{EKF} seconds is required to perform state estimation.

Finally, the processing should not be executed more than once for each measurement input, i.e. frames should not be collected more often than every Δt seconds. The demands outlined in this discussion are all fulfilled if a new group of frames is collected exactly every Δt seconds for a Δt satisfying the real-time requirement of the test setup (frequent output of estimates), and if $t_{\text{proc}} < \Delta t$. This is in accordance with the presentation of the Time Synchronization block in Figure 5.2. The sampling rate of the OptiTrack system is set to 100 FPS in this thesis, which means that measurements are updated quite frequently. Thus, if the "near real-time" aspects discussed here are satisfied, state estimates should be updated sufficiently often for the UAV to correctly perceive its position at all times (given that the estimates are correct).

That being said, MATLAB is not designed for real-time computing. A real-time system must guarantee the response to an event within a strict time limit, and such systems are often implemented using threads. MATLAB does support the use of multiple threads through the `Parallel Computing Toolbox`, but real-time analysis is not the main task of MATLAB. However, a "near real-time" implementation of the Camera Measurement Algorithm is achieved through the aspects explained here, and its performance will be evaluated in Chapter 8.

5.2.1 Assumptions and Limitations

The Camera Measurement Algorithm is designed under certain assumptions. As explained above, a "near real-time" implementation is achieved by letting the algorithm control when a new measurement is available. That is, rather than the sensor system transmitting measurements whenever new ones are detected, it is assumed that the sensors sample at a constant rate, and the positioning algorithm updates its measurements according to that sampling rate. Thus, delays or an otherwise time-varying sampling rate in the camera system are not considered in this thesis. An alternative could be a real-time implementation using a thread continuously polling for new measurements, and interrupting the algorithm when appropriate. However, the chosen approach to obtaining a positioning algorithm that works in the lab setup scenario seems justified. The sampling rate of the

OptiTrack system appears to be sufficiently high and constant for the "near real-time" implementation to achieve satisfactory performance if the processing time satisfies the limit given above.

5.3 Communication between the Camera System and Algorithm

To access the measurements obtained by the camera system presented in Chapter 3, support for communication between the two environments must be implemented. The API mentioned above, which is used to obtain and stream the necessary real-time data to Matlab, is presented in this section. The function `TT_Tools_demo`[1] has greatly inspired the OptiTrack to Matlab communication in this thesis.

5.3.1 Dynamic-Link Library

The Tracking Tools API is a set of C/C++ function calls and a *Dynamic-Link Library* (DLL), which can be loaded by the application wanting to use the API [15]. A DLL is a shared library, and can thus be accessed by several programs at the same time. Programs written in different languages can access the same DLL, and such libraries provide a general method for storing functions that are to be used by many applications.

One of the tasks performed as part of the start-up procedures in the Camera Measurement Algorithm is to load the Tracking Tools DLL `NPTrackingTools` and its header, and the library is frequently accessed throughout the algorithm. `NPTrackingTools` contains all Tracking Tools functionality which can be accessed through the C/C++ function calls mentioned above.

5.3.2 Use of the API

Before the API can be used to access position and orientation data for a trackable, a Tracking Tools *project* must be created and saved. Such a project contains defined trackables and a camera calibration result (see Section 3.3), and is stored in a .tpp file. The API provides a function for loading the .tpp file in Matlab, an action which is performed as part of the start-up procedures in the Camera Measurement Algorithm.

The main functionality of the measurement acquisition part of the algorithm includes polling for frames, detecting the UAV in the frames, and obtaining its position and orientation. The API function `TT_UpdateSingleFrame` processes a single group of camera data, triangulates, solves rigid bodies, and streams the

[1]Source: https://www.mathworks.com/matlabcentral/fileexchange/26449-tracking-tools-optitrack

tracking data to the application calling the function [15]. It must be called every Δt seconds to ensure no loss of data, as explained in Subsection 5.2. Calling `TT_UpdateSingleFrame` constitutes the main time usage of the measurement acquisition part of the Camera Measurement Algorithm.

`TT_TrackableLocation` is another API function which is called frequently. It returns the position and orientation of the selected rigid body whenever it is in the current group of frames and the data has been streamed to MATLAB by calling `TT_UpdateSingleFrame` [15]. The Tracking Tools API also provides functions for obtaining time stamps for the groups of frames as well as returning information about whether a selected trackable is in a group of frames (`TT_IsTrackableTracked`). Additionally, individual marker positions can be acquired and various camera and trackable properties may be obtained and set through the API. However, in this thesis, the three functions presented above as well as some assisting functions are sufficient for obtaining the required information from OptiTrack.

5.4 Extensions to Increase Robustness

The presentation of the Camera Measurement Algorithm thus far has considered the normal operation of the algorithm, i.e. possible deviations from the normal flow have not been addressed. In this section, the subsystems Measurement Update and Preprocessing are elaborated because several deviations from the normal flow may occur here: The measurements from OptiTrack may be lost (corresponding to the UAV being outside of the capture volume), or a connection loss between OptiTrack and MATLAB might happen. Furthermore, outliers in the measurements, i.e. receiving faulty position and/or orientation measurements, need be considered. Other subsystems in a motion control system often perform most of the active error handling. However, a positioning subsystem can identify and discard outliers as well as attempt to improve the quality of the state estimation as much as possible during measurement outages.

Figure 5.4 is a more detailed schematic of the Measurement Update and Preprocessing subsystems shown in Figure 5.2 (the corresponding segment of Figure 5.2 is shown in the upper left corner of Figure 5.4). It depicts the measurement acquisition from OptiTrack (the blue block containing "Update frame"), and the subsequent processing performed before $\mathbf{z}^{\mathrm{i}}$ is obtained and sent to the State Estimation block of Figure 5.2. The blue blocks represent actions taken by the algorithm, while the green blocks act as conditional statements for which "YES" or "NO" decide what action to be performed next. The "Unwrap angles" action corresponds to adjusting the angular limits to $\pm\infty$, as discussed in Section 5.1.

If the trackable has moved outside of the capture volume (result "NO" from the "UAV in frame?" block), it is proposed to adjust the EKF matrices $\mathbf{Q_1}$ and $\mathbf{R_1}$, which are used to achieve the desired state estimate behavior (these matrices will be presented below). During measurement outages, $\mathbf{Q}_{\mathbf{1},\text{measError}}$ and $\mathbf{R}_{\mathbf{1},\text{measError}}$ are used rather than the regular matrices. It is proposed to use the last (correctly) received measurement ($\mathbf{z}_{\text{last}}$) as input to the state estimator until a new one is available or a time limit has been reached, in combination with the adjusted EKF matrices. These are tuned to put little emphasis on the measurement because it is stale. The control system should be notified when an old measurement is

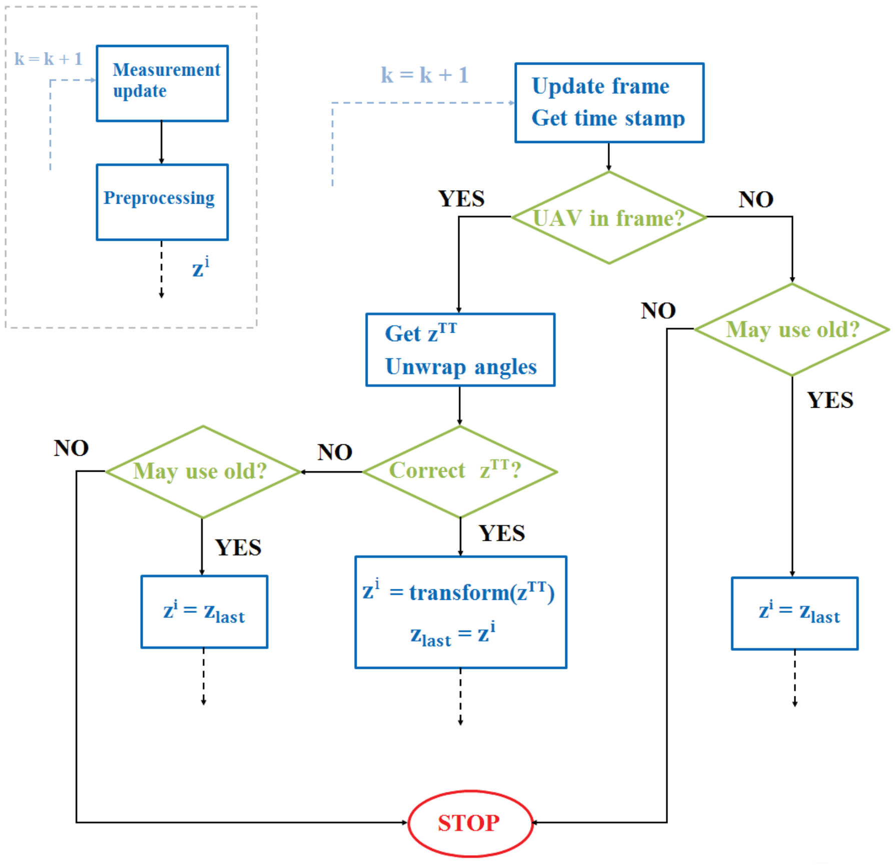

Figure 5.4: *More detailed schematic of the Measurement Update and Preprocessing subsystems (corresponding segment of Figure 5.2 shown in the upper left corner). The blue blocks represent actions taken by the positioning algorithm, while the green blocks act as conditional statements. The output of these subsystems is z^i for the current time step.*

used. If the above-mentioned time limit is reached (result "NO" from the "May use old?" blocks), the system should be aborted due to stale measurements and consequently incorrect state estimates. The same should happen if an error in the communication channel occurs, i.e. if the system fails during a call to the Tracking Tools library.

A similar approach is proposed for handling faulty measurements (result "NO" from the "Correct $\mathbf{z}^{\mathrm{TT}}$?" block). Such a measurement is simply discarded, and the previously available $\mathbf{z}$ is used in combination with the appropriate tuning matrices mentioned above ($\mathbf{Q}_{\mathbf{1},\mathrm{measError}}$ and $\mathbf{R}_{\mathbf{1},\mathrm{measError}}$) for one time step (or however long the measurements are incorrect). A measurement is characterized as an outlier if it is larger than some threshold value, i.e. deviates too much from the previous measurement. A time limit for accepting old measurements is set for this erroneous behavior as well because the algorithm is not capable of operating without new measurements indefinitely. The Camera Measurement Algorithm is implemented with account for the possibility of the errors outlined here to occur, as will be shown in the pseudocode presented in Section 5.7.

5.5 Mathematical Modelling

The intended functionality of the Camera Measurement Algorithm has now been presented, and the rest of this chapter is dedicated to modelling the system, presenting the state estimator, and discussing the performed implementation of the algorithm. It is assumed that the UAV with the trackable attached to it is a rigid body. The (kinematic) equations of motion for the Parrot AR.Drone 2.0 were derived for planar motion in the project work [17]. In this thesis, the full 6 DOF motion is considered, and the model is therefore expanded.

5.5.1 Kinematics

The following is based on [4]. To relate the time derivative of the position vector $\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}}$ to the velocity vector $\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}}$, a rotation matrix is used to describe the necessary rotation between the BODY and NED frames. For further details, see [4], Appendix A or the project work [17].

$$\mathbf{R}^{\mathrm{i}}_{\mathrm{b}}(\boldsymbol{\Theta}_{\mathrm{ib}}) = \mathbf{R}_{\mathrm{z},\psi}\mathbf{R}_{\mathrm{y},\theta}\mathbf{R}_{\mathrm{x},\phi}, \tag{5.1}$$

where $\mathbf{R}_{\mathrm{a},\alpha}$ is the *principal rotation* of α about the a axis, i.e. a one-axis rotation. Thus, the total rotation between the BODY and NED frames can be described by the matrix product of three principal rotations, one about each axis by an Euler angle. Hence,

$$\dot{\mathbf{p}}^{\mathrm{i}}_{\mathrm{b/i}} = \mathbf{R}^{\mathrm{i}}_{\mathrm{b}}(\boldsymbol{\Theta}_{\mathrm{ib}})\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}} \tag{5.2}$$

The transformation matrix $\mathbf{T}_{\Theta}(\boldsymbol{\Theta}_{\mathrm{ib}})$ is used to describe a similar relation between the time derivative of the Euler angles and the body-fixed angular velocity vector $\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}}$ (see also Appendix A):

$$\dot{\boldsymbol{\Theta}}_{\mathrm{ib}} = \mathbf{T}_{\Theta}(\boldsymbol{\Theta}_{\mathrm{ib}})\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} \tag{5.3}$$

5.5.2 Continuous System Model

Expansion of Eqs. (5.2) and (5.3), considering that

$$\mathbf{x_1} = \left[(\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}})^{\intercal}, (\boldsymbol{\Theta}_{\mathrm{ib}})^{\intercal}, (\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}})^{\intercal}, (\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}})^{\intercal}\right]^{\intercal} \tag{5.4}$$

yields the 6 DOF kinematic system model:

$$\dot{\mathbf{x}}_1 = \mathbf{f}_{\mathbf{1},\mathrm{cont}}(\mathbf{x_1}) = \tag{5.5}$$
$$\begin{bmatrix} c\psi c\theta \cdot v_{\mathrm{x}} + (c\psi s\theta s\phi - s\psi c\phi) \cdot v_{\mathrm{y}} + (s\psi s\phi + c\psi c\phi s\theta) \cdot v_{\mathrm{z}} \\ s\psi c\theta \cdot v_{\mathrm{x}} + (c\psi c\phi + s\phi s\theta s\psi) \cdot v_{\mathrm{y}} + (s\theta s\psi c\phi - c\psi s\phi) \cdot v_{\mathrm{z}} \\ -s\theta \cdot v_{\mathrm{x}} + c\theta s\phi \cdot v_{\mathrm{y}} + c\theta c\phi \cdot v_{\mathrm{z}} \\ \omega_{\mathrm{x}} + s\phi t\theta \cdot \omega_{\mathrm{y}} + c\phi t\theta \cdot \omega_{\mathrm{z}} \\ c\phi \cdot \omega_{\mathrm{y}} - s\phi \cdot \omega_{\mathrm{z}} \\ \frac{s\phi}{c\theta} \cdot \omega_{\mathrm{y}} + \frac{c\phi}{c\theta} \cdot \omega_{\mathrm{z}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In Equation (5.5), *sin* is abbreviated s and c refers to *cos*. A singularity occurs for $\theta = \pm 90^\circ = \pm\frac{\pi}{2}\mathrm{rad}$ ($\cos\theta = 0$) when using Euler angles. The control system of the UAV should ensure that this does not happen.

The kinetic equations are not considered in this thesis, and the linear and angular velocities are modelled constant in the Camera Measurement Algorithm.

5.6 Extended Kalman Filter

This section is based on theory outlined in [16].

5.6.1 Discrete Filter Model

The system will be expressed in the following general form to be used in a discrete extended Kalman filter [16]:

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \mathbf{u}(k)] + \mathbf{w}(k) \tag{5.6}$$
$$\mathbf{z}(k) = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \tag{5.7}$$

where it is assumed that the process and measurement noise $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are additive, Gaussian distributed and described through the covariance matrices $\mathbf{Q}(k)$ and $\mathbf{R}(k)$, respectively. In the Camera Measurement Algorithm, the following filter model is obtained:

$$\begin{aligned}\mathbf{x_1}(k+1) &= \mathbf{x_1}(k) + \Delta t \cdot \mathbf{f}_{\mathbf{1},\text{cont}}[\mathbf{x_1}(k)] + \mathbf{w_1}(k) \\ &= \mathbf{f_1}[\mathbf{x_1}(k)] + \mathbf{w_1}(k) \qquad (5.8)\\ \mathbf{z}(k) &= \left[x(k), y(k), z(k), \phi(k), \theta(k), \psi(k)\right]^{\intercal} + \mathbf{v_1}(k) \\ &= \mathbf{h}[\mathbf{x_1}(k)] + \mathbf{v_1}(k), \qquad (5.9)\end{aligned}$$

as presented in Eqs. (4.5) and (4.6). Discretization of the planar kinematic model was performed using forward Euler discretization in the project work, and the same method is applied in this thesis. For details, see [17].

5.6.2 Functionality and Characteristics of the EKF

The key assumptions when designing a Kalman filter is that the underlying system is linear and observable [16]. However, as discussed in the project work, an *extended* Kalman filter can be used when the underlying system is nonlinear [17]. It uses noisy measurements from a dynamical system as inputs, and outputs estimates of measured and unmeasured system states. In the case of linear system dynamics, a Kalman filter will produce optimal estimates with respect to minimum variance [16]. The EKF is an extension of this optimal state estimator, designed to account for the fact that most real systems inhabit nonlinear characteristics.

The linear as well as the extended Kalman filters are recursive processes divided in a *corrector* and a *predictor* part, where the corrector corrects its state estimates using a new measurement whenever one is available. These updated state estimates are then used by the predictor to predict the states at the next time step [17].

The diagonal covariance matrices briefly mentioned above, $\mathbf{Q}(k)$ and $\mathbf{R}(k)$, are suspect to tuning when implementing a Kalman filter. That is, these matrices represent assumptions made with regards to the system, and may be corrected if found necessary. They are therefore often referred to as the design matrices of a Kalman filter. In this thesis, the values of the design matrices are different during measurement outages to emphasize modelled behavior. $\mathbf{Q}(k)$ contains assumptions made with regards to the process noise $\mathbf{w}(k)$, while $\mathbf{R}(k)$ represents the amount of noise $\mathbf{v}(k)$ assumed in the measurements. $\mathbf{Q}(k)$ and $\mathbf{R}(k)$ do not influence the state update directly, they have an impact on the *Kalman gain* $\mathbf{K}(k)$ and the *error covariance matrix* $\mathbf{P}(k)$. A high Kalman gain makes the estimates follow the measurements closely, while a low $\mathbf{K}$ results in the filter emphasizing the modelled behavior, thus becoming less responsive to new inputs. As mentioned earlier, the linear and angular velocities are unmeasured and modelled constant in the Camera Measurement Algorithm. Through tuning of the design matrices, quite accurate estimates of these states can be obtained using an extended Kalman filter even though their true behavior is dynamical.

5.6.3 Discrete EKF Equations

The equations of a discrete extended Kalman filter are stated below [16], [17]. The state and measurement vectors refer to the general system dynamics given by Equations (5.6) and (5.7).

Design matrices:

$$\mathbf{Q}(k) = \mathbf{Q}^{\mathrm{T}}(k) > 0, \mathbf{R}(k) = \mathbf{R}^{\mathrm{T}}(k) > 0 \tag{5.10}$$

Initial conditions:

$$\bar{\mathbf{x}}(0) = \mathbf{x}_0 \tag{5.11}$$

$$\bar{\mathbf{P}}(0) = E[(\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^{\mathrm{T}}] = \mathbf{P}_0 \tag{5.12}$$

Corrector equations:

$$\mathbf{H}(k) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\bar{\mathbf{x}}(k)} \tag{5.13}$$

$$\mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{H}^{\mathrm{T}}(k)[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}^{\mathrm{T}}(k) + \mathbf{R}(k)]^{-1} \tag{5.14}$$

$$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{z}(k) - \mathbf{h}[\bar{\mathbf{x}}(k)]] \tag{5.15}$$

$$\hat{\mathbf{P}}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^{\mathrm{T}} + \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}^{\mathrm{T}}(k) \tag{5.16}$$

Predictor equations:

$$\mathbf{\Phi}(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(k)} \tag{5.17}$$

$$\bar{\mathbf{x}}(k+1) = \mathbf{f}[\hat{\mathbf{x}}(k), \mathbf{u}(k)] \tag{5.18}$$

$$\bar{\mathbf{P}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}(k)\mathbf{\Phi}^{\mathrm{T}}(k) + \mathbf{Q}(k) \tag{5.19}$$

5.7 Implementation

The Camera Measurement Algorithm is implemented for use with the camera system and UAV, i.e. for online use in the lab setup. However, in order to accurately test the state estimates against known ground truths, the algorithm is implemented for simulated measurements as well. Also, a simulation may reveal errors connected to the algorithm prior to setting up the real system. Furthermore, an offline version of the algorithm using recorded measurements from OptiTrack is implemented for comparing the results to those obtained using the Integrated Camera System/INS Algorithm for the exact same scenario. This section presents relevant details of the implementation which have not already been discussed. Pseudocode for the Camera Measurement Algorithm concludes the section.

5.7.1 Measurement Preprocessing

As mentioned earlier, OptiTrack transmits measured angles between ±180°, allowing the measurements to contain leaps whenever a limit is reached. Whenever a measurement is received from the camera system, it is processed by using the built-in MATLAB function `unwrap` with the new as well as the previously obtained correct OptiTrack measurement as inputs. If one of the angles in a measurement vector contains a ±360° leap with respect to the previous, a corresponding multiple of 360° is added or subtracted to obtain the same angle, but without the leap. The `unwrap` function seems to produce a considerably smaller, but still unacceptable leap for one time step. This smaller leap is characterized as a faulty measurement, and can thus be handled by the measurement outlier mechanism presented in Section 5.4. The result is smooth angular measurements, which subsequently are transformed into the inertial frame and used as part of the measurement input to the EKF.

5.7.2 Offline Processing

Measurements from OptiTrack can be recorded in the lab by storing the data as a .csv (Comma Separated Values) file containing virtually all information about a performed OptiTrack session. This measurement recording is performed by OptiTrack. The .csv files are read into MATLAB using the function `readtext`[2] from `Matlab Central`, and the Camera Measurement Algorithm can be executed using real measurements.

5.7.3 Preliminary Aspects: Analytical Calculation of EKF Matrices

The extended Kalman filter is used when the underlying system is nonlinear, and a linearization is required at each time step. The matrix $\mathbf{\Phi}(k)$, as can be seen in the predictor equations presented above, contains a linearized version of the system function $\mathbf{f}[\mathbf{x}(k), \mathbf{u}(k)]$. In a similar manner, the matrix $\mathbf{H}(k)$ is the Jacobian of the measurement function $\mathbf{h}[\mathbf{x}(k)]$. The functions are linearized about the best current estimate at each time step, $\hat{\mathbf{x}}(k)$ and $\bar{\mathbf{x}}(k)$, but the structures of $\mathbf{\Phi}(k)$ and $\mathbf{H}(k)$ do not change and may be calculated offline. In fact, the Jacobian of the measurement function is constant in this thesis, as is shown below:

$$\mathbf{h}[\mathbf{x_1}(k)] = \left[x(k), y(k), z(k), \phi(k), \theta(k), \psi(k)\right]^{\mathsf{T}} \implies$$

$$\mathbf{H_1} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x_1}}\right|_{\mathbf{x_1}=\bar{\mathbf{x}}_1(k)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.20}$$

[2]Source: www.mathworks.com/matlabcentral/fileexchange/10946-readtext

The analytical calculation of the $\mathbf{\Phi}(k)$ matrix for the system in the Camera Measurement Algorithm (that is, $\mathbf{\Phi_1}(k)$) is shown in Equation (5.22) below. The matrix is displayed using submatrices due to lack of space.

$$\mathbf{f_1}[\mathbf{x_1}(k)] = \tag{5.21}$$
$$\begin{bmatrix} x + \Delta t[c\psi c\theta \cdot v_x + (c\psi s\theta s\phi - s\psi c\phi) \cdot v_y + (s\psi s\phi + c\psi c\phi s\theta) \cdot v_z] \\ y + \Delta t[s\psi c\theta \cdot v_x + (c\psi c\phi + s\phi s\theta s\psi) \cdot v_y + (s\theta s\psi c\phi - c\psi s\phi) \cdot v_z] \\ z + \Delta t[-s\theta \cdot v_x + c\theta s\phi \cdot v_y + c\theta c\phi \cdot v_z] \\ \phi + \Delta t[\omega_x + s\phi t\theta \cdot \omega_y + c\phi t\theta \cdot \omega_z] \\ \theta + \Delta t[c\phi \cdot \omega_y - s\phi \cdot \omega_z] \\ \psi + \Delta t[\frac{s\phi}{c\theta} \cdot \omega_y + \frac{c\phi}{c\theta} \cdot \omega_z] \\ v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \Longrightarrow$$

$$\mathbf{\Phi_1}(k) = \left.\frac{\partial \mathbf{f_1}}{\partial \mathbf{x_1}}\right|_{\mathbf{x_1}=\hat{\mathbf{x}}_\mathbf{1}(k)} = \begin{bmatrix} \mathbf{\Phi_{1}}_{,1} & \mathbf{\Phi_{1}}_{,2} & \mathbf{\Phi_{1}}_{,3} \\ \mathbf{\Phi_{1}}_{,4} & \mathbf{\Phi_{1}}_{,5} & \mathbf{\Phi_{1}}_{,6} \\ \mathbf{\Phi_{1}}_{,7} & \mathbf{\Phi_{1}}_{,8} & \mathbf{\Phi_{1}}_{,9} \end{bmatrix}_{\mathbf{x_1}=\hat{\mathbf{x}}_\mathbf{1}(k)}, \tag{5.22}$$

where the elements of $\mathbf{\Phi_1}(k)$ are 4×4 matrices, and can be found in Appendix C. The design matrices are assumed constant during normal flow, i.e. $\mathbf{Q_1}(k) = \mathbf{Q_1}$ and $\mathbf{R_1}(k) = \mathbf{R_1}$. However, the weighting is somewhat different during measurement outages, as mentioned above. That is, two different, constant versions of $\mathbf{Q_1}$ and $\mathbf{R_1}$ are used.

5.7.4 Pseudocode

Pseudocode for the Camera Measurement Algorithm is shown in Algorithm 1 on the next page. In the pseudocode, the state estimation is highlighted as a *procedure* in which calculation of the various EKF matrices at each time step are performed according to Equations (5.13)-(5.19) by calling functions (e.g. `calc_K`, see Algorithm 1). The pseudocode, like the implemented algorithm, accounts for the measures taken to increase robustness, as opposed to Figure 5.2, which depicts the main functionality of the algorithm. The arguments k are omitted from the variables and functions for readability.

The `if` statement in line 5 of the pseudocode (which ends the algorithm if a communication failure has occurred) is meant to account for connection loss during all function calls to the Tracking Tools library.

Algorithm 1 Camera Measurement Algorithm

```
 1: initialize(library, API, UAVtrackable, project)
 2: set($\hat{\mathbf{x}}_\mathbf{1}(0)$, $\hat{\mathbf{P}}_\mathbf{1}(0)$, $\bar{\mathbf{x}}_\mathbf{1}(0)$, $\bar{\mathbf{P}}_\mathbf{1}(0)$, $\Delta t$, timeSteps, $\mathbf{z}^{\mathrm{TT}}(0)$, $\mathbf{z}^{\mathrm{i}}(0)$, $\mathbf{z}^{\mathrm{i}}_{\mathrm{last}}(0)$,
    $\mathbf{z}^{\mathrm{TT}}_{\mathrm{last}}(0)$, $\mathbf{Q}_\mathbf{1}$, $\mathbf{R}_\mathbf{1}$, $\mathbf{Q}_{\mathbf{1},\mathrm{measError}}$, $\mathbf{R}_{\mathbf{1},\mathrm{measError}}$, stepsNoMeasurement(0),
    noMeasLimit, outlierLimit)
 3: k = 0
 4: while k < timeSteps do
 5:     if communicationFailure then
 6:         end while
 7:     end if
 8:     increment(k)
 9:     tStart = get(time)
10:     update_frame
11:     if is_tracked(UAVtrackable) then
12:         $\mathbf{z}^{\mathrm{TT}}$ = get_measurement(UAVtrackable)
13:         if $\mathbf{z}^{\mathrm{TT}}$ > outlierLimit then
14:             if stepsNoMeasurement < noMeasLimit then
15:                 goTo(line 36)
16:             else
17:                 end while
18:             end if
19:         else
20:             reset(stepsNoMeasurement)
21:             $\mathbf{z}^{\mathrm{TT}}_{\mathrm{last}} = \mathbf{z}^{\mathrm{TT}}$
22:             $\mathbf{z}^{\mathrm{i}}$ = transform($\mathbf{z}^{\mathrm{TT}}$)
23:             $\mathbf{z}^{\mathrm{i}}_{\mathrm{last}} = \mathbf{z}^{\mathrm{i}}$
24:             procedure STATE__ESTIMATION($\mathbf{z}^{\mathrm{i}}$, $\mathbf{Q}_\mathbf{1}$, $\mathbf{R}_\mathbf{1}$, $\mathbf{H}_\mathbf{1}$, $\hat{\mathbf{x}}_\mathbf{1}$, $\hat{\mathbf{P}}_\mathbf{1}$, $\bar{\mathbf{x}}_\mathbf{1}$, $\bar{\mathbf{P}}_\mathbf{1}$)
25:                 $\mathbf{K}_\mathbf{1}$ = calc_K($\bar{\mathbf{P}}_\mathbf{1}$, $\mathbf{H}_\mathbf{1}$, $\mathbf{R}_\mathbf{1}$)
26:                 $\mathbf{h}(\bar{\mathbf{x}}_\mathbf{1}) = \bar{\mathbf{x}}_\mathbf{1}(1:6)$
27:                 $\hat{\mathbf{x}}_\mathbf{1}$ = calc_x̂($\mathbf{K}_\mathbf{1}$, $\mathbf{z}^{\mathrm{i}}$, $\mathbf{h}(\bar{\mathbf{x}}_\mathbf{1})$)
28:                 $\hat{\mathbf{P}}_\mathbf{1}$ = calc_P̂($\mathbf{K}_\mathbf{1}$, $\mathbf{H}_\mathbf{1}$, $\bar{\mathbf{P}}_\mathbf{1}$, $\mathbf{R}_\mathbf{1}$)
29:                 $\bar{\mathbf{x}}_\mathbf{1}$ = calc_f($\hat{\mathbf{x}}_\mathbf{1}$)
30:                 $\Phi_\mathbf{1}$ = calc_Φ($\hat{\mathbf{x}}_\mathbf{1}$)
31:                 $\bar{\mathbf{P}}_\mathbf{1}$ = calc_P̄($\Phi_\mathbf{1}$, $\hat{\mathbf{P}}_\mathbf{1}$, $\mathbf{Q}_\mathbf{1}$)
32:                 return $\hat{\mathbf{x}}_\mathbf{1}$
33:             end procedure
34:         end if
35:     else if stepsNoMeasurement < noMeasLimit then
36:         increment(stepsNoMeasurement)
37:         STATE__ESTIMATION($\mathbf{z}^{\mathrm{i}}_{\mathrm{last}}$, $\mathbf{Q}_{\mathbf{1},\mathrm{measError}}$, $\mathbf{R}_{\mathbf{1},\mathrm{measError}}$, $\mathbf{H}_\mathbf{1}$, $\hat{\mathbf{x}}_\mathbf{1}$, $\hat{\mathbf{P}}_\mathbf{1}$,
38:         $\bar{\mathbf{x}}_\mathbf{1}$, $\bar{\mathbf{P}}_\mathbf{1}$)
39:     else
40:         end while
41:     end if
42:     tStop = get(time)
43:     sleep($\Delta t$ - (tStop - tStart))
44: end while
45: shutdown(all)
```

Chapter 6

Integrated Camera System/INS Algorithm

The Integrated Camera System/INS Algorithm is an expanded version of the Camera Measurement Algorithm, i.e. the two consist of the same overall components: scene measurement acquisition, preprocessing of the measurements, and state estimation. The features which are common to both algorithms will not be repeated here. For this reason, some of the aspects of the Integrated Camera System/INS Algorithm are explained by referring to the presentation of the Camera Measurement Algorithm in Chapter 5.

The two proposed indoor positioning algorithms differ with respect to available sensors, and the implications this has on the setup of the extended Kalman filter used for state estimation. The algorithm presented in this chapter utilizes measurements from both the camera system (Ch. 3) and on-board inertial sensors. Thus, the Integrated Camera System/INS Algorithm is an *integrated solution* in which the extended Kalman filter is used as an *integration filter* by exploiting its sensor fusion feature. The algorithm is inspired by GPS/INS integration, which is a widely used technique for obtaining accurate position estimates in large area navigation systems [16] and by the system presented in [6].

The first section of this chapter is an introduction to the theory of inertial navigation, while relevant aspects of measurement integration theory are presented in Section 6.2. An overview of the Integrated Camera System/INS Algorithm is given in Section 6.3, followed by an updated system model incorporating the available sensors and their impacts on the system in Section 6.4. Finally, the chosen EKF sensor fusion structure is stated (Section 6.5) and implementation aspects are elaborated.

6.1 Inertial Navigation Systems

In the Integrated Camera System/INS Algorithm, inertial sensors measuring linear acceleration and angular velocity are used in the process of obtaining accurate and timely pose estimates for the Parrot AR.Drone 2.0. More specifically, a strapdown inertial system is considered. As mentioned in Section 2.3, such a system is attached to the object of interest and moves with it [16]. The system consists of an inertial measurement unit for obtaining the required measurements, and differential equations (the strapdown equations) for computing position, orientation, and velocity from the measurements.

The measurement part of an inertial navigation system, the IMU, is made up of an *Inertial Sensor Assembly* (ISA) and hardware/low level software to interface the ISA [16]. A schematic of the various layers and their connection in an INS is depicted in Figure 6.1.

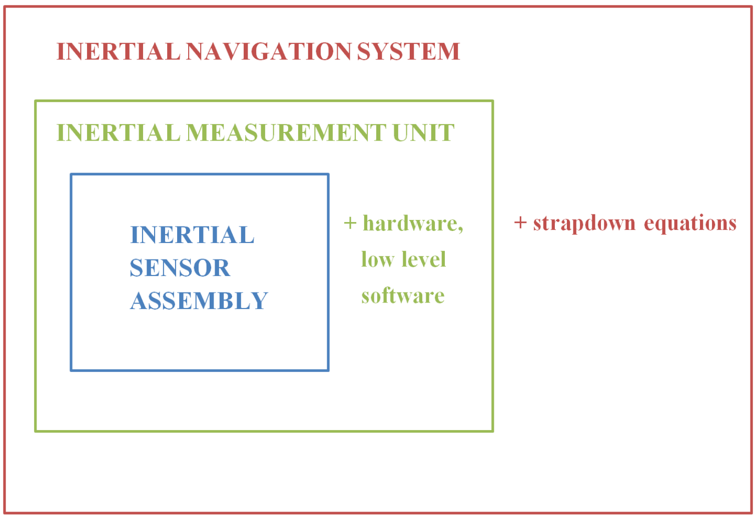
INERTIAL NAVIGATION SYSTEM
INERTIAL MEASUREMENT UNIT
INERTIAL SENSOR ASSEMBLY
+ hardware, low level software
+ strapdown equations

Figure 6.1: *Figure inspired by [16]. Schematic of a strapdown INS. The outermost (red) rectangle represents the whole inertial navigation system, which consists of an IMU and the strapdown equations. The green rectangle represents the IMU, which consists of an ISA as well as hardware and low level software. The smallest, blue rectangle represents the ISA, which is a collection of three accelerometers and three gyroscopes.*

6.1.1 Inertial Sensor Assembly

The inertial sensor assembly is the collection of sensors found in an INS (see Figure 6.1). It consists of three accelerometers and three gyroscopes for measuring the three components of linear acceleration and of angular velocity, respectively.

Accelerometers Accelerometers measure *specific force* $\mathbf{a}^{\mathrm{b}}_{\mathrm{meas}}$, i.e. the acceleration relative to free fall, decomposed in the *b* frame. The acceleration $\mathbf{a}^{\mathrm{b}}_{\mathrm{b/i}}$ of the object of interest is the quantity in which one is interested. The two are related as follows [16]:

When the platform to which accelerometers are attached is at rest, the specific force measurements made by the accelerometers will be

$$\mathbf{a}^{\mathrm{b}}_{\mathrm{meas}} = -\mathbf{R}^{\mathrm{b}}_{\mathrm{i}}(\boldsymbol{\Theta}_{\mathrm{ib}})\mathbf{g}^{\mathrm{i}}, \tag{6.1}$$

where

$$\mathbf{g}^{\mathrm{i}} = \left[0, 0, \mathrm{g}\right]^{\mathsf{T}} = \left[0, 0, 9.81\mathrm{m/s}^2\right]^{\mathsf{T}} \tag{6.2}$$

is the gravity due to the mass of the Earth (assumed constant). During acceleration, the measurements will be

$$\mathbf{a}^{\mathrm{b}}_{\mathrm{meas}} = \mathbf{a}^{\mathrm{b}}_{\mathrm{b/i}} - \mathbf{R}^{\mathrm{b}}_{\mathrm{i}}(\boldsymbol{\Theta}_{\mathrm{ib}})\mathbf{g}^{\mathrm{i}}, \tag{6.3}$$

where

$$\mathbf{a}^{\mathrm{b}}_{\mathrm{b/i}} = \left[a_{\mathrm{x}}, a_{\mathrm{y}}, a_{\mathrm{z}}\right]^{\mathsf{T}} \tag{6.4}$$

is the acceleration of the platform (the NED frame is assumed inertial). Hence, gravity compensation must be included in the velocity dynamics.

The output from the accelerometers, $\mathbf{a}^{\mathrm{b}}_{\mathrm{IMU}}$, is often noticeably influenced by biases, especially with low-cost sensors, as is the case in this thesis. Thus, the accelerometer output model accounts for biases $\mathbf{b}^{\mathrm{b}}_{\mathrm{acc}}$ as well as sensor measurement noise $\mathbf{w}^{\mathrm{b}}_{\mathrm{acc}}$ [4]:

$$\begin{aligned}\mathbf{a}^{\mathrm{b}}_{\mathrm{IMU}} &= \mathbf{a}^{\mathrm{b}}_{\mathrm{meas}} + \mathbf{b}^{\mathrm{b}}_{\mathrm{acc}} + \mathbf{w}^{\mathrm{b}}_{\mathrm{acc}} \\ &= \left[a_{\mathrm{IMU,x}}, a_{\mathrm{IMU,y}}, a_{\mathrm{IMU,z}}\right]^{\mathsf{T}},\end{aligned} \tag{6.5}$$

where $\mathbf{w}^{\mathrm{b}}_{\mathrm{acc}}$ is modelled as additive zero-mean measurement noise, and bias modelling is discussed as part of the system model in Section 6.4.

Gyroscopes Gyroscopes measure angular velocity of the BODY frame relative to the Earth Centered Inertial (ECI) frame, decomposed in the BODY frame; $\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/ECI}}$ [16]. However, in this thesis, the NED frame is assumed inertial (with a fixed position relative to the Earth), and the Earth rotation is neglected. Thus, the approximation of assuming that the gyroscopes measure $\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}}$ is made. The assumptions are further discussed below. The noise-free measurements made by the three gyroscopes are

$$\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} = \left[\omega_{\mathrm{x}}, \omega_{\mathrm{y}}, \omega_{\mathrm{z}}\right]^{\mathsf{T}} \tag{6.6}$$

Similarly to the accelerometers described above, the output from the gyroscopes, $\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{IMU}}$, is often influenced by biases. Thus, the gyroscope output model accounts for biases $\mathbf{b}^{\mathrm{b}}_{\mathrm{gyro}}$ as well as sensor measurement noise $\mathbf{w}^{\mathrm{b}}_{\mathrm{gyro}}$ [4]:

$$\begin{aligned}\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{IMU}} &= \boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} + \mathbf{b}^{\mathrm{b}}_{\mathrm{gyro}} + \mathbf{w}^{\mathrm{b}}_{\mathrm{gyro}} \\ &= \left[\omega_{\mathrm{IMU,x}}, \omega_{\mathrm{IMU,y}}, \omega_{\mathrm{IMU,z}}\right]^{\mathsf{T}},\end{aligned} \tag{6.7}$$

where $\mathbf{w}^{\mathrm{b}}_{\mathrm{gyro}}$ is modelled as additive zero-mean measurement noise.

The biases and measurement noise mentioned above are not the only errors present in measurements from IMUs. Scale-factor and misalignment errors are also part of the output. However, for low dynamic applications, compensation of bias error is sufficient to achieve satisfactory performance [16]. Thus, bias estimation is included in the Integrated Camera System/INS Algorithm.

6.1.2 Initialization

Before reliable positioning data can be obtained from an inertial navigation system, initialization of the system need be performed. In [16], three steps are outlined: Position and velocity initialization, Attitude initialization (known as alignment), and Instrument calibration. The platform utilized in this thesis performs the attitude initialization automatically when the battery is connected to the Parrot AR.Drone 2.0, while the instrument calibration is performed through bias estimation in the integrated scheme of the Integrated Camera System/INS Algorithm. The initial position is known from the camera system measurements, while the velocity is initially zero.

6.1.3 Velocity Strapdown Equations

It has been shown in [16] that the velocity dynamics in a strapdown inertial navigation system may be written

$$\dot{\mathbf{v}}^{\mathrm{i}} = \mathbf{a}^{\mathrm{i}}_{\mathrm{meas}} + \mathbf{g}^{\mathrm{i}} - [2\mathbf{S}(\boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{E/ECI}}) + \mathbf{S}(\boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{i/E}})]\mathbf{v}^{\mathrm{i}}, \tag{6.8}$$

where E and ECI are the Earth Centered Earth Fixed (ECEF) and ECI coordinate systems, respectively. These two frames both have their origins in the center of the Earth, but the axes of the ECI frame point in the same directions regardless of the Earth rotation, while the ECEF frame is fixed to the Earth and rotates with it [16]. In addition, $\mathbf{S}$ in Equation (6.8) is a skew-symmetric matrix (see Appendix B).

$\boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{E/ECI}}$ represents the rate of the Earth rotation (called turn rate) expressed in the NED frame, while $\boldsymbol{\omega}^{\mathrm{i}}_{\mathrm{i/E}}$ is the function of change of the NED frame with respect to ECEF (also called transport rate) [16]. The effects of these rates must be accounted for when navigating in large areas and at high speeds. However, in this thesis, a small UAV will be operating at low speeds in a relatively small lab room. Thus, the NED frame (i) can be assumed inertial (with a fixed position relative to the Earth) and the Earth rotation can be neglected [4], [6]. Hence, both the turn and the transport rates are neglected in this thesis. The resulting velocity dynamics are given by

$$\begin{aligned}
\dot{\mathbf{v}}^{\mathrm{i}}_{\mathrm{b/i}} &= \mathbf{a}^{\mathrm{i}}_{\mathrm{meas}} + \mathbf{g}^{\mathrm{i}} \\
&= \mathbf{R}^{\mathrm{i}}_{\mathrm{b}}(\mathbf{\Theta}_{\mathrm{ib}})[\mathbf{a}^{\mathrm{b}}_{\mathrm{IMU}} - \mathbf{b}^{\mathrm{b}}_{\mathrm{acc}} - \mathbf{w}^{\mathrm{b}}_{\mathrm{acc}}] + \mathbf{g}^{\mathrm{i}} \\
&\Updownarrow \\
\dot{\mathbf{v}}^{\mathrm{b}}_{\mathrm{b/i}} &= \mathbf{a}^{\mathrm{b}}_{\mathrm{IMU}} - \mathbf{b}^{\mathrm{b}}_{\mathrm{acc}} - \mathbf{w}^{\mathrm{b}}_{\mathrm{acc}} + \mathbf{R}^{\mathrm{b}}_{\mathrm{i}}(\mathbf{\Theta}_{\mathrm{ib}})\mathbf{g}^{\mathrm{i}} \qquad (6.9)
\end{aligned}$$

In this thesis, the velocity strapdown equations are added to the UAV dynamics in the system model. The position and attitude are already modelled (in the Camera Measurement Algorithm, Section 5.5), while bias modelling will be discussed in Section 6.4. The velocity strapdown equations provide a connection between the acceleration measurements from the IMU and the resulting motion.

6.2 GPS/INS Integration

The Integrated Camera System/INS Algorithm seeks to combine measurements obtained by the camera system with inertial sensor outputs from the quadcopter. The intention is to provide redundancy in the sensors while achieving equally frequent state estimate updates with respect to the results obtained using the camera system only. Two separate measurement sources may provide continued satisfactory estimation through short outages in one of the sensor systems. As mentioned earlier, the utilized measurement integration scheme is greatly inspired by GPS/INS integration. That is, the GPS measurements are replaced by image information in this thesis, which is more suitable for indoor use.

GPS and INS are often combined to provide an integrated navigation system for wide-area operation (see also Sec. 2.3). The integration is performed to exploit the individual advantages of each system, i.e. the high output rate and excellent short-term accuracy of INS is combined with the long-term accuracy of GPS [16]. Furthermore, should one of the systems experience an error, the other may ensure continued navigation. In [16], several architectures for GPS/INS integration are presented which differ in terms of level of coupling and choice of variables. This section is based on theory based in [16], and presents the most common GPS/INS architectures in order to provide background theory on the subject of measurement integration before the chosen integration scheme in this thesis is presented in the next section.

As shown in Figure 6.2, an integrated GPS/INS system consists of three subsystems; the GPS, the INS, and the integration filter. Several schemes for implementing the integration exist, known as integration architectures. The most common ones are *uncoupled*, *loosely coupled*, *tightly coupled*, and *deeply coupled* integration.

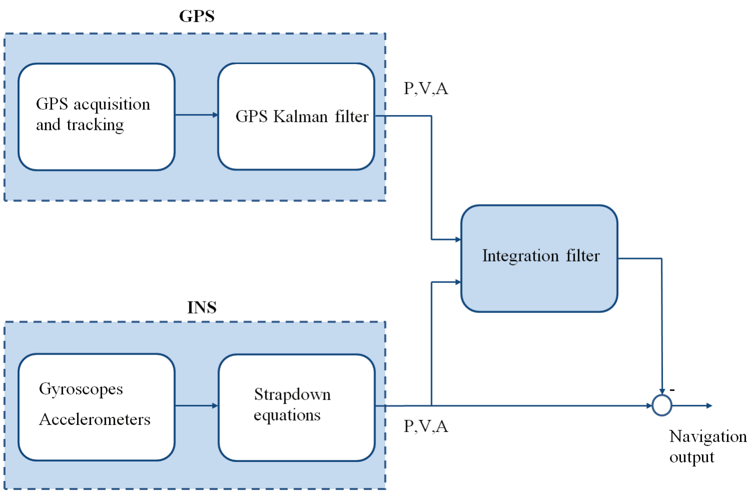

Figure 6.2: *Figure reproduced from [16]. An uncoupled GPS/INS integration architecture is depicted. In the figure, P, V, and A refer to position, velocity, and attitude. A large amount of measurement processing is performed in the GPS and INS subsystems, resulting in position, velocity, and attitude being the input variables to the integration filter.*

As the names suggest, the differences between the setups are connected to amount of individual processing performed in the GPS and INS subsystems before data is sent to the integration filter, and what information is fed back to the measurement subsystems. That is, the level of coupling. In an uncoupled integration architecture (illustrated in Figure 6.2), position, velocity, and attitude information is sent from both measurement subsystems to the integration filter, i.e. a large amount of measurement processing is performed in each subsystem individually. As the figure suggests, there is no information feedback from the integration filter. An advantage of the uncoupled scheme is that a high degree of redundancy is obtained, and errors in a subsystem may be tolerated.

At the other outer limit, the integration filter performs virtually all processing in a deeply coupled architecture. Such a scheme is often implemented using *direct* variables, referring to the estimation of the whole states in the integration filter. The other possibility is using *indirect* variables, an approach in which error states are estimated. Indirect variables are often used in uncoupled and loosely coupled architectures.

Between the two boundaries, loosely and tightly coupled integration also represent quite different systems. A tightly coupled GPS/INS system uses raw measurements instead of position, velocity, and attitude as inputs to the filter, and is illustrated in Figure 6.3. Hence, the GPS and strapdown equations need be implemented in the integration filter when using this integration architecture.

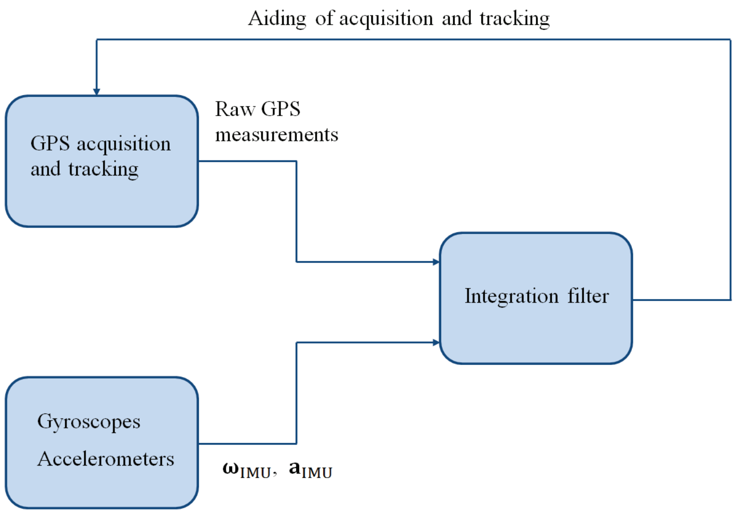
Aiding of acquisition and tracking
GPS acquisition and tracking
Raw GPS measurements
Integration filter
Gyroscopes
Accelerometers
$\boldsymbol{\omega}_{\mathrm{IMU}}$, $\mathbf{a}_{\mathrm{IMU}}$

Figure 6.3: *Figure reproduced from [16]. A tightly coupled GPS/INS integration architecture is depicted. In the figure, $\boldsymbol{\omega}_{IMU}$ and $\boldsymbol{a}_{IMU}$ represent the raw gyroscope and accelerometer measurements.*

In a loosely coupled integration, preprocessed measurements of position, velocity, and attitude are used as integration filter inputs, similarly to the uncoupled scheme outlined in Figure 6.2. The difference between the uncoupled and loosely coupled architectures is the use of feedback. Several possibilities for feedback are present in the loosely coupled scheme. Of relevance in this thesis is the *reset* feedback, which provides calibration of the INS using error feedback to the strapdown equation computation.

6.3 Overview of the Second Positioning Algorithm

This section provides an overview of the Integrated Camera System/INS Algorithm utilizing the theory outlined in the previous two sections as well as the presentation of the Camera Measurement Algorithm in Chapter 5. The main extension from the Camera Measurement Algorithm is the use of inertial sensor measurements. As a consequence, synchronization and communication between the OptiTrack system, the computer running the algorithm and the IMU mounted on the UAV is required and discussed in this section. Furthermore, the normal flow of the Integrated Camera System/INS Algorithm as well as the handling of possible deviations from it due to various system errors are presented. The "near real-time", system modelling, and state estimation aspects are updated with respect to Chapter 5 to account for the new system setup, before the implementation is discussed in Section 6.6.

The integration architecture chosen in the Integrated Camera System/INS Algorithm is shown in Figure 6.4. The architecture is a combination of an uncoupled and a tightly coupled GPS/INS scheme, as presented above. That is, the Camera measurement subsystem, which replaces the GPS, provides processed measurements ($\mathbf{p}_{b/i}$ and $\mathbf{\Theta}_{ib}$), while raw IMU measurements ($\mathbf{a}_{\mathrm{IMU}}$ and $\boldsymbol{\omega}_{\mathrm{IMU}}$) are used. No information is fed back to the measurement subsystems, but the inertial sensor biases are estimated in the filter and used to limit the influence of these errors on the estimates. This may be thought of as a version of the *reset* feedback mentioned in Section 6.2.

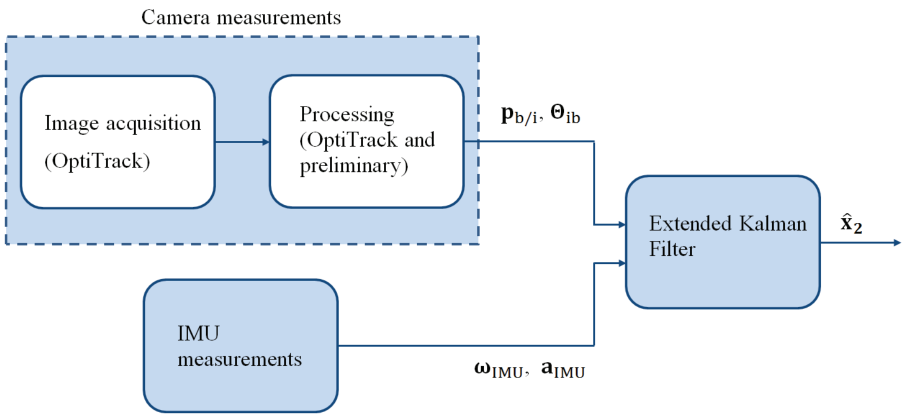
Camera measurements
Image acquisition (OptiTrack)
Processing (OptiTrack and preliminary)
$\mathbf{p}_{b/i}$, $\mathbf{\Theta}_{ib}$
Extended Kalman Filter
$\hat{\mathbf{x}}_2$
IMU measurements
$\boldsymbol{\omega}_{\mathrm{IMU}}$, $\mathbf{a}_{\mathrm{IMU}}$

Figure 6.4: *Figure inspired by GPS/INS integration figures in [16]. A high-level illustration of the Integrated Camera System/INS Algorithm is shown, with the two measurement subsystems to the left, and the EKF to the right. The camera measurement subsystem consists of image acquisition and processing, while raw measurements are transmitted from the inertial sensors.*

The measurements are fused using an extended Kalman filter, which, in addition to the kinematic equations for position and attitude dynamics as well as bias estimation, performs computation of the velocity strapdown equations presented in Section 6.1. The outputs from the integration filter ($\hat{\mathbf{x}}_2$) are full state estimates, i.e. direct variables are used. The choice of integration architecture and overall structure is based on published papers utilizing the same scheme for similar positioning systems fusing video and inertial measurements, [6] in particular, and it is greatly inspired by [16]. Using position, orientation, angular velocity, and linear acceleration measurements fits nicely with a somewhat altered version of the kinematic system model presented in Chapter 5 for the Camera Measurement Algorithm, as will be shown below.

6.3.1 Communication and Synchronization with the UAV

To use the inertial measurements from the on-board sensors of the UAV, communication between the computer running the Integrated Camera System/INS Algorithm and the Parrot AR.Drone 2.0 is needed. This task is performed through the use of a remote computer running Linux, which communicates with the UAV directly, and transmits the required navigation data to the local computer (running the positioning algorithm). The remote computer is connected to the UAV through WiFi, while the connection between the computers is obtained using wired Transmission Control Protocol/Internet Protocol (TCP/IP). In the positioning algorithm implemented in this thesis, the TCP/IP connection request from the remote computer is accepted, and messages containing the inertial measurements are received online. The rest of the required communication is performed outside of the algorithm implemented in this thesis.

Furthermore, the two sensor systems need be synchronized to ensure that the measurements from the systems comply. That is, the coordinate systems assumed by the two need be synchronized such that the BODY frame of the UAV when all angles are zero coincides with the inertial frame to which camera measurements are referenced (see Figure 4.1). This is accomplished by placing the UAV accordingly, and connecting the battery. As mentioned earlier, the attitude initialization of the IMU is performed automatically on start-up.

6.3.2 Main Functionality of the Integrated Camera System/INS Algorithm

The overall flow of the Integrated Camera System/INS Algorithm is quite similar to the Camera Measurement Algorithm due to the purposes of the positioning algorithms being identical - obtaining accurate position, orientation, and velocity estimates for the UAV. Thus, Figure 5.2 represents an overall view of the main functionality of the Integrated Camera System/INS Algorithm as well. However, the second positioning algorithm of this thesis considers measurements from two separate sensor systems which require individual start-up procedures and measurement update schemes. Furthermore, the setup of the state estimator is altered because of the measurement integration performed.

A more detailed schematic of the normal flow of the algorithm (not accounting for deviations due to errors) is shown in Figure 6.5. In the figure, the gray block in the upper left corner represents the part of the inertial measurement update *not* performed by the algorithm developed in this thesis. That is, the WiFi communication with the UAV and the *sending* side of the TCP/IP connection, as explained above. The start-up procedures for the inertial sensors (left side of Figure 6.5) include establishing the TCP/IP connection between two computers as well as initialization of required variables. This is performed in the upper part of the gray as well as the topmost, left blue block (above the respective dashed lines), while the sending and receiving of navdata is executed every 4th iteration (when the "Available measurement?" block evaluates to "YES"). In between receiving updated navdata, the last received $\mathbf{u}_{\text{IMU}}$ is used. This will be further explained in Subsection 6.3.4.

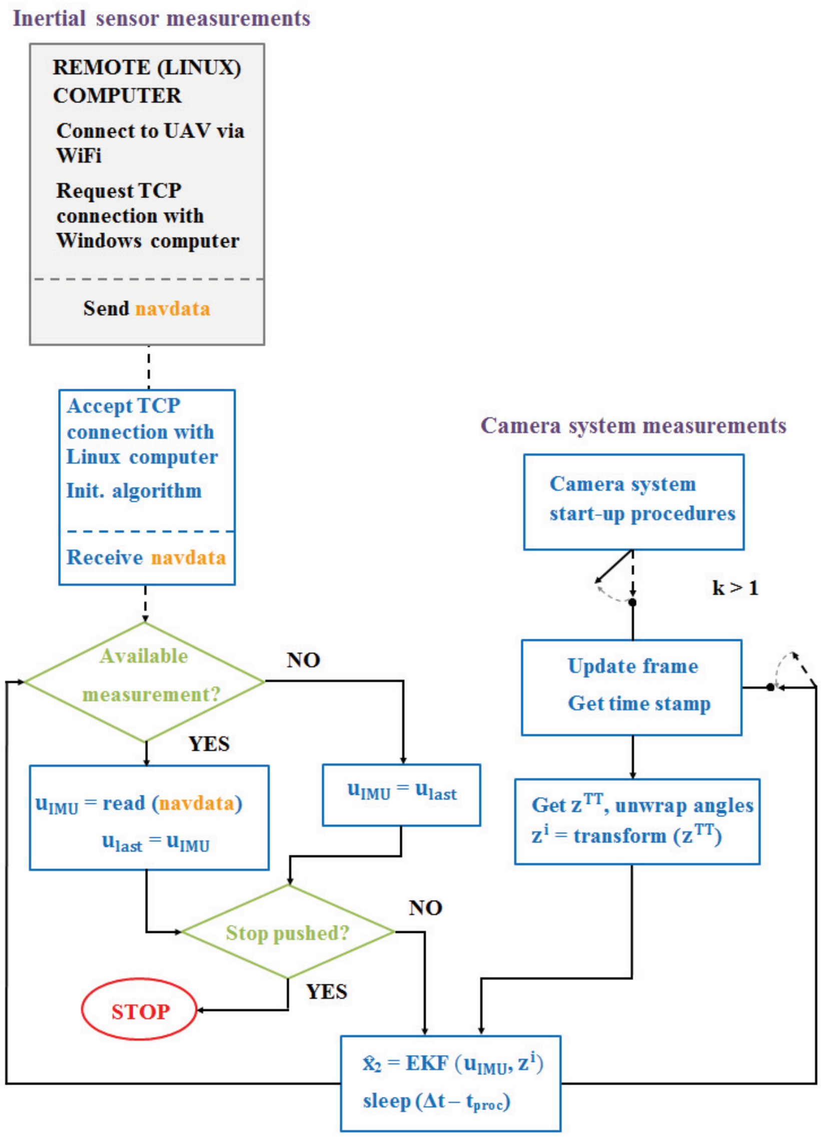
Inertial sensor measurements
REMOTE (LINUX) COMPUTER
Connect to UAV via WiFi
Request TCP connection with Windows computer
Send navdata
Accept TCP connection with Linux computer
Init. algorithm
Receive navdata
Available measurement?
NO
YES
u_{IMU} = read (navdata)
$u_{last} = u_{IMU}$
$u_{IMU} = u_{last}$
Stop pushed?
NO
YES
STOP
Camera system measurements
Camera system start-up procedures
k > 1
Update frame
Get time stamp
Get z^{TT}, unwrap angles
z^i = transform (z^{TT})
$\hat{x}_2$ = EKF (u_{IMU}, z^i)
sleep ($\Delta t - t_{proc}$)

Figure 6.5: *Main functionality of the Integrated Camera System/INS Algorithm (no deviations from normal flow depicted). The inertial measurement part is shown to the left, while the camera measurement subsystem is depicted to the right. The green blocks act as conditional statements for which "YES" or "NO" decide what action to be performed next, while the blue blocks are actions taken. The gray block contains functionality not performed in this thesis. The lower block represents the state estimator and required synchronization of time.*

As shown in Figure 6.5, the navigation data (navdata) from the IMU is stored in the variable $\mathbf{u_{IMU}}$ whenever available, and used as input to the EKF along with the measurements from OptiTrack ($\mathbf{z}^{\mathrm{i}}$). The modules of the Camera Measurement Algorithm presented in Section 5.1 are performed in the Integrated Camera System/INS Algorithm as well (right side of Fig. 6.5). The algorithm is aborted when a "stop" indicator (which is the response to a user keyboard input) is received via TCP/IP from the Linux program.

6.3.3 Deviations from the Normal Flow

Similarly to the implementation of the Camera Measurement Algorithm presented in Ch. 5, failure in various components of the Integrated Camera System/INS Algorithm have been accounted for. For the OptiTrack subsystem, the same possible errors are considered: Measurement loss, communication loss, and faulty measurements. The deviations from the normal flow are handled in the manner presented in Section 5.4 (Fig. 5.4), aided by the now available IMU measurements, which influence the weighting in the $\mathbf{Q}_{\mathbf{2},\mathrm{measError}}$ and $\mathbf{R}_{\mathbf{2},\mathrm{measError}}$ matrices.

Similarly, the inertial sensor subsystem components may fail, i.e. the IMU measurements may be lost, the TCP/IP communication may fail, and the received measurements may contain outliers. Measurement outages from the inertial sensors are handled by using the last correctly received measurement until a new one is obtained or a time limit has been reached, while failure in the TCP/IP connection results in the algorithm being aborted. Outliers are identified and discarded, analogously to the actions taken when receiving incorrect camera system measurements (see Section 5.4). That is, the extensions to increase robustness in the Integrated Camera System/INS Algorithm are similar to the aspects discussed for the Camera Measurement Algorithm, and are not elaborated further.

6.3.4 "Near Real-Time" Aspects Revisited

Because the algorithm presented in this chapter fuses measurements from two separate sources, the "near real-time" aspects outlined for the Camera Measurement Algorithm need be updated. Now, the synchronization and processing of data from two sensor systems is performed, while the overall aim remains the same: Timely (and correct) updates of the state estimates.

The sampling rate of the IMU is considerably less frequent than that of the camera system, i.e. $\Delta t_{\mathrm{TT}} = 0.01$ s while $\Delta t_{\mathrm{IMU}} \simeq 0.04$ s. The sampling interval of the extended Kalman filter is set equal to $\Delta t_{\mathrm{TT}} = \Delta t$. Thus, it is assumed that all required processing is performed within this time limit; $t_{\mathrm{proc}} < \Delta t$. Hence, in an error-free running of the Integrated Camera System/INS Algorithm, IMU measurement updates are performed once for every 4^{th} time step, while camera system measurements are updated every time step. In between available IMU measurement updates, the last updated acceleration and angular velocity measurements are used along with updated camera system measurements. A similar mechanism for synchronization of time as in the Camera Measurement

Algorithm is used, i.e. a `sleep` block ensures that each execution of the camera system measurement update is performed after $\Delta t - t_{\text{proc}}$ seconds, which subsequently implies the update of the inertial sensor measurements every 0.04 seconds.

This "near real-time" discussion reveals that similar assumptions as in the Camera Measurement Algorithm (Subsection 5.2.1) have been made with regards to the sensor systems: The camera system is still assumed to deliver new measurements every 0.01 seconds, while the inertial measurements are assumed to be updated every 0.04 seconds. Furthermore, the algorithm polls for new measurements according to these sampling intervals, disregarding any delays. For the IMU, a worst-case sampling rate has been used. It is assumed and tested that the total processing time t_{proc} is still considerably shorter than Δt.

6.4 System Model Revisited

The state vector from Chapter 5 (Eq. (4.1)) is somewhat altered when the integrated solution using inertial and camera measurements is proposed. The vector is augmented to include inertial sensor biases, while the angular velocities are included in the input vector $\mathbf{u}$ rather than in the state vector in this integrated positioning system (see Sec. 4.3). The result is estimation of the states of $\mathbf{x_2}$ presented in Equation (4.2). This corresponds to an integration scheme using direct variables as proposed in [16].

6.4.1 Bias Modelling

To estimate and compensate for the sensor biases present in the IMU measurements, a model is required. In [16], the *first order Gauss-Markov process* is proposed as a suitable model for slowly varying processes, while the *Wiener process* is used to model near-constant processes. The first order Gauss-Markov process is described by letting white noise travel through a low-pass filter [16]:

$$\dot{\mathrm{b}}(t) = -\frac{1}{T}\mathrm{b}(t) + \mathrm{w}(t),$$

where $\mathrm{w}(t)$ is zero-mean white noise and T is the Markov time constant. Letting T become large results in the Wiener process:

$$\dot{\mathrm{b}}(t) = \mathrm{w}(t)$$

In this thesis, the first order Gauss-Markov process is used to model the sensor biases. However, experiments with various values of the time constant T have been performed, and large values seem to lead to satisfactory results. The biases are expressed as follows:

$$\begin{bmatrix} \dot{\mathbf{b}}_{\text{acc}} \\ \dot{\mathbf{b}}_{\text{gyro}} \end{bmatrix} = -\mathbf{T}^{-1} \begin{bmatrix} \mathbf{b}_{\text{acc}} \\ \mathbf{b}_{\text{gyro}} \end{bmatrix} + \mathbf{w}_{\text{G-M}}, \tag{6.10}$$

where $\mathbf{T}$ is a vector of time constants while $\mathbf{w}_{\text{G-M}}$ is a vector of zero-mean white noise.

6.4.2 Continuous System Model

Disregarding the procedure in which accelerations are obtained, i.e. measurements of specific force influenced by bias and noise, the velocity dynamics decomposed in the BODY frame are simply

$$\dot{\mathbf{v}}^{\mathrm{b}}_{\mathrm{b/i}} = \mathbf{a}^{\mathrm{b}}_{\mathrm{b/i}} \tag{6.11}$$

The altered state vector,

$$\mathbf{x_2} = \left[(\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}})^{\intercal}, (\mathbf{\Theta}_{\mathrm{ib}})^{\intercal}, (\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}})^{\intercal}, (\mathbf{b}^{\mathrm{b}}_{\mathrm{acc}})^{\intercal}, (\mathbf{b}^{\mathrm{b}}_{\mathrm{gyro}})^{\intercal} \right]^{\intercal}, \tag{6.12}$$

in combination with Equations (5.5), (6.10), and (6.11) results in the following augmented continuous system model:

$$\dot{\mathbf{x}}_\mathbf{2} = \mathbf{f}_{\mathbf{2},\mathrm{cont}}(\mathbf{x_2}, \mathbf{a}^{\mathrm{b}}_{\mathrm{b/i}}, \boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}}) + \mathbf{w}_{\mathbf{2},\mathrm{cont}} = \tag{6.13}$$

$$\begin{bmatrix} c\psi c\theta \cdot v_{\mathrm{x}} + (c\psi s\theta s\phi - s\psi c\phi) \cdot v_{\mathrm{y}} + (s\psi s\phi + c\psi c\phi s\theta) \cdot v_{\mathrm{z}} \\ s\psi c\theta \cdot v_{\mathrm{x}} + (c\psi c\phi + s\phi s\theta s\psi) \cdot v_{\mathrm{y}} + (s\theta s\psi c\phi - c\psi s\phi) \cdot v_{\mathrm{z}} \\ -s\theta \cdot v_{\mathrm{x}} + c\theta s\phi \cdot v_{\mathrm{y}} + c\theta c\phi \cdot v_{\mathrm{z}} \\ \omega_{\mathrm{x}} + s\phi t\theta \cdot \omega_{\mathrm{y}} + c\phi t\theta \cdot \omega_{\mathrm{z}} \\ c\phi \cdot \omega_{\mathrm{y}} - s\phi \cdot \omega_{\mathrm{z}} \\ \frac{s\phi}{c\theta} \cdot \omega_{\mathrm{y}} + \frac{c\phi}{c\theta} \cdot \omega_{\mathrm{z}} \\ a_{\mathrm{x}} \\ a_{\mathrm{y}} \\ a_{\mathrm{z}} \\ -\frac{1}{T_1} \cdot b_{\mathrm{acc,x}} \\ -\frac{1}{T_2} \cdot b_{\mathrm{acc,y}} \\ -\frac{1}{T_3} \cdot b_{\mathrm{acc,z}} \\ -\frac{1}{T_4} \cdot b_{\mathrm{gyro,x}} \\ -\frac{1}{T_5} \cdot b_{\mathrm{gyro,y}} \\ -\frac{1}{T_6} \cdot b_{\mathrm{gyro,z}} \end{bmatrix} + \mathbf{w}_{\mathbf{2},\mathrm{cont}},$$

where $\mathbf{w}_{\mathbf{2},\mathrm{cont}}$ is the continuous system noise including the zero-mean white noise $\mathbf{w}_{\text{G-M}}$ of the Gauss-Markov processes. Equation (6.13) is invalid for $\theta = \pm 90° = \pm\frac{\pi}{2}\mathrm{rad}$ ($\cos\theta = 0$).

6.5 Integrated Solution - EKF Setup

The extended Kalman filter used as an integration filter exploits the sensor fusion properties of the state estimator. In addition to performing the state estimation, the filter can be tuned to achieve the best possible weighting of the available measurement sources. In theory, an extra measurement will always improve the resulting state estimates [16]. The additional measurements in the Integrated Camera System/INS Algorithm (with respect to the first version of the algorithm) are modelled as *inputs* $\mathbf{u}_{\text{IMU}}$ to the filter rather than augmenting the measurement vector $\mathbf{z}$ [6] (see Sec. 4.3). Hence, $\mathbf{z}$ remains unchanged, while $\mathbf{u}_{\text{IMU}}$ is given by Equations (4.9) and (4.10) (repeated here for clarity):

$$\mathbf{u} = \begin{bmatrix} a_{\text{meas,x}} + b_{\text{acc,x}} \\ a_{\text{meas,y}} + b_{\text{acc,y}} \\ a_{\text{meas,z}} + b_{\text{acc,z}} \\ \omega_{\text{x}} + b_{\text{gyro,x}} \\ \omega_{\text{y}} + b_{\text{gyro,y}} \\ \omega_{\text{z}} + b_{\text{gyro,z}} \end{bmatrix} = \begin{bmatrix} u_{\text{acc,x}} \\ u_{\text{acc,y}} \\ u_{\text{acc,z}} \\ u_{\text{gyro,x}} \\ u_{\text{gyro,y}} \\ u_{\text{gyro,z}} \end{bmatrix}, \mathbf{u}_{\text{IMU}} = \mathbf{u} + \mathbf{w}_{\text{IMU}} = \begin{bmatrix} \mathbf{a}_{\text{IMU}} \\ \boldsymbol{\omega}_{\text{IMU}} \end{bmatrix}$$

The IMU output models (Eqs. (6.5) and (6.7)) are used to account for biases and noise in the inertial sensor readings.

The updated system will now be expressed in the form given by Equations (5.6) and (5.7) for use in the second version of the extended Kalman filter:

$$\begin{aligned} \mathbf{x_2}(k+1) &= \mathbf{x_2}(k) + \Delta t \cdot [\mathbf{f}_{\mathbf{2},\text{cont}}[\mathbf{x_2}(k), \mathbf{u}(k)] + \mathbf{w}_{\mathbf{2},\text{cont}}(k)] \\ &= \mathbf{f_2}[\mathbf{x_2}(k), \mathbf{u}(k)] + \mathbf{w_2}(k), \end{aligned} \tag{6.14}$$

where $\mathbf{w_2}(k)$ is the process noise which also accounts for the noise included in the measurements from the inertial sensors, $\mathbf{w}_{\text{IMU}}$.

$$\begin{aligned} \mathbf{z}(k) &= \left[x(k), y(k), z(k), \phi(k), \theta(k), \psi(k)\right]^{\mathsf{T}} + \mathbf{v_2}(k) \\ &= \mathbf{h}[\mathbf{x_2}(k)] + \mathbf{v_2}(k), \end{aligned} \tag{6.15}$$

where $\mathbf{v_2}(k)$ is the measurement noise included in the information from Tracking Tools.

The EKF equations given by Eqs. (5.10)-(5.19) are used in the second positioning algorithm as well, for updated system model as well as state, measurement, and input vectors.

6.6 Implementation

The Integrated Camera System/INS Algorithm is implemented for both simulated IMU measurements as well as offline and online use of real measurements. The version using simulated IMU measurements is implemented to be able to accurately test the state estimation against known ground truths, while the online version of the algorithm is of practical use for the proposed lab setup. Furthermore, the offline version in combination with its Camera Measurement Algorithm equivalent may be used for comparison of the two. This section aims to present relevant details of the implementation which have not already been discussed. As a consequence, neither the OptiTrack measurement preprocessing nor the offline processing of OptiTrack measurements (presented in Section 5.7) are repeated here. The choice of not including an equivalent to the Camera Measurement Algorithm pseudocode (Algorithm 1) has been made because the similarities as well as differences between the two have already been presented. Thus, it is assumed that little new information would be revealed by doing so. Furthermore, all code produced in this thesis is available upon request.

6.6.1 Simulated IMU Measurements and Offline Processing

The version of the Integrated Camera System/INS Algorithm using *simulated* IMU measurements is obtained by first acquiring camera system measurements from the lab, and subsequently processing the data offline using the Camera Measurement Algorithm. Furthermore, the angular velocities as well as linear accelerations are extracted from the processed camera system measurements, and sensor biases are added. Thus, IMU measurements are simulated. The recorded OptiTrack and simulated IMU measurements are then processed by the Integrated Camera System/INS Algorithm, and the resulting state estimates can be compared to a known ground truth.

To process *real* IMU data offline, measurements are recorded by storing data as .txt files. These files may then be read into MATLAB through use of the built-in function `textscan`. To perform offline processing using the Integrated Camera System/INS Algorithm, the recorded measurements from the two sources need be synchronized. As explained earlier, the two sensor systems are accessed by separate computers (the remote Linux computer receives inertial sensor measurements while the local PC communicates with OptiTrack). As a consequence, the synchronization is achieved by manually starting the programs on both computers simultaneously. Furthermore, a visual correction is performed when needed. That is, if the recorded measurements are displaced with respect to each other, the appropriate time steps are deleted from the files.

6.6.2 TCP/IP Communication

The part of the data exchange between the Parrot AR.Drone 2.0 and the computer running the Integrated Camera System/INS Algorithm performed in this thesis (presented in Subsection 6.3.1) is implemented through use of the `Matlab Central` function `jtcp`[1]. The function uses MATLAB's Java interface to perform TCP/IP communication with a remote computer. As mentioned earlier, the UAV to Linux computer data exchange is not part of this thesis. The `jtcp` function is used to establish a TCP/IP connection between the two computers, and subsequently read messages containing the latest IMU navigation data.

[1]Source: www.mathworks.com/matlabcentral/fileexchange/24524-tcpip-communications-in-matlab

6.6.3 Preliminary Aspects: Analytical Calculation of EKF Matrices Revisited

Similarly to the Camera Measurement Algorithm (Subsection 5.7.3), the Jacobians of the system and measurement functions, the $\mathbf{\Phi_2}(k)$ and $\mathbf{H_2}(k) = \mathbf{H_2}$ matrices, are calculated offline. $\mathbf{\Phi_2}(k)$ is displayed in a similar manner as its Camera Measurement Algorithm equivalent. That is, submatrices are used due to lack of space. The arguments k are omitted for simplicity.

$$\mathbf{h}[\mathbf{x_2}(k)] = \left[x(k), y(k), z(k), \phi(k), \theta(k), \psi(k)\right]^\intercal \implies$$

$$\mathbf{H_2} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x_2}}\right|_{\mathbf{x_2}=\bar{\mathbf{x}}_\mathbf{2}(k)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.16}$$

$$\mathbf{f_2}[\mathbf{x_2}(k), \mathbf{u}(k)] =$$

$$\begin{bmatrix} x + \Delta t[c\psi c\theta \cdot v_x + (c\psi s\theta s\phi - s\psi c\phi) \cdot v_y + (s\psi s\phi + c\psi c\phi s\theta) \cdot v_z] \\ y + \Delta t[s\psi c\theta \cdot v_x + (c\psi c\phi + s\phi s\theta s\psi) \cdot v_y + (s\theta s\psi c\phi - c\psi s\phi) \cdot v_z] \\ z + \Delta t[-s\theta \cdot v_x + c\theta s\phi \cdot v_y + c\theta c\phi \cdot v_z] \\ \phi + \Delta t[(u_{\text{gyro,x}} - b_{\text{gyro,x}}) + s\phi t\theta \cdot (u_{\text{gyro,y}} - b_{\text{gyro,y}}) + c\phi t\theta \cdot (u_{\text{gyro,z}} - b_{\text{gyro,z}})] \\ \theta + \Delta t[c\phi \cdot (u_{\text{gyro,y}} - b_{\text{gyro,y}}) - s\phi \cdot (u_{\text{gyro,z}} - b_{\text{gyro,z}})] \\ \psi + \Delta t[\frac{s\phi}{c\theta} \cdot (u_{\text{gyro,y}} - b_{\text{gyro,y}}) + \frac{c\phi}{c\theta} \cdot (u_{\text{gyro,z}} - b_{\text{gyro,z}})] \\ v_x + \Delta t[u_{\text{acc,x}} - b_{\text{acc,x}} - s\theta g] \\ v_y + \Delta t[u_{\text{acc,y}} - b_{\text{acc,y}} + c\theta s\phi g] \\ v_z + \Delta t[u_{\text{acc,z}} - b_{\text{acc,z}} + c\theta c\phi g] \\ b_{\text{acc,x}} - \Delta t[\frac{1}{T_1} b_{\text{acc,x}}] \\ b_{\text{acc,y}} - \Delta t[\frac{1}{T_2} b_{\text{acc,y}}] \\ b_{\text{acc,z}} - \Delta t[\frac{1}{T_3} b_{\text{acc,z}}] \\ b_{\text{gyro,x}} - \Delta t[\frac{1}{T_4} b_{\text{gyro,x}}] \\ b_{\text{gyro,y}} - \Delta t[\frac{1}{T_5} b_{\text{gyro,y}}] \\ b_{\text{gyro,z}} - \Delta t[\frac{1}{T_6} b_{\text{gyro,z}}] \end{bmatrix} \implies \tag{6.17}$$

$$\mathbf{\Phi_2}(k) = \left.\frac{\partial \mathbf{f_2}}{\partial \mathbf{x_2}}\right|_{\mathbf{x_2}=\hat{\mathbf{x}}_\mathbf{2}(k)} = \begin{bmatrix} \mathbf{\Phi_{2}}_{,1} & \mathbf{\Phi_{2}}_{,2} & \mathbf{\Phi_{2}}_{,3} \\ \mathbf{\Phi_{2}}_{,4} & \mathbf{\Phi_{2}}_{,5} & \mathbf{\Phi_{2}}_{,6} \\ \mathbf{\Phi_{2}}_{,7} & \mathbf{\Phi_{2}}_{,8} & \mathbf{\Phi_{2}}_{,9} \end{bmatrix}_{\mathbf{x_2}=\hat{\mathbf{x}}_\mathbf{2}(k),} \tag{6.18}$$

where the elements of $\mathbf{\Phi_2}(k)$ are 5×5 matrices, and can be found in Appendix C.

Chapter 7

Simulations, Testing, and Results

Several aspects of the two implemented positioning algorithms have been tested to investigate the applicability of the algorithms to the proposed lab setup; Preprocessing of the OptiTrack measurements, the "near real-time" demands, state estimation under the best possible conditions, and state estimation when loss of or erroneous measurements are experienced. Additionally, both algorithms were tested using simulated as well as recorded measurements, and the inertial measurements with corresponding bias estimates used in the Integrated Camera System/INS Algorithm were investigated for a stationary scenario. This chapter presents the setups and input parameters as well as the results of the performed tests, and it is organized as follows:

Camera Measurement Algorithm (Sections 7.1 & 7.2)
The first two sections are dedicated to analysis of the Camera Measurement Algorithm. First, tests performed using simulated measurements are presented (Section 7.1), while investigation of its performance in the lab setup is conducted in Section 7.2.

State vector: $\mathbf{x_1} = \left[(\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}})^\intercal, (\mathbf{\Theta}_{\mathrm{ib}})^\intercal, (\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}})^\intercal, (\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}})^\intercal\right]^\intercal$
Available measurements: $\mathbf{h}(\mathbf{x_1}) = \left[x, y, z, \phi, \theta, \psi\right]^\intercal$

Integrated Camera System/INS Algorithm (Sections 7.3 & 7.4)
The third and fourth sections contain tests organized similarly, but for the Integrated Camera System/INS Algorithm. Section 7.3 consists of tests performed using recorded measurements from the camera system combined with simulated IMU measurements, while Section 7.4 presents the results achieved in the lab setup.

State vector: $\mathbf{x_2} = \left[(\mathbf{p}^{\mathrm{i}}_{\mathrm{b/i}})^\intercal, (\mathbf{\Theta}_{\mathrm{ib}})^\intercal, (\mathbf{v}^{\mathrm{b}}_{\mathrm{b/i}})^\intercal, (\mathbf{b}^{\mathrm{b}}_{\mathrm{acc}})^\intercal, (\mathbf{b}^{\mathrm{b}}_{\mathrm{gyro}})^\intercal\right]^\intercal$

Available measurements: $\mathbf{h}(\mathbf{x_2}) = \left[x, y, z, \phi, \theta, \psi\right]^\intercal$,
$\mathbf{u} = \left[u_{\text{acc,x}}, u_{\text{acc,y}}, u_{\text{acc,z}}, u_{\text{gyro,x}}, u_{\text{gyro,y}}, u_{\text{gyro,z}}\right]^\intercal$

Finally, an offline comparison of the two positioning algorithms concludes the chapter in Section 7.5.

7.1 Camera Measurement Algorithm - Simulated Measurements

The Camera Measurement Algorithm using real measurements is implemented for 6 DOF motion in the lab. However, in order to detect possible errors, investigate the results under known circumstances, and accurately test the performance of the *unmeasured* state estimation, the algorithm is first evaluated using simulated measurements. Two specific scenarios were simulated - the UAV flying diagonally with a constant velocity (**Scenario 1**), and hovering in a circular pattern (**Scenario 2**). For both scenarios, an error-free running as well as the impacts of measurement errors are tested. The input parameters common to all simulations are presented in Table 7.1, while process and measurement noise were added to the simulated plants according to $0.01 \cdot \text{std}(\mathbf{Q_1})$ and $0.01 \cdot \text{std}(\mathbf{R_1})$, respectively. This section presents **Scenario 1** (diagonal flying with constant velocity), while the results of **Scenario 2** are given in Appendix D.1.

Table 7.1: *Input Parameters, Simulations of Camera Measurement Algorithm*

$\mathbf{Q_1}$	diag [0.1, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.1, 0.1, 0.1]
$\mathbf{R_1}$	diag [0.001, 0.001, 0.001, 0.1, 0.1, 0.1]
Δt	0.1 s
$\bar{\mathbf{P}}_\mathbf{1}(0)$	diag [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
$\mathbf{Q}_{\mathbf{1},\text{measError}}$	diag [0.01, 0.01, 0.01, 0.001, 0.001, 0.001, 0.01, 0.01, 0.06, 0.02, 0.02, 0.02]
$\mathbf{R}_{\mathbf{1},\text{measError}}$	diag [100000, 100000, 100000, 100000, 100000, 100000]

7.1.1 Performance of State Estimation, Simulation Scenario 1

The initial conditions for the state vector used in the plant of **Scenario 1** are given by

$$\mathbf{x_1}(0) = \left[0, 0, 0.5, 0, 0, 0, 1, 0.4, 0, 0, 0, 0\right]^\intercal, \tag{7.1}$$

while the initial $\bar{\mathbf{x}}_\mathbf{1}$ was

$$\bar{\mathbf{x}}_\mathbf{1}(0) = \left[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right]^\intercal \tag{7.2}$$

Due to the v_x and v_y values (elements 7 and 8 in the state vector) being constant in the plant model and initially given by Eq. (7.1), these velocities will remain approximately constant throughout the simulation. As a consequence, the UAV would have followed a straight, diagonal path in the noise-free case. In this first test, a normal running of the Camera Measurement Algorithm (without any special cases) was simulated.

The results of the test using the initial conditions given above as well as the input parameters of Table 7.1 are shown in Figures 7.1 and 7.2 for the measured and unmeasured states, respectively. The oscillations in the measurements and corresponding estimates are a result of the process and measurement noise added to the simulation.

The values of the plant (and estimated) positions (x, y, z [m]) are larger than the actual indoor lab setup due to the main goal being the testing of (unmeasured) velocity estimation under realistic conditions. That is, with the plant being simulated using the same model as in the filter ($\dot{\mathbf{p}}^{\mathrm{i}} = \mathbf{R}^{\mathrm{i}}_{\mathrm{b}}\mathbf{v}^{\mathrm{b}}$, $\dot{\mathbf{v}}^{\mathrm{b}} = 0$), the x and y positions will increase linearly when the initial velocities are constant and non-zero.

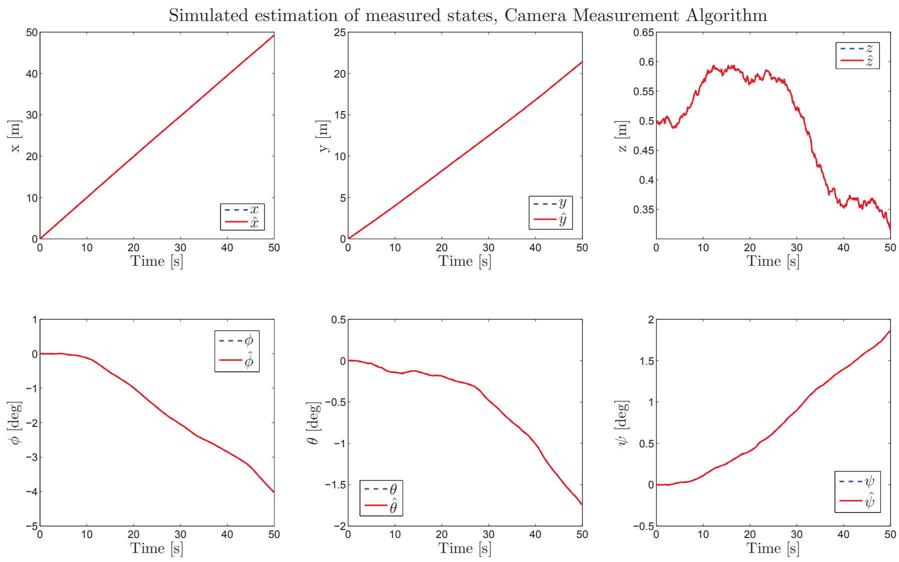

Figure 7.1: *Measured state estimation, Simulation Scenario 1, Camera Measurement Algorithm. The UAV is flying in a diagonal manner with constant v_x and v_y velocities. The real values from the plant are shown as blue, dashed lines while the estimates are represented by red lines. The top-most row depicts the position, while the lower row shows the orientation.*

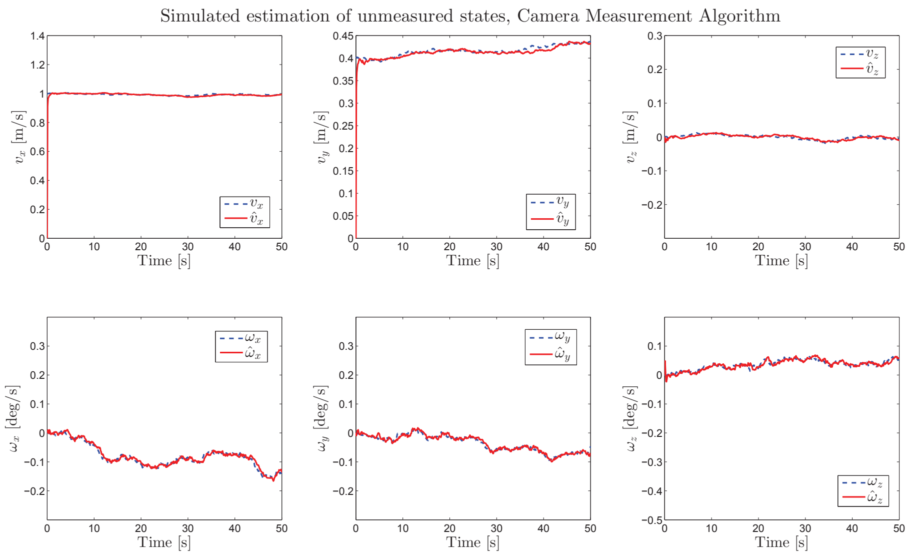

Figure 7.2: *Unmeasured state estimation, Simulation Scenario 1, Camera Measurement Algorithm. The UAV is flying in a diagonal manner with constant v_x and v_y velocities. The real values from the plant are shown as blue, dashed lines while the estimates are represented by red lines. The linear velocities are shown in the top-most row, while the angular velocities are depicted in the second row.*

7.1.2 Impacts of Measurement Loss, Simulation Scenario 1

The impacts of measurement loss on the state estimation are tested using simulated measurements for the same scenario as in the previous subsection (**Scenario 1**). The same input parameters were used (Table 7.1 and Eqs. (7.1) and (7.2)), and this test required the $\mathbf{Q}_{\mathbf{1},\text{measError}}$ and $\mathbf{R}_{\mathbf{1},\text{measError}}$ matrices to be put to use. These tuning matrices are used during measurement outages to ensure more appropriate weighting of modelled vs. measured behavior.

For four seconds at $t = 20 - 24$ s and two seconds at $t = 41 - 43$ s, the simulated camera measurements were lost. The actions taken by the algorithm when receiving incorrect measurements are equal to the response to measurement loss, i.e. the previously received (correct) measurement is used as input to the state estimator (along with appropriate $\mathbf{Q}_{\mathbf{1},\text{measError}}$ and $\mathbf{R}_{\mathbf{1},\text{measError}}$) until a new, correct $\mathbf{z}$ is received (see Section 5.4). Hence, the impacts of both types of measurement errors on the state estimates were tested.

In Figures 7.3 and 7.4, the *estimation errors* are depicted, i.e. the difference between real (plant) and estimated values. The impact of measurement loss is visible in Figure 7.3 as the regions in which these estimation errors increase.

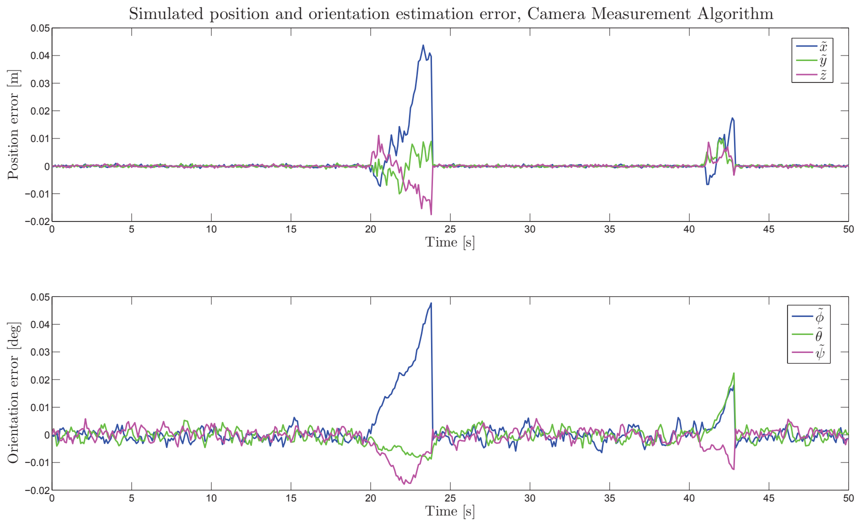

Figure 7.3: *Impact of measurement loss on position and orientation estimation, Simulation Scenario 1, Camera Measurement Algorithm. The UAV is flying in a diagonal manner with constant v_x and v_y velocities, and the measurements are lost twice. The top-most row shows the position error, while the lower row depicts orientation error.*

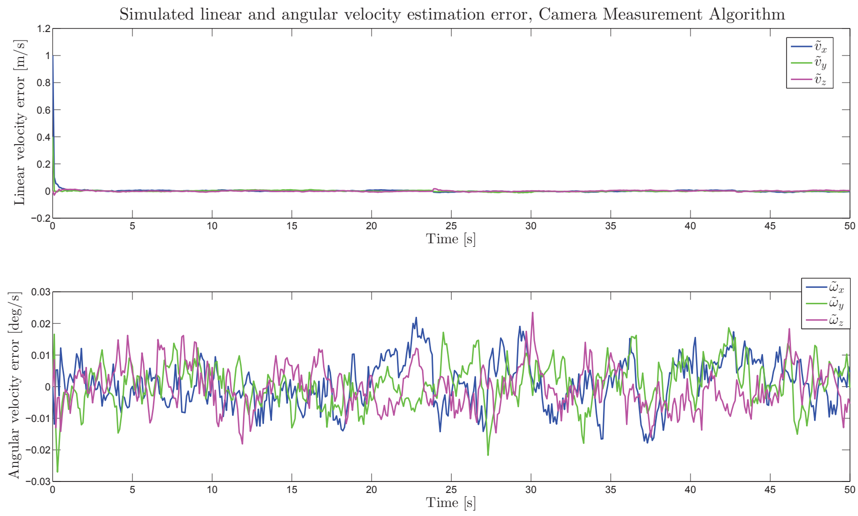

Figure 7.4: *Impact of measurement loss on linear and angular velocity estimation, Simulation Scenario 1, Camera Measurement Algorithm. The UAV is flying in a diagonal manner with constant v_x and v_y velocities, and the measurements are lost twice. The top-most row shows the error in linear velocities, while the lower row depicts angular velocity error.*

The measured state estimation errors reach peaks with acceptable values for both outages, while the unmeasured state estimates seem unaffected by the loss of measurements. That is, by close inspection of Figure 7.4, it can be seen that the estimates are affected to some degree, although not significantly.

7.2 Camera Measurement Algorithm - Online Implementation

This section presents the results of running the Camera Measurement Algorithm in the lab setup using the OptiTrack system. The input parameters of Table 7.2 were used in all tests. As can be seen in the table, the design matrices have been tuned compared to the values used during the simulations presented above (Table 7.1). It was discovered through trial and error that running the algorithm with the real system required the relative weighting presented in Table 7.2 to perform in the best possible manner.

Table 7.2: *Input Parameters, Online Tests of Camera Measurement Algorithm*

$\mathbf{Q_1}$	diag [0.1, 0.1, 0.1, 0.01, 0.01, 0.01, 0.5, 0.5, 0.5, 0.3, 0.3, 0.3]
$\mathbf{R_1}$	diag [0.001, 0.001, 0.001, 0.1, 0.1, 0.1]
Δt	0.01 s
$\mathbf{\bar{P}_1}(0)$	diag [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
$\mathbf{\bar{x}_1}(0)^\intercal$	$\begin{bmatrix}0,0,0,0,0,0,0,0,0,0,0,0\end{bmatrix}$
$\mathbf{Q}_{\mathbf{1},\text{measError}}$	diag [0.01, 0.01, 0.01, 0.001, 0.001, 0.001, 0.0005, 0.0005, 0.0005, 0.001, 0.001, 0.001]
$\mathbf{R}_{\mathbf{1},\text{measError}}$	diag [1000, 1000, 1000, 1000, 1000, 1000]
noMeasLimit	500 time steps

In this section, the estimated linear velocities have been rotated to the inertial frame before being shown in figures. This is performed to enable direct comparison with the corresponding positions for each test due to there being no measurements to compare the velocity estimates to. The original estimates, which are expressed in the BODY frame, are found in Appendix D.2.

The start-up procedures of the camera system are visible in the figures depicting real-life scenarios as the period (from $t = 0$ to approximately $t \simeq 5$ seconds) in which the measurements approach their initial values. The unmeasured state estimation begins when this initialization procedure is over.

7.2.1 Time Test

In the presentation of the Camera Measurement Algorithm in Chapter 5, the "near real-time" demands of the positioning algorithm were discussed. It was stated that in order to avoid loss of data from OptiTrack while at the same time not performing state estimation more than once for each measurement input, a new measurement should be collected from OptiTrack exactly every Δt seconds, where Δt is the sampling period of the system. For each measurement update, processing demanding the total time of t_{proc} need be performed. Hence, $t_{\text{proc}} < \Delta t$ is required. These "near real-time" aspects are investigated in this test.

The `while` loop of the Camera Measurement Algorithm was run for 4000 time steps, and for each step, the processing time as well as the `sleep` interval were logged. The time spent during the call to the `sleep` function was logged in order to investigate the accuracy of this function, which influences when a new measurement input is collected (see Algorithm 1). The difference between the desired `sleep` interval (`sleep`$(\Delta t - t_{\text{proc}})$) and the actual time spent pausing was investigated.

Table 7.3: *Results of Time Test, Camera Measurement Algorithm*

`max` (t_{proc})	0.0221 s
`mean` (t_{proc})	0.0016 s
`max` $(pause\_error)$	0.0310 s
`mean` $(pause\_error)$	0.0013 s

The maximum values of both t_{proc} and *pause_error* were detected during the system start-up procedures, and they are the only unacceptably large values for both variables. That is, all other t_{proc} values are significantly smaller than the sampling interval $\Delta t = 0.01$ s, fulfilling the $t_{\text{proc}} < \Delta t$ demand.

As Table 7.3 shows, the pause interval is somewhat inaccurate with a mean error value of 0.0013 s. The error was even larger when using built-in Matlab pause functions, thus Java's `sleep` function was utilized. As mentioned above, this function decides how long the system is delayed waiting for a new measurement to be available, and is ideally given by $\Delta t - t_{\text{proc}}$. However, the online tests of the performance of the algorithm show satisfying results despite the inaccuracy in the pause function.

7.2.2 State Estimation under Ideal Conditions, Test 1

The performance of the Camera Measurement Algorithm was tested in a realistic setting, i.e. the position, orientation, and velocities were estimated for the UAV while it was flying around the lab setup. During the tests, the cameras were collecting frames and sending measurements to Matlab. This subsection presents the results of **Test 1**, while another scenario (**Test 2**) is presented in Appendix D.2.2. No incorrect or loss of measurements were experienced during these two tests, i.e. the main functionality is tested.

The results of **Test 1** are presented in Figures 7.5 and 7.6 for the measured and unmeasured states, respectively. Some of the larger spikes in the blue, dashed lines (which represent the measurements from the camera system) in Figure 7.5 may be measurement outliers, which are handled by the faulty measurement mechanism described in Section 5.4. However, in the ϕ and θ angles especially, some of the spikes are due to these angles receiving control signals to induce motion [12]. That is, the non-planar angles are controlled to obtain motion in the x and y directions, resulting in angular motion as well.

Figure 7.5 shows that the estimates of the measured states follow the measurements quite closely, while filtering out what was assumed to be errors in the measurements from the OptiTrack system. Measurement noise is visible particularly in the orientation states, which is in compliance with the measurement noise observed during the testing of the OptiTrack to MATLAB communication in Subsection 3.4.2 (Table 3.3 and Figure 3.6). It can be seen that the state estimator performs noise filtering particularly in these states.

The measured yaw angle ψ and its estimate cross the 180° limit in Figure 7.5, and the result of the measurement preprocessing is visible. No leaps occur because the angle continues to increase rather than jump from 180° to −180°.

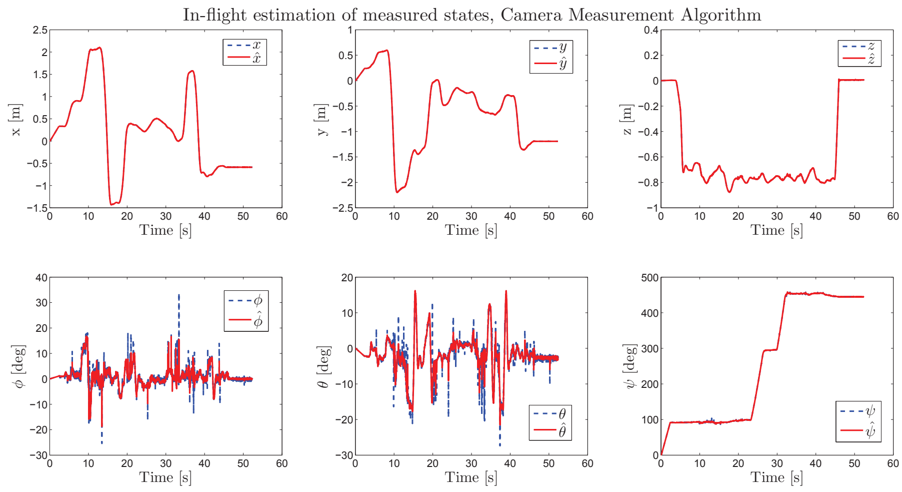

Figure 7.5: *In-flight measured state estimation, Test 1, Camera Measurement Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Camera Measurement Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

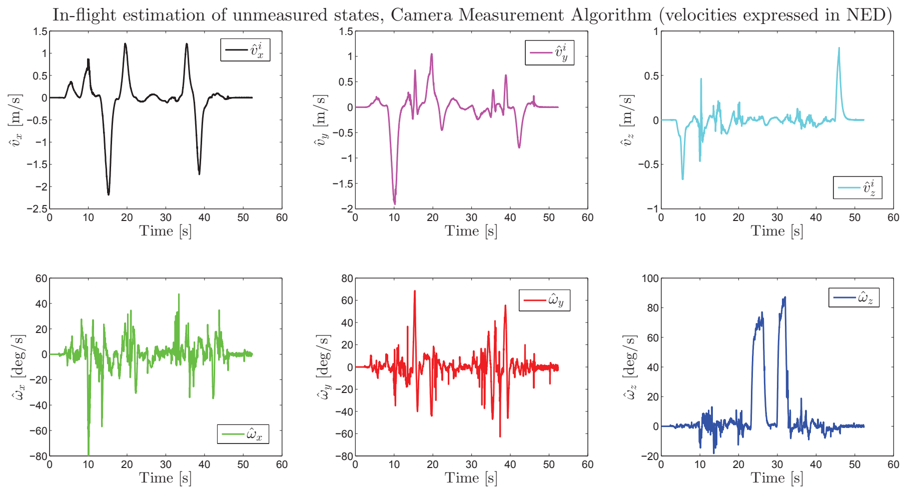

Figure 7.6: *In-flight unmeasured state estimation, Test 1, Camera Measurement Algorithm. The linear velocity estimates (expressed in the inertial frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

Figure 7.6 depicts the results of estimating the unmeasured states, which naturally have no measurements to be compared to. As mentioned in the beginning of this section, the velocity estimates are rotated to the inertial frame to enable direct comparison with the corresponding positions in Figure 7.5, while the original velocity estimates (expressed in BODY) can be found in Appendix D.2.1 for this test.

7.2.3 State Estimation with Measurement Errors

This test was performed under similar conditions as the scenario in the previous subsection. However, the UAV was now moved manually around the lab setup in order to provoke OptiTrack measurement loss. As mentioned earlier, receiving incorrect measurements has the same consequences as if the measurements were lost. That is, the impacts of both types of measurement errors are investigated in this test.

Figure 7.7 depicts the measured state estimation for a scenario in which the trackable connected to the UAV was missed by the cameras several times. As can be seen in the figure, the state estimation was affected by the measurement losses, which occurred between $t \simeq 14.5$ s and $t \simeq 16$ s, and for approximately two seconds at $t \simeq 27$ s. OptiTrack outputs the last obtained measurement until the trackable is within the capture volume once more, that is, the measurements become constant. The states are estimated by emphasizing the modelled behavior, and do not become constant.

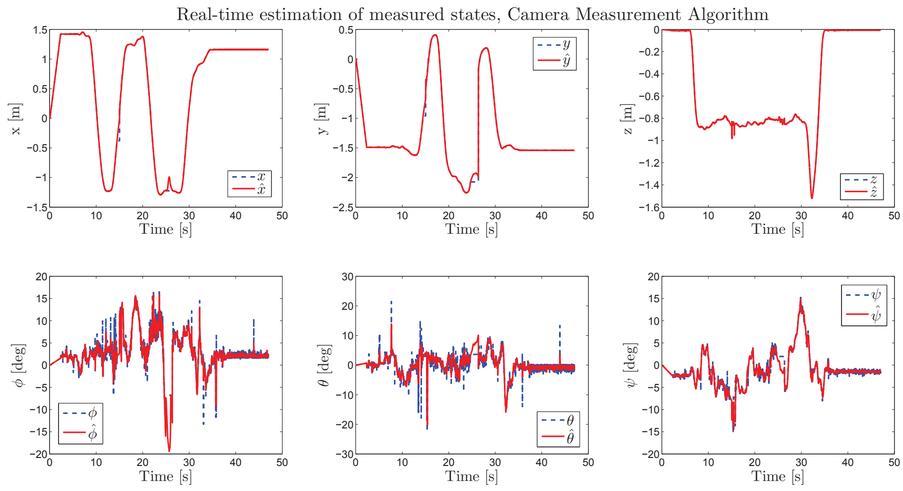

Figure 7.7: *Online measured state estimation with measurement loss, Camera Measurement Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Camera Measurement Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

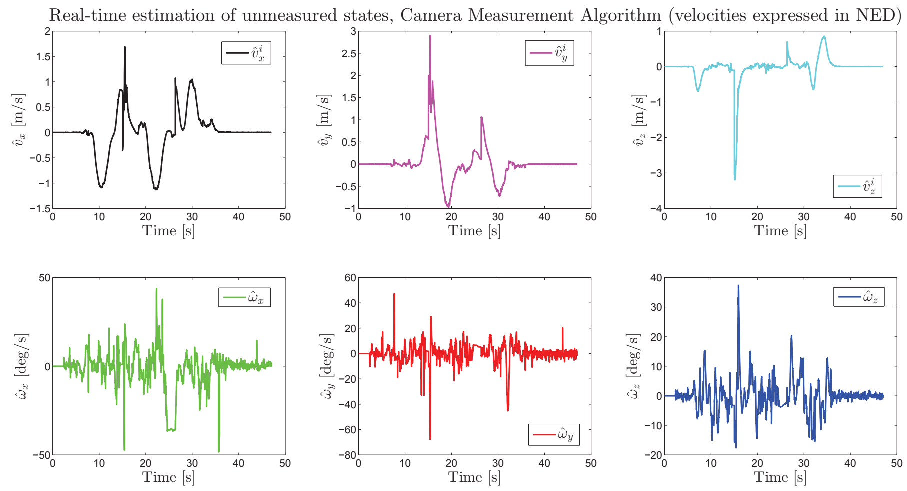

Figure 7.8: *Online unmeasured state estimation with measurement loss, Camera Measurement Algorithm. The linear velocity estimates (expressed in the inertial frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

The validity of the state estimates during measurement outages is difficult to test accurately. Incorrect or inaccurate state estimates should not be sent to the control system, which is the reason for setting a time limit for allowing state estimation without inputs before the algorithm is aborted.

In Figure 7.8, the linear velocity estimates expressed in the inertial frame are depicted, while the original estimates (relative to BODY) can be found in Appendix D.2.3. From Figure 7.8, it is seen that the unmeasured state estimation was also affected by the loss of measurements. The impacts of measurement loss are more difficult to see due to there being no ground truth in the real system, but in the results obtained for simulated measurements (Subsec. 7.1.2), the impacts were visible in the measured state estimates, while the velocity estimates were less affected.

This concludes the results obtained for the Camera Measurement Algorithm. The remaining sections of this chapter are dedicated to testing the performance of the Integrated Camera System/INS Algorithm, and to a comparison of the two using recorded measurements of identical scenarios.

7.3 Integrated Camera System/INS Algorithm - Simulated IMU Measurements

The Integrated Camera System/INS Algorithm is, like the Camera Measurement Algorithm, implemented for use with real measurements in the lab. However, results obtained by the second positioning algorithm using recorded camera and simulated IMU measurements are presented in this section for similar reasons as in the Camera Measurement Algorithm equivalent in Section 7.1. As velocity ground truths, trajectories estimated using the first positioning algorithm were used (see Subsection 6.6.1 for details on the IMU measurement simulation justifying this), while the biases were set to be slow-varying processes similar to the real biases observed in stationary inertial measurements from the Parrot AR.Drone 2.0. The input parameters of Table 7.4 were used in all simulations.

Table 7.4: *Input Parameters, Simulations of Integrated Camera System/INS Algorithm*

$\mathbf{Q_2}$	diag [1, 1, 1, 0.01, 0.01, 0.01, 0.2, 0.2, 0.2, 0.1, 0.1, 0.1, 0.001, 0.001, 0.001]
$\mathbf{R_2}$	diag [0.01, 0.01, 0.01, 0.1, 0.1, 0.1]
Δt	0.01 s
$\bar{\mathbf{P}}_\mathbf{2}(0)$	diag [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
$\bar{\mathbf{x}}_\mathbf{2}(0)^\mathsf{T}$	$\begin{bmatrix}0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\end{bmatrix}$
$\mathbf{T}$	$\begin{bmatrix}1000, 1000, 1000, 100, 100, 100\end{bmatrix}$
$\mathbf{Q}_{\mathbf{2},\text{measError}}$	diag [0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.06, 0.06, 0.06, 0.001, 0.001, 0.001, 0.0001, 0.0001, 0.0001]
$\mathbf{R}_{\mathbf{2},\text{measError}}$	diag [100000, 100000, 100000, 100000, 100000, 100000]

7.3.1 Performance of State Estimation

In the test performed in this subsection, a successful running of the Integrated Camera System/INS Algorithm is investigated (analogously to the first tests of the Camera Measurement Algorithm). That is, no measurement outages were experienced. Consequently, the state estimation results under ideal conditions are tested. The results obtained using the input parameters of Table 7.4 are shown in Figures 7.9, 7.10, and 7.11 for the measured states, velocities, and biases, respectively.

It can be seen in Figure 7.9 that the position and orientation estimates follow the measurements quite closely, while filtering out what was assumed to be measurement noise from the camera system. Furthermore, the velocity estimates in Figure 7.10 also follow the real values obtained through processing of recorded camera system measurements. As mentioned above, these processed measurements were used as ground truths in this simulation although it cannot be said with absolute certainty that these velocities are the ground truth. However, this test shows that the Integrated Camera System/INS Algorithm is capable of estimating velocities that follow the corresponding real values, while a verification of the authenticity of the utilized real values is outside the scope of this test.

Finally, Figure 7.11 depicts the bias estimates against the set ground truth. It can be seen that the estimates converge quite quickly, although some fluctuations are present in the estimates, possibly reflecting motion in the corresponding orientation states (by comparison with Figure 7.9).

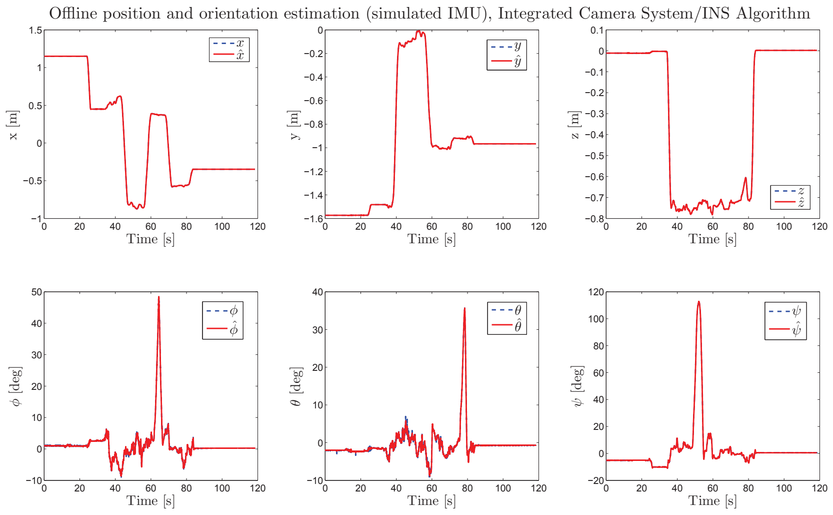

Figure 7.9: *Position and orientation estimation (simulated IMU), Integrated Camera System/INS Algorithm. Recorded measurements from the camera system are used. The real values are shown as blue, dashed lines while the estimates are represented by red lines. The top-most row depicts the position, while the lower row shows the orientation.*

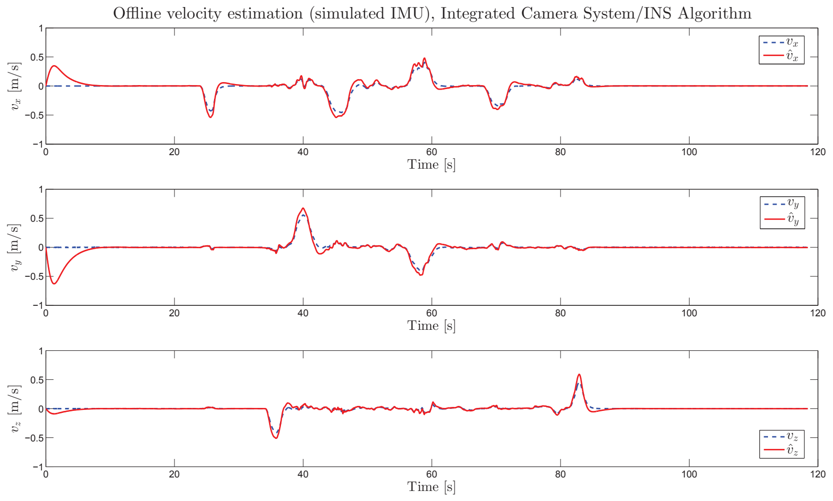

Figure 7.10: *Velocity estimation (simulated IMU), Integrated Camera System/INS Algorithm. The velocities are expressed in the BODY frame. Recorded measurements from the camera system are used. The real values are shown as blue, dashed lines while the estimates are represented by red lines.*

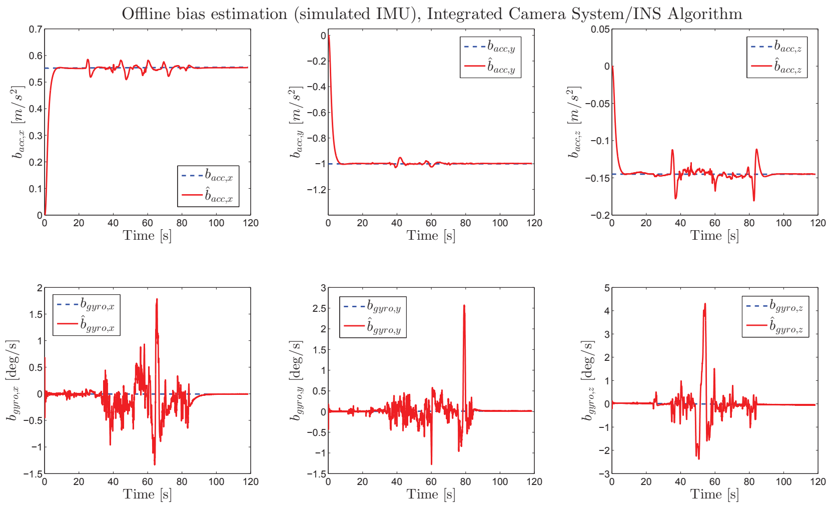

Figure 7.11: *Bias estimation (simulated IMU), Integrated Camera System/INS Algorithm. Recorded measurements from the camera system are used. The real values are shown as blue, dashed lines while the estimates are represented by red lines. The top-most row depicts the accelerometer biases, while the lower row shows the gyro biases.*

7.3.2 Impacts of Measurement Loss

The Integrated Camera System/INS Algorithm uses two separate measurement sources and therefore, two separate measurements may be lost or faulty. In this subsection, the results of losing or receiving faulty measurements from both sources are investigated for the same scenario as in the previous subsection. The figures in this section depict *error signals*, i.e. the difference between actual and estimated values.

Camera System Measurement Loss The effects of camera measurement outages on the state estimates produced by the Integrated Camera System/INS Algorithm are presented in this paragraph. The measurements from OptiTrack were lost twice: for $t = 40 - 44$ s, and $t = 75 - 77$ s, coinciding with the regions in Figure 7.12 where the estimation errors increase. The measured state estimation errors increase and reach quite high values during the two outages, the first one in particular for the positions. Furthermore, it can be seen that the velocity estimates are also affected by the camera measurement losses (Figure 7.13).

The bias estimation errors are depicted in Figure 7.14, and it can be seen that especially the gyroscope bias estimates are more affected by UAV motion than by the camera measurement losses (by comparison with Figure 7.9, which depicts the motion experienced for the error-free version of the same scenario).

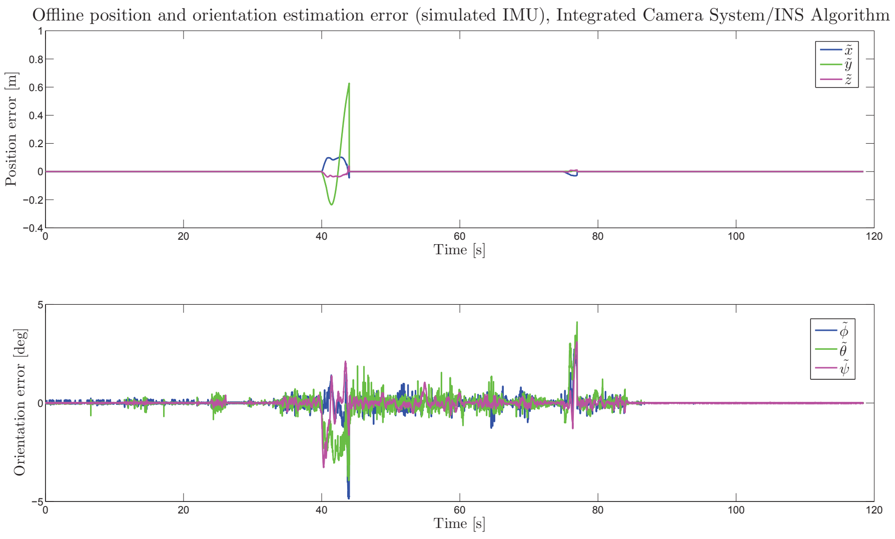

Figure 7.12: *Position and orientation estimation error with camera system outages (simulated IMU), Integrated Camera System/INS Algorithm. The top-most row contains position estimation errors, while the second row depicts orientation estimation errors.*

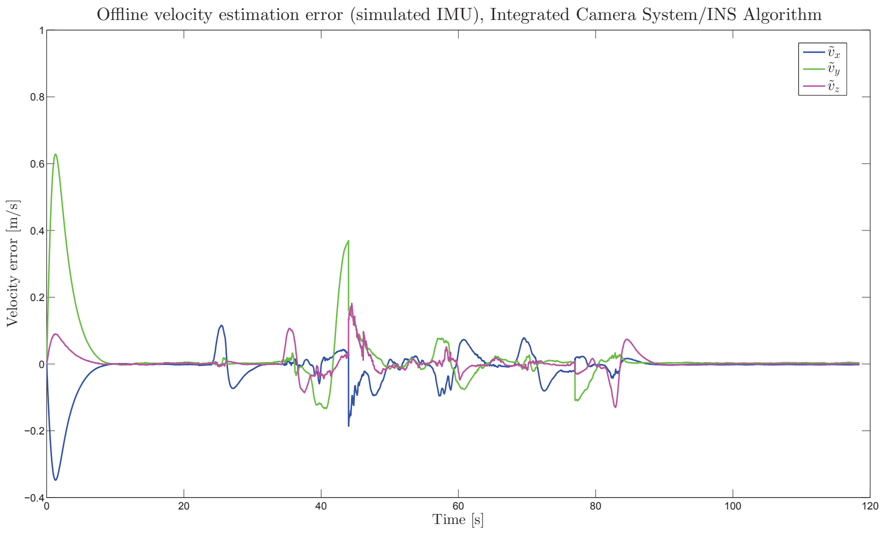

Figure 7.13: *Velocity estimation error with camera system outages (simulated IMU), Integrated Camera System/INS Algorithm.*

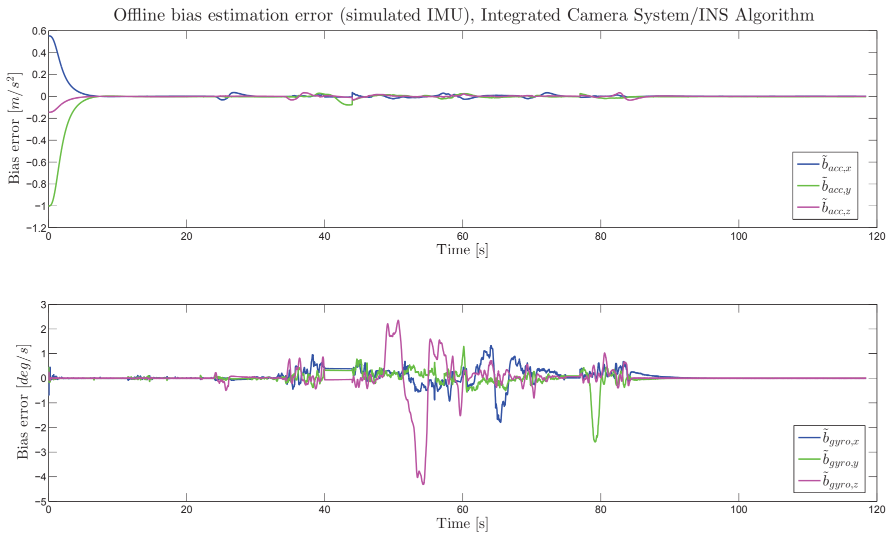

Figure 7.14: *Bias estimation error with camera system outages (simulated IMU), Integrated Camera System/INS Algorithm. The top-most row contains accelerometer bias estimation errors, while the second row depicts gyro bias estimation errors.*

The gyro bias estimation errors seem to become approximately constant for the duration of the camera measurement outages, while the accelerometer bias estimates seem to be affected by the first outage in particular.

Inertial Sensor Measurement Loss The effects of inertial sensor outages on the state estimates are presented in this paragraph. The measurements from the inertial sensors were also lost twice, for the time intervals in which camera measurement outages were experienced for the same scenario above.

Figure 7.15 depicts the errors in the measured state estimation, i.e. position and orientation estimation errors. The figure shows that the position estimates are virtually unaffected by the inertial measurement losses, with the error resulting from the first outage having a magnitude of 10^{-4}, and the error resulting from the second measurement loss being even smaller. The orientation estimation is not significantly influenced either.

The velocity estimation error is depicted in Figure 7.16, and it can be seen that the estimates are affected by the first inertial sensor outage especially.

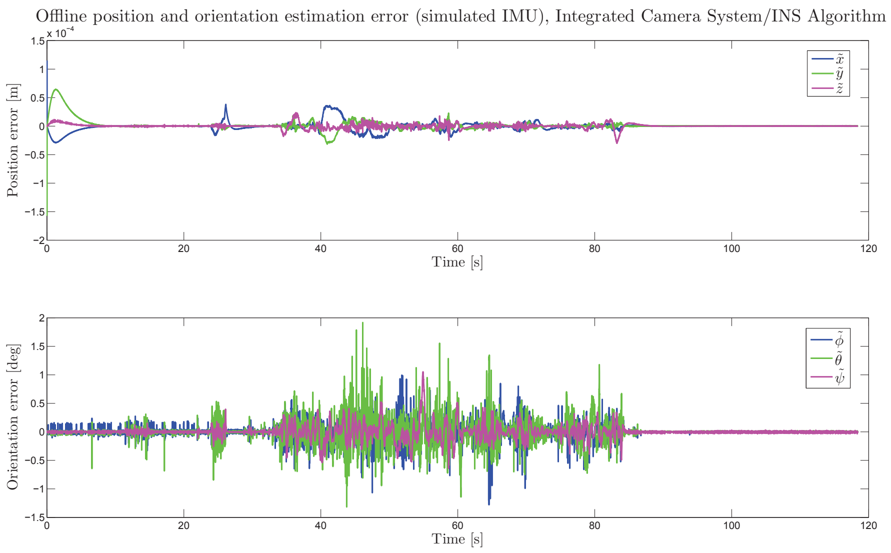

Figure 7.15: *Position and orientation estimation error with inertial sensor measurement loss (simulated IMU), Integrated Camera System/INS Algorithm. The top-most row contains position estimation errors, while the second row depicts orientation estimation errors.*

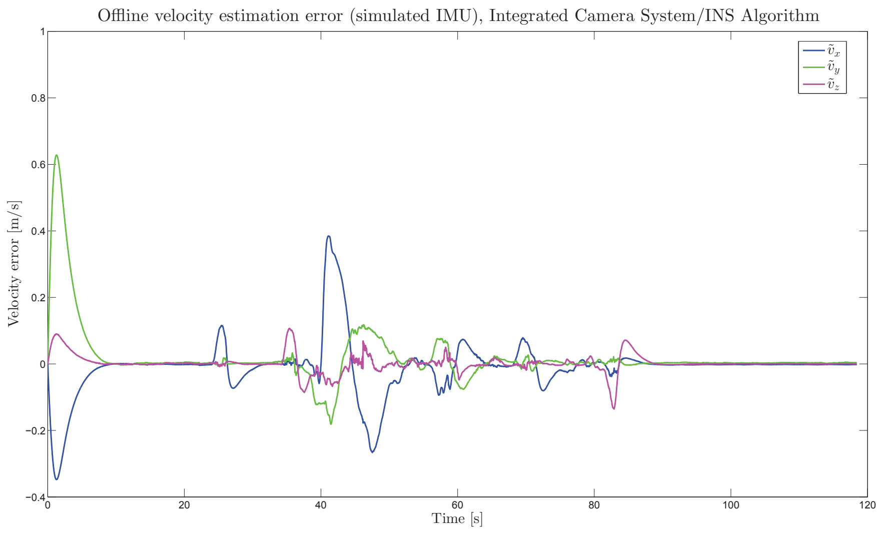

Figure 7.16: *Velocity estimation error with inertial sensor measurement loss (simulated IMU), Integrated Camera System/INS Algorithm.*

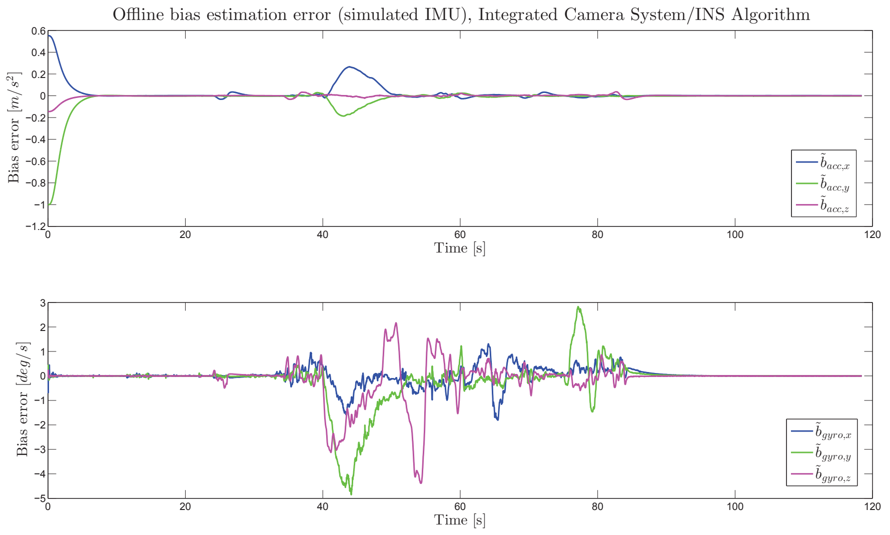

Figure 7.17: *Bias estimation error with inertial sensor measurement loss (simulated IMU), Integrated Camera System/INS Algorithm. The top-most row contains accelerometer bias estimation errors, while the second row depicts gyro bias estimation errors.*

Figure 7.17 shows the bias estimation errors. The gyro bias estimation errors in the y and z directions ($\tilde{b}_{\text{gyro,y}}$ and $\tilde{b}_{\text{gyro,z}}$) seem to be influenced the most by the outages, while all gyro bias estimates still seem to react to motion in the UAV (by comparison with Figure 7.9, which depicts the position and orientation for the error-free version of the same scenario). The accelerometer bias estimates seem to be influenced by the first measurement outage, while the second inertial measurement loss is not visible in the accelerometer bias estimation errors depicted in Figure 7.17.

7.4 Integrated Camera System/INS Algorithm - Online Implementation

This section presents the results of running the Integrated Camera System/INS Algorithm in the lab setup using measurements from the OptiTrack system as well as the inertial sensors on-board the UAV. The results are organized analogously to the corresponding Camera Measurement Algorithm section (Sec. 7.2), although this section is introduced by a stationary analysis of the received inertial measurements and the corresponding bias estimation. The input parameters presented in Table 7.5 were used in all the performed tests. Again, the design matrices have been tuned compared to its values for the simulated tests performed above. Through trial and error, the values presented in Table 7.5 seemed to produce the best results.

The estimated linear velocities presented in this section are also rotated to the inertial frame to enable direct comparison with the corresponding positions, while the original estimates (relative to the BODY frame) can be found in Appendix E. The start-up procedures of the camera system are visible in all figures depicting real-life scenarios, similarly to the situation in the testing of the Camera Measurement Algorithm in Section 7.2.

Table 7.5: *Input Parameters, Online Tests of Integrated Camera System/INS Algorithm*

$\mathbf{Q_2}$	diag [0.1, 0.1, 0.1, 0.01, 0.01, 0.01, 10, 10, 10, 1, 1, 0.3, 0.01, 0.01, 0.01]
$\mathbf{R_2}$	diag [0.01, 0.01, 0.01, 0.1, 0.1, 0.1]
Δt	0.01 s
$\bar{\mathbf{P}}_\mathbf{2}(0)$	diag [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
$\bar{\mathbf{x}}_\mathbf{2}(0)^\intercal$	$\begin{bmatrix}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\end{bmatrix}$
$\mathbf{T}$	[1000, 1000, 1000, 100, 100, 100]
$\mathbf{Q}_{\mathbf{2},\text{measError}}$	diag [0.01, 0.01, 0.01, 0.01, 0.0001, 0.01, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
$\mathbf{R}_{\mathbf{2},\text{measError}}$	diag [1000, 1000, 1000, 1000, 1000, 1000]
noMeasLimit, OT	400 time steps
noMeasLimit, IMU	300 time steps

7.4.1 Stationary Analysis of Inertial Measurements and Bias Estimation

A test to investigate the magnitudes of the actual accelerometer and gyroscope biases as well as the rate of convergence of the estimates was performed with the UAV sitting on the floor. According to theory, the measurements made by the three accelerometers will contain the appropriate component of gravity when no motion is experienced (Section 6.1). With the UAV being stationary on the floor, the non-planar angles should be zero ($\phi = 0$, $\theta = 0$), and only the received acceleration in the z direction ($a_{\text{IMU,z}}$) contains gravity (Eqs. (6.1) and (6.5)). However, the outputs from both types of inertial sensors used in this thesis also contain noise as well as biases, which is the topic of this analysis.

Figure 7.18 shows the six components of the inertial measurements obtained from the IMU: $a_{\mathrm{IMU,x}}$, $a_{\mathrm{IMU,y}}$, $a_{\mathrm{IMU,z}}$, $\omega_{\mathrm{IMU,x}}$, $\omega_{\mathrm{IMU,y}}$, and $\omega_{\mathrm{IMU,z}}$. The influence of gravity is clearly visible in $a_{\mathrm{IMU,z}}$ (the upper, right-most part of Figure 7.18), while the biases are visible in all components. That is, it is assumed that the mean stationary values of the measurements would be zero (-g for $a_{\mathrm{IMU,z}}$) if biases were not present. The fluctuations reflect the measurement noise.

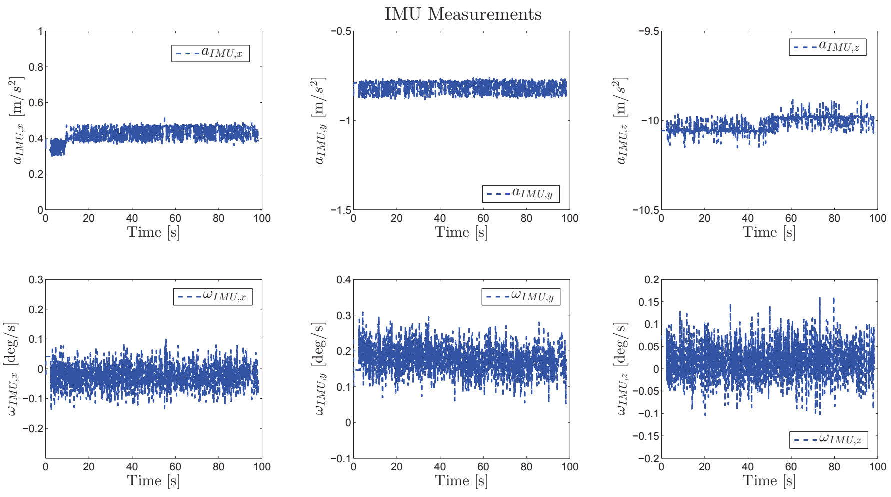

Figure 7.18: *Stationary IMU measurements.*

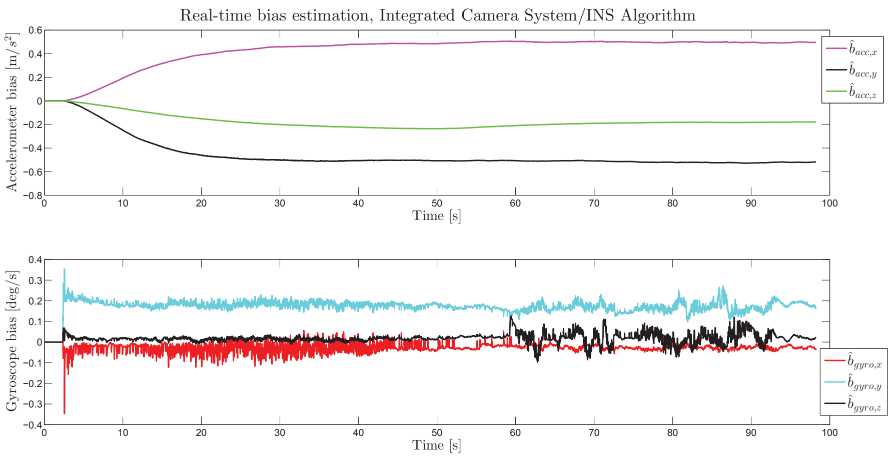

Figure 7.19: *Stationary bias estimation.*

Figure 7.19 depicts the estimated biases for the same scenario. By comparing the two figures, it can be seen that the accelerometer bias estimates (top-most row of Figure 7.19) converge quite quickly, although a steady-state error can be observed particularly in $\hat{b}_{\text{acc,y}}$. The gyroscope bias estimates converge quickly, while containing some oscillations.

7.4.2 Time Test

The updated "near real-time" aspects discussed in Subsection 6.3.4 were tested by inspection of the t_{proc} and *pause_error* intervals first presented in Section 7.2. The test is analogous to the investigation performed for the Camera Measurement Algorithm. That is, the `while` loop was run for 4000 time steps, and the interesting time intervals were logged. Table 7.6 shows that the mean processing time is still considerably shorter than the sampling interval $\Delta t = 0.01$ s, thus fulfilling the $t_{\text{proc}} < \Delta t$ demand. The maximum values for both time intervals were detected during the start-up procedures.

Table 7.6: *Results of Time Test, Integrated Camera System/INS Algorithm*

`max` (t_{proc})	0.0402 s
`mean` (t_{proc})	0.0036 s
`max` (*pause_error*)	0.0519 s
`mean` (*pause_error*)	0.0011 s

7.4.3 State Estimation under Ideal Conditions, Test 1

In this subsection, the performance of the Integrated Camera System/INS Algorithm is investigated in a realistic setting. That is, the UAV was flying around the lab while the cameras and the inertial sensors were transmitting measurements to the algorithm, which in turn estimated the system states online. The results of **Test 1** are shown here, while a second test is presented in Appendix E.2 (**Test 2**). Results of running the implemented positioning algorithm online using real measurements are shown in Figures 7.20, 7.21, and 7.22 below.

The first figure depicts the online position and orientation estimation performed by the positioning algorithm. It shows that the estimates follow the measurements while filtering out what was assumed to be measurement noise. The figure also shows that the measured and estimated ψ angles cross the $-180°$ limit without a leap occurring, i.e. the measurement preprocessing is successful.

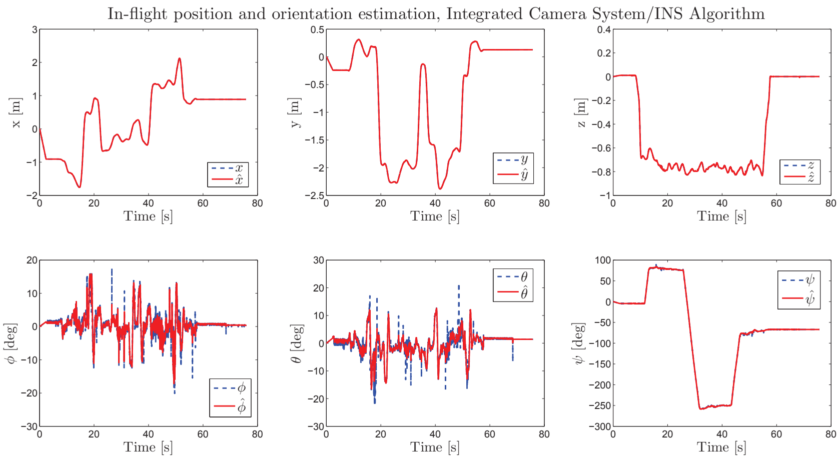

Figure 7.20: *In-flight measured state estimation, Test 1, Integrated Camera System/INS Algorithm. The blue, dashed lines are the measurements obtained from OptiTrack, while the red lines represent the estimated states. The top-most row depicts the position, while the lower row shows the orientation.*

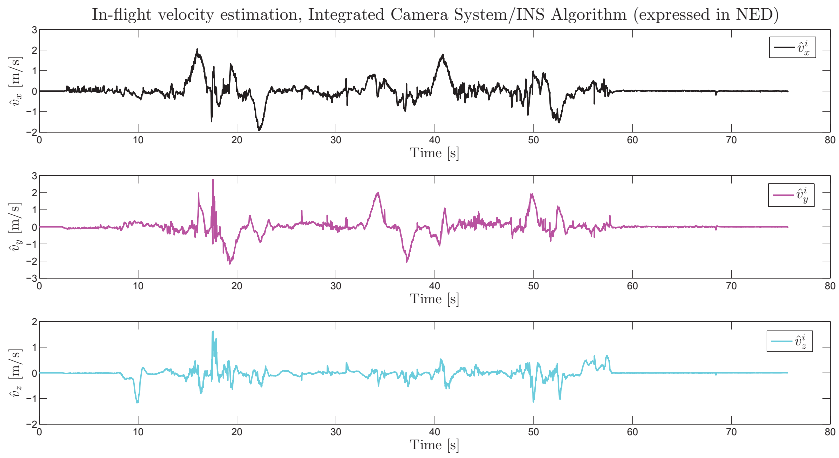

Figure 7.21: *In-flight velocity estimation, Test 1, Integrated Camera System/INS Algorithm. The velocity estimates are expressed in the inertial frame.*

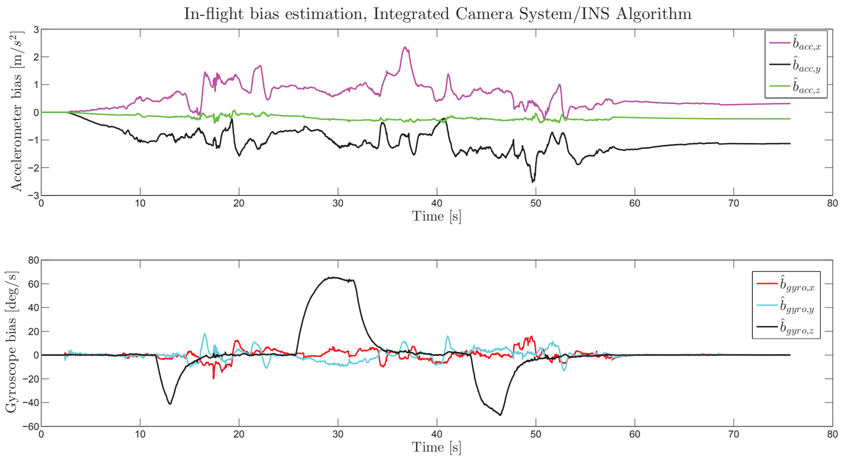

Figure 7.22: *In-flight bias estimation, Test 1, Integrated Camera System/INS Algorithm. The top-most row shows the accelerometer bias estimates, while the gyro bias estimates are shown in the lower row.*

The velocity estimates (Figure 7.21), which have no measurements to be compared to, seem reasonable when compared to the corresponding positions shown in Figure 7.20. However, they seem to contain more oscillations than their corresponding values estimated using the Camera Measurement Algorithm. The scenario ended with the UAV landing gently, which is the reason for the behavior of $\hat{v}_z^{\mathrm{i}}$ for $t \simeq 57 - 58$ s. The original estimates (expressed in BODY) are found in Appendix E.1.

Lastly, the estimated biases are influenced by motion, i.e. when the UAV is moving, the estimates are not constant (see Figure 7.20). The three figures from this test show that when the UAV is standing still ($58 < t < 78$ s), the bias estimates converge towards the true values for stationary scenarios shown in Figure 7.18.

7.4.4 State Estimation with Camera Measurement Loss

A scenario in which the trackable connected to the UAV was outside of the capture volume is presented in this subsection. For $t \simeq 27 - 28.5$ s, the camera system was unable to calculate the position and orientation of the trackable, and the algorithm therefore did not receive camera measurements for this time period. The measurements from the inertial sensors were error-free during the test.

The loss of measurements occurred at $t \simeq 27$ s, and is especially visible by close inspection of $\hat{x}$ in Figure 7.23. As mentioned earlier, the OptiTrack system outputs the last calculated position and orientation for the duration of time in which the trackable is outside of the capture volume, while the Integrated Camera System/INS Algorithm seeks to estimate the states based on prediction as well as the available IMU measurements.

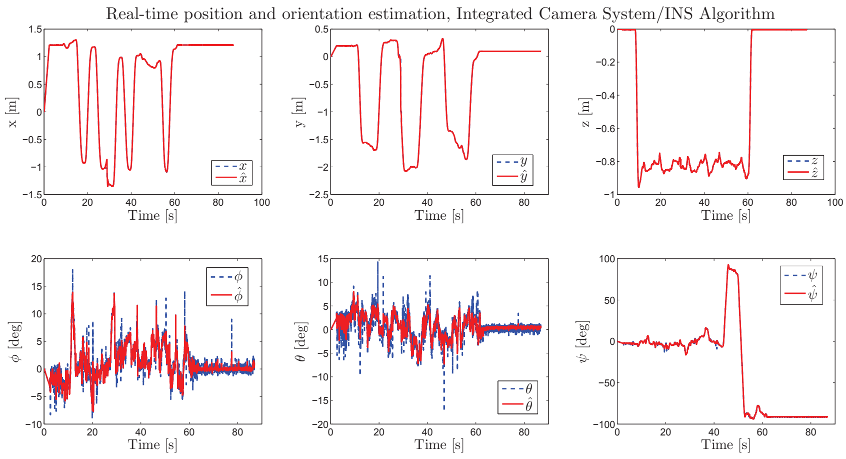

Figure 7.23: *Online position and orientation estimation with camera measurement loss, Integrated Camera System/INS Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Integrated Camera System/INS Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

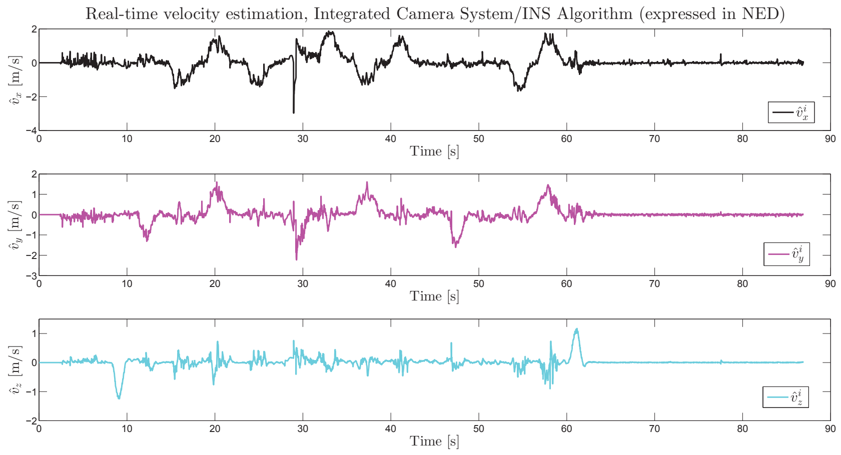

Figure 7.24: *Online velocity estimation with camera measurement loss, Integrated Camera System/INS Algorithm. The velocity estimates are expressed in the inertial frame.*

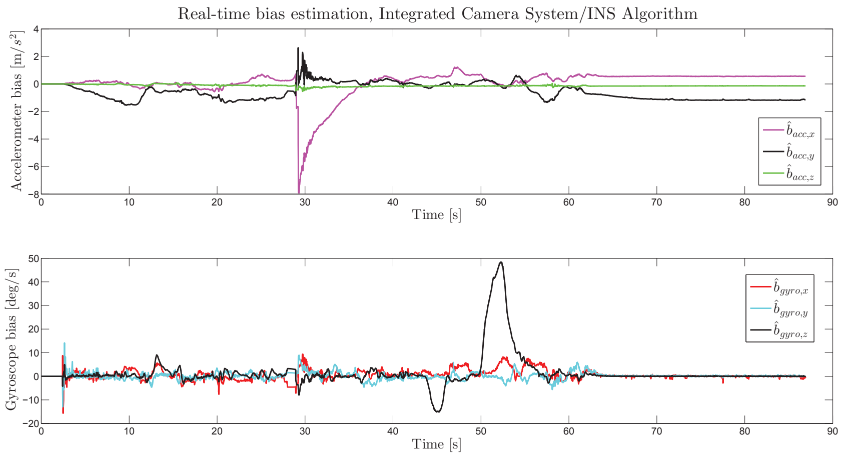

Figure 7.25: *Online bias estimation with camera measurement loss, Integrated Camera System/INS Algorithm. The top-most row shows the accelerometer bias estimates, while the gyro bias estimates are shown in the lower row.*

The impact of measurement loss on the velocity estimates is visible especially in $\hat{v}^{\mathrm{i}}_x$ (Figure 7.24), which is reasonable considering that $\hat{x}$ was most influenced as well. However, the estimate quickly follows a likely progression when the outage is over. Similarly to other velocity estimate figures, the estimates presented here are expressed in the inertial frame, while the corresponding figure of the velocity estimates expressed in b can be found in Appendix E.3.

For the estimated biases, the accelerometer values seem to be most influenced by losing the camera system measurements. The estimated gyroscope biases seem to be more affected by UAV motion than loss of camera system measurements, which correspond to the results obtained using simulated IMU measurements above.

7.5 Comparison of State Estimates

This section aims at comparing the two implemented positioning algorithms by processing measurements from the same scenario using both algorithms. For this purpose, the UAV was flown around the lab while both the camera system and the inertial sensors were collecting data. These recorded measurements were used as input to the two algorithms (the camera measurements only in the Camera Measurement Algorithm case), and the results are shown below. Only the state estimates common to both algorithms, i.e. position, orientation, and linear velocity, are depicted. The input parameters given in Tables 7.2 and 7.5 were used for the Camera Measurement and Integrated Camera System/INS Algorithms, respectively.

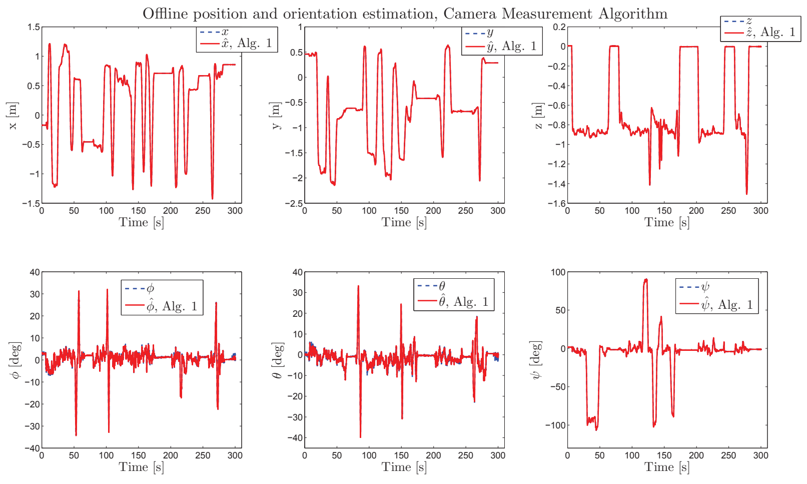

Figure 7.26: *Offline position and orientation estimation, Camera Measurement Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Camera Measurement Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

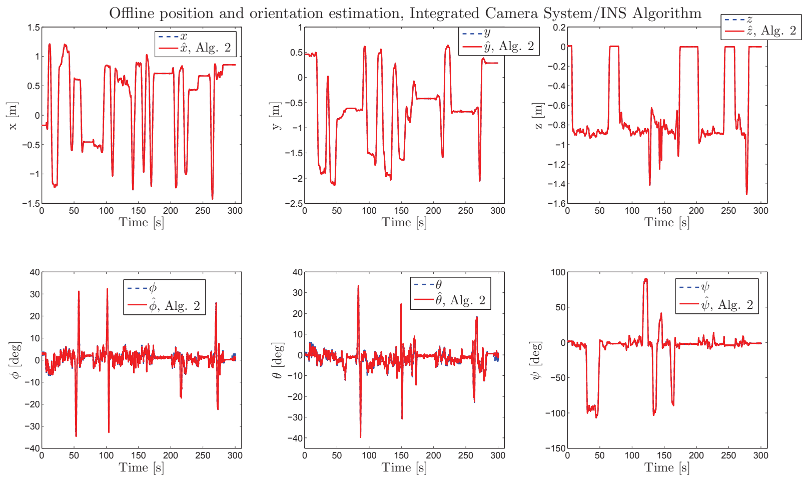

Figure 7.27: *Offline position and orientation estimation, Integrated Camera System/INS Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Integrated Camera System/INS Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

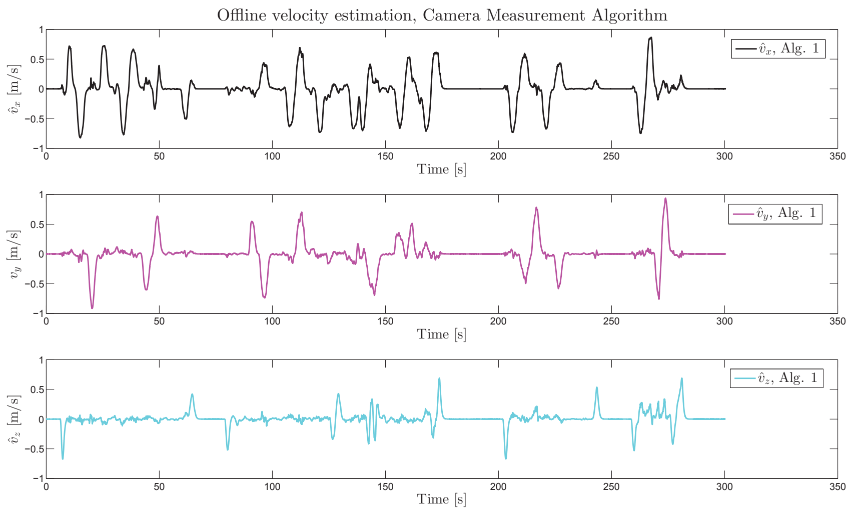

Figure 7.28: *Offline velocity estimation, Camera Measurement Algorithm. The velocities are depicted relative to the (original) BODY frame.*

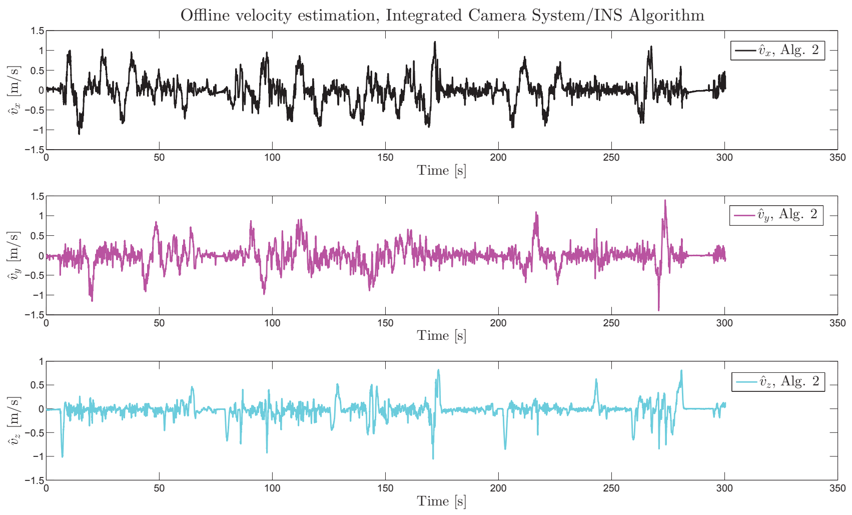

Figure 7.29: *Offline velocity estimation, Integrated Camera System/INS Algorithm. The velocities are depicted relative to the (original) BODY frame.*

The results will be discussed, along with the rest of this chapter, in the next chapter.

Chapter 8

Discussion

In this chapter, the two proposed positioning algorithms are discussed and compared with the aim of obtaining an applicable and accurate solution for the intended Parrot AR.Drone 2.0 lab scenario in mind. In particular, the results presented in Chapter 7 are analyzed, and decisions made with regards to solving the assignment given are assessed after implementation of the two chosen setups has been performed. Furthermore, the role of the positioning algorithms in a larger system for indoor motion control of the UAV is elaborated.

8.1 Camera Measurement Algorithm

The first positioning algorithm implemented in this thesis, the Camera Measurement Algorithm, was presented in Chapter 5. It uses measurements from the camera system only to estimate the position, orientation, linear, and angular velocities of the Parrot AR.Drone 2.0 online.

The results of testing the algorithm using simulated measurements show that the estimates of both the measured and unmeasured states quickly converge to their true values (and follow them for the duration of the simulation) for arbitrary initial conditions and scenarios in which the system is subject to process and measurement noise. In the real-time tests performed in the lab, the accuracy of the velocity estimates is difficult to inspect due to there being no measurements available for these states. However, a comparison with the corresponding position and orientation estimates and their measurements in conjunction with the simulation results provide a strong implication for the performance of the unmeasured state estimation being satisfactory.

The position and orientation estimates are compared to the camera system measurements in the online tests. The reception of outliers is a quite likely scenario in real-life systems, which is the reason why a mechanism for identifying and discarding faulty measurements has been implemented for both positioning algorithms, resulting in smoother measurements, especially for the orientation. However, the mechanism may have been too aggressive, with the result that correct measurements have been discarded. This may have been the case for ϕ and θ in particular, which are control variables for linear motion, and thus experience spikes when UAV motion is induced. However, the position estimates follow their measurements in a satisfactory manner, and outliers are discarded.

It should be noted that no ground truths are available in the lab setup. That is, the correctness of the measurements from the camera system against which the position and orientation estimates are compared cannot be completely verified. The reason is that there is no redundancy in the available sensors (time did not allow a comparison with inertial measurements). However, the accuracy of the planar, linear measurements from the OptiTrack system was briefly tested in Section 3.4 with seemingly satisfactory results. This test alone should not be used as a complete verification of the authenticity of the measurements, but its results can be used as an argument for using the trajectories of the OptiTrack measurements as the overall course to follow. That is, noise filtering (especially when estimating the orientation) is performed, but the noise-reduced measurements should be (and are) followed by the estimates.

8.2 Integrated Camera System/INS Algorithm

The second positioning algorithm implemented in this thesis, the Integrated Camera System/INS Algorithm, was presented in Chapter 6. It fuses the camera system measurements with outputs from the on-board inertial sensors to estimate the position, orientation, and linear velocity of the UAV as well as the inertial sensor biases.

The performance of the algorithm was first investigated using recorded camera system and simulated IMU measurements, a test which showed satisfactory results. The measured state estimates follow their ground truths, while the velocity and bias estimates quickly converge to theirs, although some fluctuations in the bias estimates are experienced. The oscillations seem to coincide with movement in the corresponding position or orientation state.

The tests performed in the lab reveal that the velocity estimates contain fluctuations about their mean values, although they seem to follow likely overall trajectories when compared to the corresponding positions. Similarly to the Camera Measurement Algorithm results, the simulations in conjunction with online behavior imply that the velocity estimation is satisfactory despite the observed noise, which was not unexpected. That is, the camera system delivers measurements of quite good quality, while IMU measurements containing noise and biases are added in the Integrated Camera System/INS Algorithm, resulting in increased noise in the system.

The bias estimates are seemingly influenced by UAV motion, resulting in non-constant values for periods of time in which the UAV was not stationary, while converging towards the values observed in measurements from the IMU when stationary. Steady-state errors were also observed in the online tests, resulting in a need for further tuning and testing of the system. However, in the simulations, the bias estimates quickly approached their true values. Thus, the reason for the observed steady-state errors in the lab tests may be connected to other factors than the state estimation. Furthermore, the velocity estimates, which might be considered more important with the goal of this thesis in mind (a well-functioning and accurate positioning algorithm for use in a proposed testbed), seem to be satisfactory (to the extent explained above) despite the obtained bias estimates.

The position and orientation estimates follow their respective measurements successfully while filtering out what was assumed to be measurement noise. However, the discussion of whether the outlier mechanism is too aggressive presented above applies to the Integrated Camera System/INS Algorithm as well.

In this thesis, it was assumed that the UAV with the trackable attached to it constitutes a rigid body. However, the tests performed in the lab setup revealed that this may not be the case. That is, the trackable (attached to the top of the UAV) seemed to move independently of the UAV. This may have caused inaccurate camera measurements as well as a moving center of gravity. Furthermore, the IMU measurements may have added more noise than observed in the stationary test (Figure 7.18) as well as varying biases when the UAV was moving.

8.3 Comparison and Final Remarks

The main difference between the Camera Measurement and Integrated Camera System/INS Algorithms is that the first algorithm uses measurements from one source, while the latter is an integrated solution utilizing two sensor systems. As a consequence, the two inhabit different advantages and disadvantages which are elaborated here.

The two algorithms were run using identical measurements in Section 7.5, a test which confirmed that the velocity estimates from the Integrated Camera System/INS Algorithm contain more noise than their equivalents resulting from use of the Camera Measurement Algorithm. As mentioned above, this was not unexpected because the IMU adds noise to the system. The test also showed that when estimating measured states, either algorithm may be used with excellent results.

When using the Camera Measurement Algorithm, estimates of angular velocities are obtained in addition to the position, orientation, and linear velocity estimates, while measurements of the angular velocities are available in the Integrated Camera System/INS Algorithm. However, the angular velocities are the estimates from the Camera Measurement Algorithm which seem to contain the most noise in the online tests, making the choice between measured and estimated angular velocities less obvious. Nevertheless, the estimates seem to behave better than unfiltered gyroscope measurements.

An argument for using the Integrated Camera System/INS Algorithm in the proposed lab setup could be the provided redundancy in sensors, especially with continued state estimation during camera measurement outages in mind. An investigation of whether the integrated solution using additional IMU measurements is able to produce satisfactory state estimates for longer camera measurement outages than the corresponding algorithm based on camera measurements only was performed. Tests were conducted for simulated scenarios as well as in the lab. The simulations show that the smallest estimation errors result from use of the Camera Measurement Algorithm when measurement outages of equal duration were experienced, which, to some degree, was unexpected. An explanation might be that the IMU measurements contain more noise than the camera system, and it seems that emphasizing the modelled behavior using the Camera Measurement Algorithm is a better choice than exploiting the IMU measurements using the integrated solution. However, time did not allow sufficient tuning and testing of the Integrated Camera System/INS Algorithm, which is a likely part of the reason for the obtained results. Furthermore, the simulated scenarios used for the two algorithms were not identical due to the Integrated Camera System/INS Algorithm using real, recorded camera measurements, and a direct comparison thus cannot be made.

A conclusion with regards to the performance of the algorithms during camera measurement outages in the lab is difficult do draw due to the nature of the problem. That is, no measurements are available for comparison, which was the reason for performing the simulations. However, the results obtained online show that all state estimates quickly approach satisfactory values when a new measurement is received. A conclusion that one of the algorithms provides better state estimation during camera measurement outages cannot be drawn based on this thesis alone, due to lack of extensive testing and possibly imperfect tuning. Further tuning and testing should be performed with this in mind.

The results of losing camera measurements as opposed to inertial sensor measurements were also investigated for the Integrated Camera System/INS Algorithm. In simulations, camera measurement outages seem to produce the highest increase in position and orientation estimation errors, while inertial sensor outages appear to have the most influence on the bias estimates (the real-life impacts of inertial measurement loss were not investigated due to the limited time available). However, the impact of camera measurement loss on the acceleration biases in the lab is visible in Figure 7.25. This could be a result of imperfect tuning, when compared to the simulation of a similar scenario. Furthermore, the velocity estimates seem to be somewhat affected by both types of outages. Limits for accepting continued state estimation without inputs can be set for each measurement source individually, depending on the priorities of the user.

8.3.1 The Positioning Algorithms in a Real-Time, Closed-Loop System

A positioning algorithm represents one subsystem of an overall motion control system. As a consequence, the performances of the two implemented algorithms should be seen in connection with the larger system of which they will be part.

An aspect which has not yet been discussed is the "near real-time" demands outlined in Sections 5.2 and 6.3.4. The delivered state estimates must be updated in a timely manner for a positioning algorithm to be useful in the proposed test setup, a challenge which was addressed using a "near real-time" approach in which the sampling rates of both sensor systems are assumed constant, and the respective algorithm updates its measurements according to these sampling rates. Furthermore, a processing time shorter than the system sampling rate is required. The results of choosing this approach were investigated through testing of the $t_{\text{proc}} < \Delta t$ demand with a Δt satisfying the requirement of timely estimate updates, and the approach seems to be satisfactory. However, the state estimates have not been used in an overall motion control system due to the limited time available, a test which must be performed before the success of the "near real-time" approach can be verified.

The measures taken to increase the robustness of the positioning algorithms may be labelled passive, i.e. when an error occurs, the algorithms merely perform as well as possible under the given circumstances. This is due to the sensors being outside the scope of the positioning algorithms, they represent a separate system which cannot be controlled by the user of the positioning algorithm. In addition, the control block in a motion control system often contains much of the error handling in the overall system. However, the actions taken to discard outliers and improve state estimation during measurement outages should ensure that acceptable estimates are sent to the control system for a longer period of time than if no actions were taken.

The goal of implementing an accurate positioning algorithm for the Parrot AR.Drone 2.0 suitable for online use in the proposed lab setup is fulfilled to the extent outlined in the discussion above. That is, further tuning and extensive testing of the algorithms need be performed. Even more importantly, the abilities of the algorithms to close the loop in a motion control system must be tested. However, both algorithms seem to perform in a satisfactory manner (the Integrated Camera System/INS Algorithm contains more noise due to the added IMU measurements) for scenarios without measurement errors, while the extensions to increase robustness should be developed further.

Considering the results obtained and discussed in this thesis, the Camera Measurement Algorithm would be the current choice for the Parrot AR.Drone 2.0 lab setup because of the quality of its velocity estimates, and the fact that the redundancy in sensors in the Integrated Camera System/INS Algorithm does not seem to have been fully utilized. That is, with more time available, the state estimates during measurement outages could have been optimized through further tuning and testing, making the decision a more difficult one.

Chapter 9

Conclusion

This thesis is motivated by the use of UAVs for acquiring measurements of the scene in a system for estimation of ice properties. Such systems are needed to ensure safety when conducting marine operations in Arctic seas. As a possible testbed for UAV guidance and estimation algorithms, the use of the Parrot AR.Drone 2.0 quadcopter has been proposed, necessitating a positioning algorithm for the UAV. The goal of this thesis was to develop an accurate indoor positioning algorithm suitable for online use in the proposed lab setup.

The first part of this thesis included a brief overview of methods for indoor positioning, focusing on their individual system setups as well as relative strengths and weaknesses. A more detailed presentation of camera-based indoor positioning was given before the main (camera-based) measurement source of this thesis was presented. This camera system provides position and orientation measurements in six DOF for a rigid body through tracking of reflective markers attached to the object of interest.

Strong implications for satisfactory performance by the Camera Measurement Algorithm in an online scenario have been shown, although a "near real-time" implementation only is performed. The algorithm performs accurate estimation of position, orientation, linear and angular velocities in simulations, and appears to obtain the same results online. Furthermore, faulty measurements are discarded and a mechanism for maintaining satisfactory state estimation during short measurement outages is included. Hence, the algorithm should be applicable to the intended Parrot AR.Drone 2.0 lab setup. However, further testing of the functionality of the algorithm, especially during measurement outages, along with experiments in which the algorithm is used in a motion control system should be performed.

The Integrated Camera System/INS Algorithm performs accurate position, orientation, linear velocity and sensor bias estimation in simulations, while tests conducted of its online behavior show increased noise in the velocity estimates when compared to the Camera Measurement Algorithm. Furthermore, its performance during camera measurement outages cannot be validated based on this thesis alone. Further tuning and testing should be performed, especially for use in a motion control system

9.1 Future Work

The algorithms implemented in this thesis are ready-to-use with the camera system in the lab. However, testing of the algorithms as part of a larger system is not performed, and the estimates are not optimized. For future work, the following is proposed:

- Testing of both algorithms used in a guidance system
- Further tuning of the extended Kalman filters
- Investigation of results of a real-time implementation (as opposed to the "near real-time" approach chosen in this thesis)
- Testing of a less aggressive outlier mechanism
- Investigation of results with the trackable attached differently

Bibliography

[1] S.M. Easa. “Space Resection in Photogrammetry using Collinearity Condition Without Linearisation”. In: *Maney Publishing Survey Review Ltd.* (2010).

[2] *Flex 13 Data Sheet.* NaturalPoint Inc. 2012.

[3] F. Forno, G. Malnati, and G. Portelli. “Design and Implementation of a Bluetooth Ad Hoc Network for Indoor Positioning”. In: *IEE Proceedings: Software* (2005).

[4] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons Ltd., 2011.

[5] G. Glanzer et al. “Semi-Autonomous Indoor Positioning Using MEMS-Based Inertial Measurement Units and Building Information”. In: *Proceedings of the 6th Workshop on Positioning, Navigation and Communication* (2009).

[6] B. Hartmann, N. Link, and G.F. Trommer. “Indoor 3D Position Estimation Using Low-Cost Inertial Sensors and Marker-Based Video-Tracking”. In: 2010.

[7] N.A. Khanina, E.V. Semeikina, and D.V. Yurin. “Color Blob and Line Detection in Scale-Space”. In: *Pattern Recognition and Image Analysis, Vol. 21, No. 2* (2011).

[8] A. Kotanen et al. “Experiments on Local Positioning with Bluetooth”. In: 2003.

[9] A. Lim and K. Zhang. “A Robust RFID-Based Method for Precise Indoor Positioning”. In: Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2006.

[10] R. Mautz and S. Tilch. “Survey of Optical Indoor Positioning Systems”. In: *IEEE* (2011).

[11] O. Mirabella et al. “A Motion Capture System for Sport Training and Rehabilitation”. In: *4th International Conference on Human System Interactions* (2011).

[12] S. Piskorski et al. *AR.Drone Developer Guide, SDK 2.0.* Parrot.

[13] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson Learning, 2008.

[14] *Tracking Tools 2.4.0 User's Guide*. NaturalPoint Inc. DBA OptiTrack. 2012.

[15] *Tracking Tools User's Manual Version 2.0*. NaturalPoint Inc. 2012.

[16] B. Vik. *Integrated Satellite and Inertial Navigation Systems*. 2012.

[17] Ylva S. Vintervold. “Methods for Indoor Positioning”.

[18] M. Wallbaum and O. Spaniol. “Indoor Positioning Using Wireless Local Area Networks”. In: *Proceedings - IEEE John Vincent Atanasoff International Symposium on Modern Computing* (2006).

[19] H. Zhang et al. “Robust Color Circle-Marker Detection Algorithm based on Color Information and Hough Transformation”. In: *Optical Engineering 48* (2009).

Appendix A

Rotation and Transformation Matrix Properties

A.1 Rotation Matrices

A rotation matrix describes the rotation between two frames b and c:

$$\mathbf{a}^c = \mathbf{R}_b^c(\alpha)\mathbf{a}^b \tag{A.1}$$

Here, $\mathbf{a}^c$ is a vector expressed in coordinate system c, and $\mathbf{a}^b$ is the same vector referenced to coordinate system b. $\mathbf{R}_b^c(\alpha)$ is the rotation matrix between the frames. Thus, frame b is rotated by the angle α to obtain frame c.

A rotation matrix is an element in *SO(3)*, which is the *special orthogonal group of order 3*:

$$\mathit{SO(3)} = \{\mathbf{R}|\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{R} \text{ is orthogonal, det } \mathbf{R} = 1\} \tag{A.2}$$

SO(3) is a subset of all *orthogonal matrices of order 3*, *O(3)*, where *O(3)* is defined as

$$\mathit{O(3)} \triangleq \{\mathbf{R}|\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}\} \tag{A.3}$$

A rotation matrix $\mathbf{R} \in \mathit{SO(3)}$ satisfies the following properties:

- $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$
- det $\mathbf{R} = 1$
- $\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$
- $\mathbf{R}_b^a\mathbf{R}_c^b = \mathbf{R}_c^a$

For further details, see [4].

A.2 Transformation Matrices

The relationship between a body-fixed angular velocity vector $\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}}$ and the Euler rate vector $\dot{\boldsymbol{\Theta}}_{\mathrm{ib}}$ is described by a transformation matrix $\mathbf{T}_{\Theta}(\boldsymbol{\Theta}_{\mathrm{ib}})$:

$$\dot{\boldsymbol{\Theta}}_{\mathrm{ib}} = \mathbf{T}_{\Theta}(\boldsymbol{\Theta}_{\mathrm{ib}})\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} \tag{A.4}$$

The transformation matrix can be derived as follows:

$$\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/i}} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}^{\intercal}_{\mathrm{x},\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}^{\intercal}_{\mathrm{x},\phi}\mathbf{R}^{\intercal}_{\mathrm{y},\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \triangleq \mathbf{T}^{-1}_{\Theta}(\boldsymbol{\Theta}_{\mathrm{ib}})\dot{\boldsymbol{\Theta}}_{\mathrm{ib}} \tag{A.5}$$

For further details, see [4].

Appendix B

Skew-Symmetrical Matrices

.

Skew-Symmetry of Matrix A matrix $\mathbf{S} \in SS(n)$, where $SS(n)$ is the set of skew-symmetric matrices of order n, is skew-symmetrical if

$$\mathbf{S} = -\mathbf{S}^{\mathsf{T}} \tag{B.1}$$

Cross Product Operator The vector cross product $\times$ is defined by

$$\boldsymbol{\lambda} \times \mathbf{a} \triangleq \mathbf{S}(\boldsymbol{\lambda})\mathbf{a}, \tag{B.2}$$

where $\mathbf{S}(\boldsymbol{\lambda}) \in SS(3)$ is skew-symmetric and defined as

$$\mathbf{S}(\boldsymbol{\lambda}) = -\mathbf{S}^{\mathsf{T}}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{B.3}$$

For further details, see [4].

Appendix C

Elements of System Function Jacobians

The elements of the Jacobians of the two system functions are shown here. The larger elements of both Jacobians are displayed at the end of the appendix due to lack of space.

C.1 Camera Measurement Algorithm

$$\mathbf{\Phi}_{\mathbf{1},1} = \begin{bmatrix} 1 & 0 & 0 & \Delta t \cdot [(c\psi s\theta c\phi + s\psi s\phi) \cdot v_y + (s\psi c\phi - c\psi s\phi s\theta) \cdot v_z] \\ 0 & 1 & 0 & \Delta t \cdot [(c\phi s\theta s\psi - c\psi s\phi) \cdot v_y - (s\theta s\psi s\phi + c\psi c\phi) \cdot v_z] \\ 0 & 0 & 1 & \Delta t \cdot [c\theta c\phi \cdot v_y - c\theta s\phi \cdot v_z] \\ 0 & 0 & 0 & 1 + \Delta t \cdot [c\phi t\theta \cdot \omega_y - s\phi t\theta \cdot \omega_z] \end{bmatrix} \tag{C.1}$$

$$\mathbf{\Phi}_{\mathbf{1},3} = \begin{bmatrix} \Delta t \cdot [s\psi s\phi + c\psi c\phi s\theta] & 0 & 0 & 0 \\ \Delta t \cdot [s\theta s\psi c\phi - c\psi s\phi] & 0 & 0 & 0 \\ \Delta t \cdot c\theta c\phi & 0 & 0 & 0 \\ 0 & \Delta t & \Delta t \cdot s\phi t\theta & \Delta t \cdot c\phi t\theta \end{bmatrix} \tag{C.2}$$

$$\mathbf{\Phi}_{\mathbf{1},4} = \begin{bmatrix} 0 & 0 & 0 & -\Delta t[s\phi \cdot \omega_y + c\phi \cdot \omega_z] \\ 0 & 0 & 0 & \Delta t[\frac{c\phi}{c\theta} \cdot \omega_y - \frac{s\phi}{c\theta} \cdot \omega_z] \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.3}$$

$$\mathbf{\Phi}_{\mathbf{1},5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \Delta t[s\phi \sec\theta t\theta \cdot \omega_y + c\phi \sec\theta t\theta \cdot \omega_z] & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.4}$$

$$\mathbf{\Phi}_{\mathbf{1},6} = \begin{bmatrix} 0 & 0 & \Delta t \cdot c\phi & -\Delta t \cdot s\phi \\ 0 & 0 & \Delta t \cdot \frac{s\phi}{c\theta} & \Delta t \cdot \frac{c\phi}{c\theta} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.5}$$

$$\mathbf{\Phi_{1}}_{,7} = \mathbf{\Phi_{1}}_{,8} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.6}$$

$$\mathbf{\Phi_{1}}_{,9} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.7}$$

C.2 Integrated Camera System/INS Algorithm

$$\mathbf{\Phi_{2}}_{,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\Delta t & -\Delta t \cdot s\phi t\theta & -\Delta t \cdot c\phi t\theta \\ 0 & 0 & 0 & -\Delta t \cdot c\phi & \Delta t \cdot s\phi \end{bmatrix} \tag{C.8}$$

$$\mathbf{\Phi_{2}}_{,5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -\Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 - \Delta t \cdot \frac{1}{T_1} \end{bmatrix} \tag{C.9}$$

$$\mathbf{\Phi_{2}}_{,6} = \begin{bmatrix} 0 & 0 & 0 & -\Delta t \cdot \frac{s\phi}{c\theta} & -\Delta t \cdot \frac{c\phi}{c\theta} \\ 0 & 0 & 0 & 0 & 0 \\ -\Delta t & 0 & 0 & 0 & 0 \\ 0 & -\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.10}$$

$$\mathbf{\Phi_{2}}_{,7} = \mathbf{\Phi_{2}}_{,8} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.11}$$

$$\mathbf{\Phi_{2}}_{,9} = \begin{bmatrix} 1 - \Delta t \cdot \frac{1}{T_2} & 0 & 0 & 0 & 0 \\ 0 & 1 - \Delta t \cdot \frac{1}{T_3} & 0 & 0 & 0 \\ 0 & 0 & 1 - \Delta t \cdot \frac{1}{T_4} & 0 & 0 \\ 0 & 0 & 0 & 1 - \Delta t \cdot \frac{1}{T_5} & 0 \\ 0 & 0 & 0 & 0 & 1 - \Delta t \cdot \frac{1}{T_6} \end{bmatrix} \tag{C.12}$$

$$\mathbf{\Phi_{1,2}} = \tag{C.13}$$
$$\begin{bmatrix} \Delta t[-s\theta c\psi \cdot v_x + c\psi c\theta s\phi \cdot v_y + c\psi c\phi c\theta \cdot v_z] & \Delta t[-s\psi c\theta \cdot v_x - (s\psi s\theta s\phi + c\theta c\phi) \cdot v_y + (c\psi s\phi - s\psi c\phi s\theta) \cdot v_z] & \Delta t \cdot c\psi c\theta & \Delta t[c\psi s\theta s\phi - s\psi c\phi] \\ \Delta t[-s\theta s\psi \cdot v_x + s\phi c\theta s\psi \cdot v_y + c\theta s\psi c\phi \cdot v_z] & \Delta t[c\psi c\theta \cdot v_x + (s\phi s\theta c\psi - s\psi c\phi) \cdot v_y + (s\theta c\psi c\phi + s\psi s\phi) \cdot v_z] & \Delta t \cdot s\psi c\theta & \Delta t[c\psi c\phi + s\phi s\theta s\psi] \\ -\Delta t[c\theta \cdot v_x + s\theta s\phi \cdot v_y + s\theta c\phi \cdot v_z] & 0 & -\Delta t \cdot s\theta & \Delta t \cdot c\theta s\phi \\ \Delta t[s\phi \sec^2\theta \cdot \omega_y + c\phi \sec^2\theta \omega_z] & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{\Phi_{2,1}} = \tag{C.14}$$
$$\begin{bmatrix} 1 & 0 & 0 & \Delta t[(c\psi s\theta c\phi + s\psi s\phi) \cdot v_y + (s\psi c\phi - c\psi s\phi s\theta) \cdot v_z] & \Delta t[(-s\theta c\psi) \cdot v_x + c\psi c\theta s\phi \cdot v_y + c\psi c\phi c\theta \cdot v_z] \\ 0 & 1 & 0 & \Delta t[(c\phi s\theta s\psi - c\psi s\phi) \cdot v_y - (s\theta s\psi s\phi + c\psi c\phi) \cdot v_z] & \Delta t[-s\theta s\psi v_x + s\phi c\theta s\psi \cdot v_y + c\theta s\psi c\phi \cdot v_z] \\ 0 & 0 & 1 & \Delta t[c\theta c\phi \cdot v_y - c\theta s\phi \cdot v_z] & -\Delta t[c\theta \cdot v_x + s\theta s\phi \cdot v_y + s\theta c\phi \cdot v_z] \\ 0 & 0 & 0 & 1 + \Delta t[c\phi t\theta \cdot (\omega_{\mathrm{IMU,y}} - b_{\mathrm{gyro,y}}) - s\phi t\theta \cdot (\omega_{\mathrm{IMU,z}} - b_{\mathrm{gyro,z}})] & \Delta t[s\phi \sec^2\theta \cdot (\omega_{\mathrm{IMU,y}} - b_{\mathrm{gyro,y}}) + c\phi \sec^2\theta \cdot (\omega_{\mathrm{IMU,z}} - b_{\mathrm{gyro,z}})] \\ 0 & 0 & 0 & -\Delta t[s\phi \cdot (\omega_{\mathrm{IMU,y}} - b_{\mathrm{gyro,y}}) + c\phi \cdot (\omega_{\mathrm{IMU,z}} - b_{\mathrm{gyro,z}})] & 1 \end{bmatrix}$$

$$\mathbf{\Phi_{2,2}} = \tag{C.15}$$
$$\begin{bmatrix} \Delta t[-s\psi c\theta \cdot v_x - (s\psi s\theta s\phi + c\psi c\phi) \cdot v_y + (c\psi s\phi - s\psi c\phi s\theta) \cdot v_z] & \Delta t \cdot c\psi c\theta & \Delta t[c\psi s\theta s\phi - s\psi c\phi] & \Delta t[s\psi s\phi + c\psi c\phi s\theta] & 0 \\ \Delta t[c\psi c\theta \cdot v_x + (s\phi s\theta c\psi - s\psi c\phi) \cdot v_y + (s\theta c\psi c\phi + s\psi s\phi) \cdot v_z] & \Delta t \cdot s\psi c\theta & \Delta t[c\psi c\phi + s\phi s\theta s\psi] & \Delta t[s\theta s\psi c\phi - c\psi s\phi] & 0 \\ 0 & -\Delta t \cdot s\theta & \Delta t \cdot c\theta s\phi & \Delta t \cdot c\theta c\phi & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{\Phi_{2}}_{,4} = \tag{C.16}$$

$$\begin{bmatrix} 0 & 0 & 0 & \Delta t[\frac{c\phi}{c\theta} \cdot (\omega_{\mathrm{IMU,y}} - b_{\mathrm{gyro,y}}) - \frac{s\phi}{c\theta} \cdot (\omega_{\mathrm{IMU,z}} - b_{\mathrm{gyro,z}})] & \Delta t[s\phi \sec\theta t\theta \cdot (\omega_{\mathrm{IMU,y}} - b_{\mathrm{gyro,y}}) + c\phi \sec\theta t\theta \cdot (\omega_{\mathrm{IMU,z}} - b_{\mathrm{gyro,z}})] \\ 0 & 0 & 0 & 0 & -\Delta t \cdot c\theta g \\ 0 & 0 & 0 & \Delta t \cdot c\phi c\theta g & -\Delta t \cdot s\theta s\phi g \\ 0 & 0 & 0 & -\Delta t \cdot s\phi c\theta g & -\Delta t \cdot s\theta c\phi g \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Appendix D

Results, Camera Measurement Algorithm

D.1 Simulated Measurements

D.1.1 Performance of State Estimation, Simulation Scenario 2

The second simulation conducted to investigate the performance of the state estimation performed by the Camera Measurement Algorithm (without any measurement errors) is presented here. The initial conditions specific for the scenario in the second simulation as well as the results are shown.

The following initial state vector and estimates were used:

$$\mathbf{x_1}(0) = \left[0, 0, 0.4, 0, 0, 0, 0.8, 0, 0, 0, 0, 7\right]^{\mathsf{T}} \tag{D.1}$$

$$\bar{\mathbf{x}}_{\mathbf{1}}(0) = \left[0, 0, 3, 0, 0, \pi, 0, 1, 0, 0, 2, 0\right]^{\mathsf{T}} \tag{D.2}$$

The initial v_x and ω_z are constant and non-zero, and all velocities are modelled constant in the plant. The result is that in a noise-free simulation, the UAV should hover in a circular manner. The results of running the algorithm using the input parameters of Table 7.1, the process and measurement noise given in Section 7.1, and the initial conditions given by Eqs. (D.1) and (D.2) are shown in Figures D.1 and D.2 for the measured and unmeasured states, respectively.

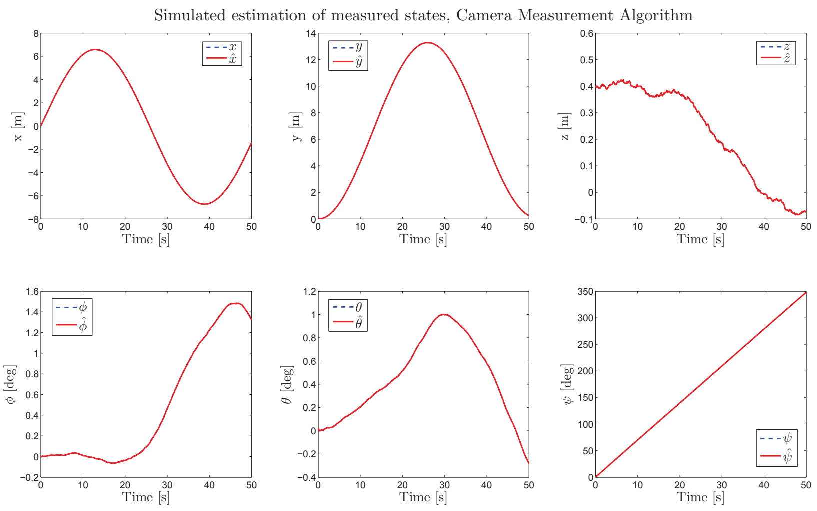

Figure D.1: *Measured state estimation, Simulation Scenario 2, Camera Measurement Algorithm. The UAV is hovering with constant linear velocity v_x and constant angular velocity ω_z. The real values from the plant are shown as blue, dashed lines while the estimates are represented by red lines. The top-most row depicts the position, while the lower row shows the orientation.*

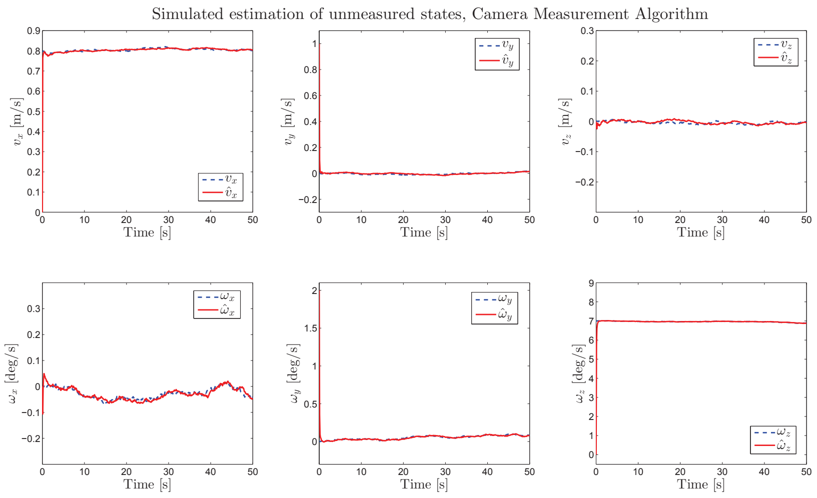

Figure D.2: *Unmeasured state estimation, Simulation Scenario 2, Camera Measurement Algorithm. The UAV is hovering with constant linear velocity v_x and constant angular velocity ω_z. The real values from the plant are shown as blue, dashed lines while the estimates are represented by red lines. The linear velocities are shown in the top-most row, while the angular velocities are depicted in the second row.*

D.1.2 Impacts of Measurement Loss, Simulation Scenario 2

The second simulation conducted to investigate the performance of the Camera Measurement Algorithm when measurement losses are experienced is presented here. The initial conditions specific for the scenario in the second simulation are given by Eqs. (D.1) and (D.2), while the input parameters were shown in Table 7.1.

Process and measurement noise was added to the simulation, equivalently to **Scenario 1** (see Section 7.1) as well as the error-free version of **Simulation Scenario 2** presented above. Two occurrences of measurement loss were experienced during the simulation, the first after $t = 20$ seconds and the second at $t = 41$ seconds (analogously to the situation in **Simulation Scenario 1**). The results are shown in Figures D.3 and D.4 for the measured and unmeasured states, respectively. The measurement losses are visible in the figures as the regions in which the estimation errors increase.

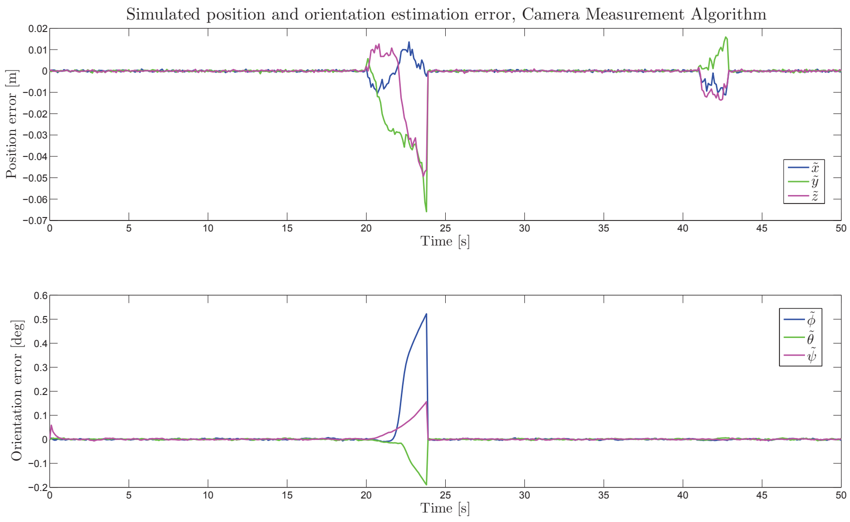

Figure D.3: *Impact of measurement loss on position and orientation estimation, Simulation Scenario 2, Camera Measurement Algorithm. The UAV is hovering with constant linear velocity v_x and constant angular velocity ω_z, and the measurements are lost twice. The top-most row shows the position error, while the lower row depicts orientation error.*

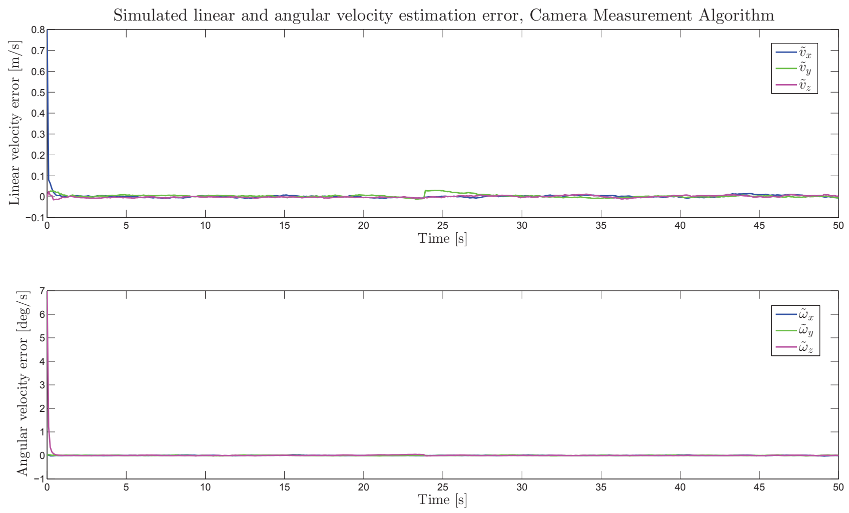

Figure D.4: *Impact of measurement loss on linear and angular velocity estimation, Simulation Scenario 2, Camera Measurement Algorithm. The UAV is hovering with constant linear velocity v_x and constant angular velocity ω_z, and the measurements are lost twice. The top-most row shows the error in linear velocities, while the lower row depicts angular velocity error.*

D.2 Online State Estimation

D.2.1 State Estimation under Ideal Conditions, Test 1

The estimated linear and angular velocities for the first test of state estimation under ideal conditions performed in Section 7.2 (**Test 1**), expressed in the BODY frame, are depicted here.

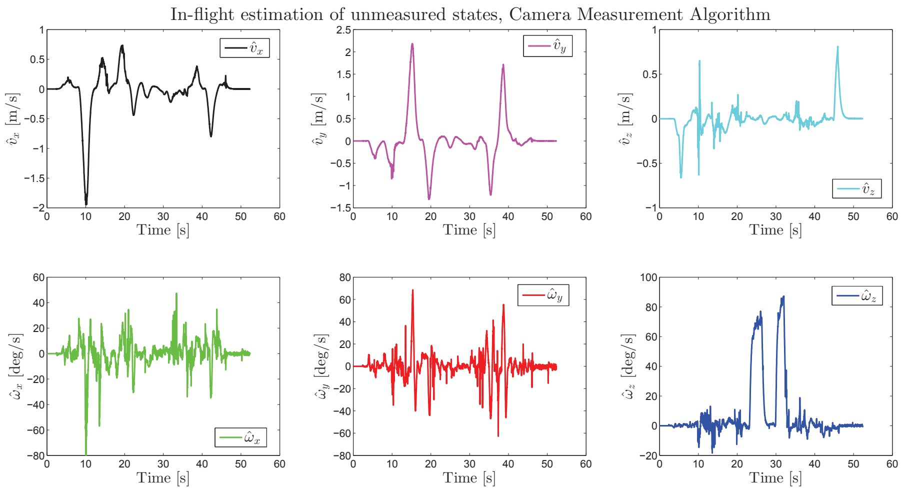

Figure D.5: *In-flight estimation of unmeasured states, Test 1, Camera Measurement Algorithm. The linear velocity estimates (expressed in the BODY frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

D.2.2 State Estimation under Ideal Conditions, Test 2

The second online test of the main functionality of the Camera Measurement Algorithm is presented here. The input parameters presented in Table 7.2 were used, and a realistic scenario in which the UAV was flying around the lab setup while the cameras collected measurements is investigated. The linear velocities are expressed in the (original) BODY as well as the inertial frames in the figures depicted below.

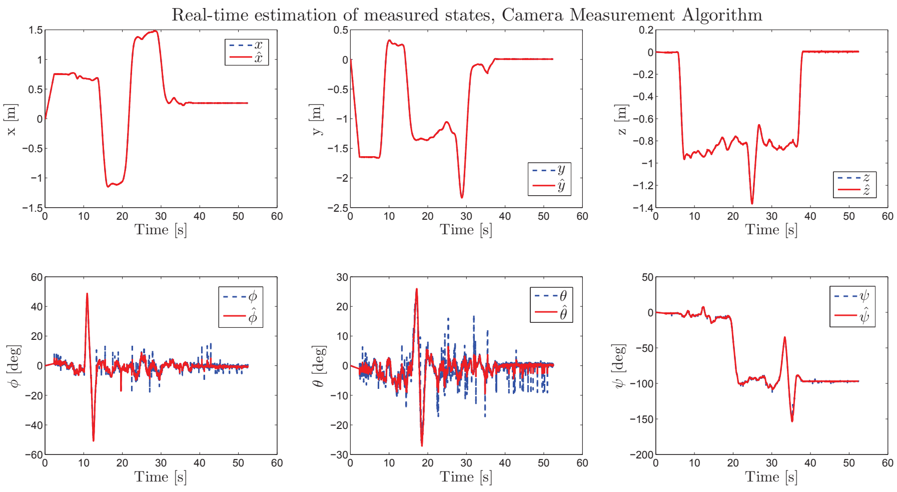

Figure D.6: *In-flight measured state estimation, Test 2, Camera Measurement Algorithm. The blue, dashed lines are the measured values obtained from OptiTrack, while the red lines represent the states estimated using the Camera Measurement Algorithm. The top-most row depicts the position, while the lower row shows the orientation.*

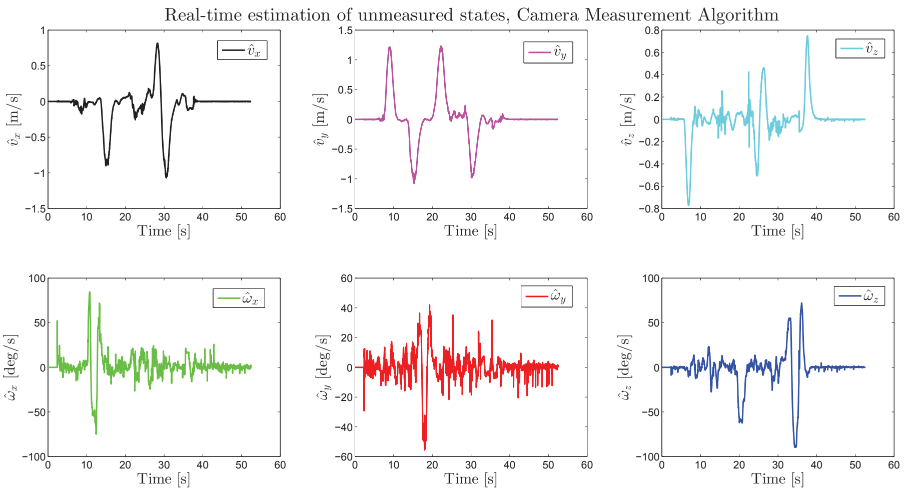

Figure D.7: *In-flight unmeasured state estimation, Test 2, Camera Measurement Algorithm. The linear velocity estimates (expressed in the BODY frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

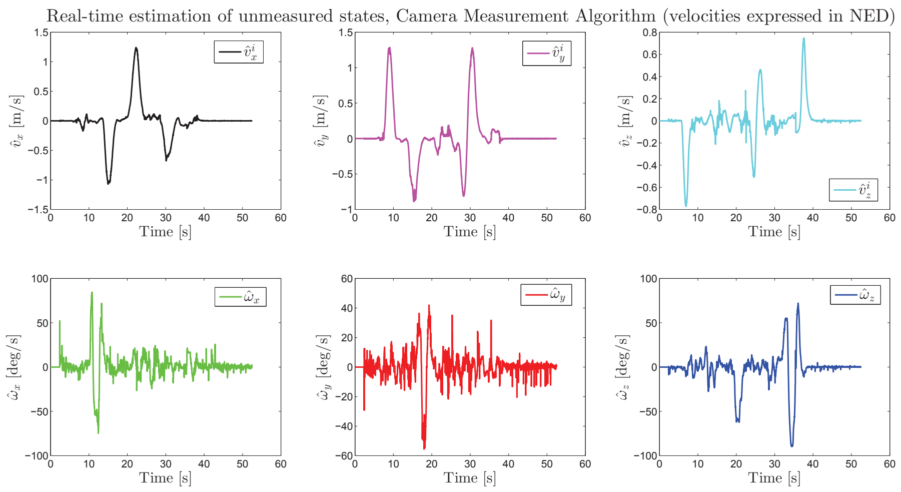

Figure D.8: *In-flight unmeasured state estimation, Test 2, Camera Measurement Algorithm. The linear velocity estimates (expressed in the inertial frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

D.2.3 State Estimation with Measurement Errors

The estimated linear and angular velocities for the scenario with measurement loss in Section 7.2, expressed in the BODY frame, are depicted here.

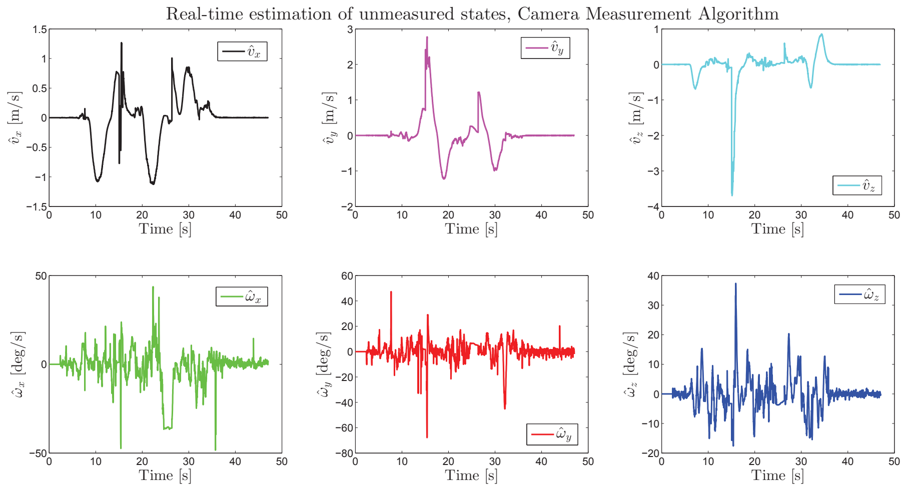

Figure D.9: *Online unmeasured state estimation, several measurement losses, Camera Measurement Algorithm. The linear velocity estimates (expressed in the BODY frame) are shown in the top-most row, while the angular velocity estimates are depicted in the second row.*

Appendix E

Results, Integrated Camera System/INS Algorithm

E.1 Online State Estimation under Ideal Conditions, Test 1

The estimated linear velocities for Test 1 of state estimation under ideal conditions, expressed in the BODY frame, are depicted here.

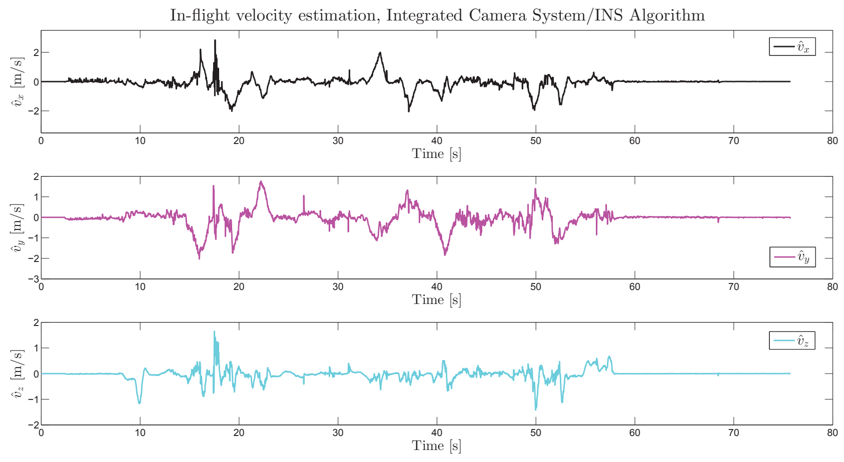

Figure E.1: *In-flight velocity estimation, Test 1, Integrated Camera System/INS Algorithm. The original velocity estimates (expressed in BODY) are shown.*

E.2 Online State Estimation under Ideal Conditions, Test 2

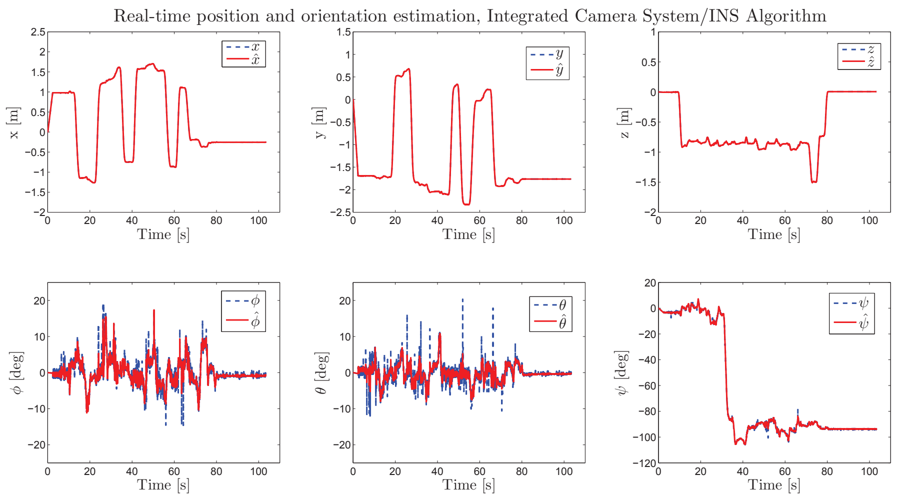

Figure E.2: *In-flight measured state estimation, Test 2, Integrated Camera System/INS Algorithm. The blue, dashed lines are the measurements obtained from OptiTrack, while the red lines represent the estimated states. The top-most row depicts the position, while the lower row shown the orientation.*

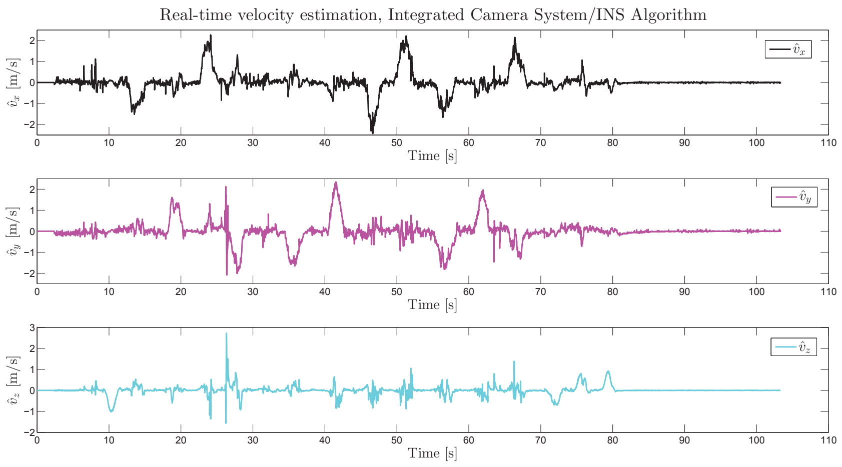

Figure E.3: *In-flight velocity estimation, Test 2, Integrated Camera System/INS Algorithm. The original velocity estimates (expressed in BODY) are shown.*

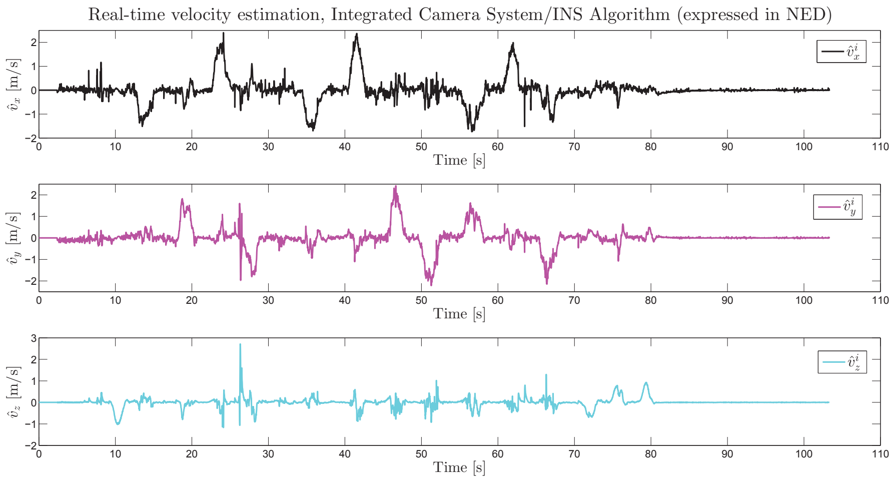

Figure E.4: *In-flight velocity estimation, Test 2, Integrated Camera System/INS Algorithm. The velocity estimates are expressed in the inertial frame.*

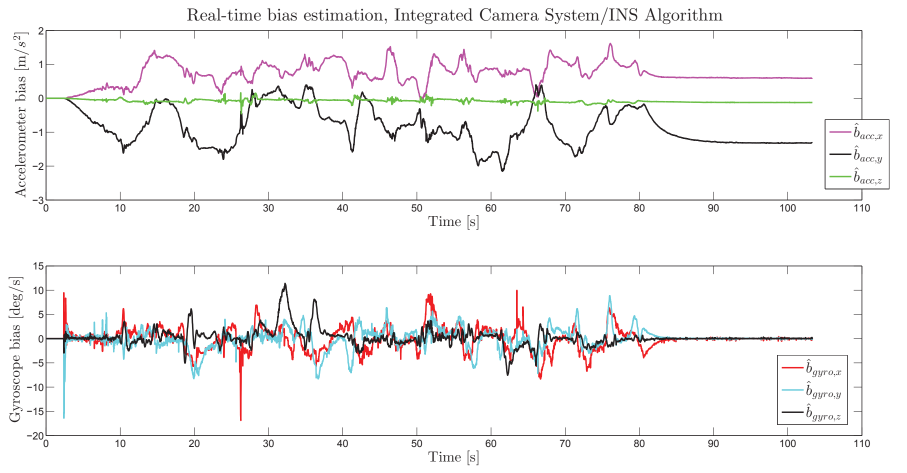

Figure E.5: *In-flight bias estimation, Test 2, Integrated Camera System/INS Algorithm. The top-most row shows the accelerometer bias estimates, while the gyro bias estimates are shown in the lower row.*

E.3 Online State Estimation with Camera Measurement Loss

The estimated linear velocities for the scenario with measurement loss in Section 7.4, expressed in the BODY frame, are depicted here.

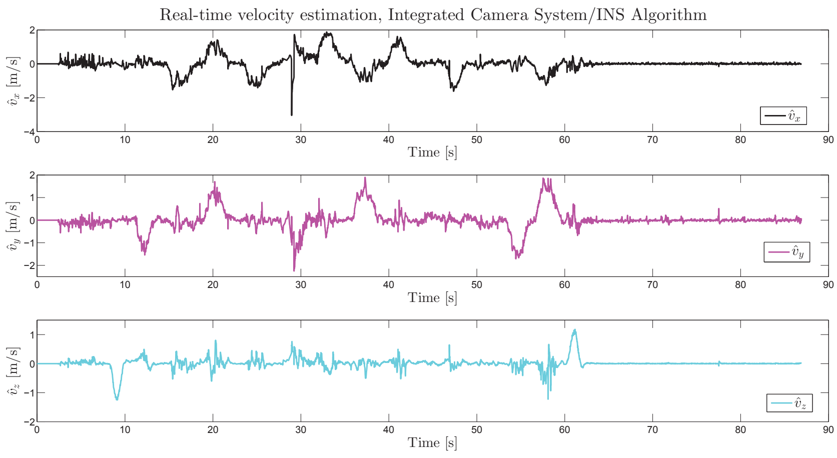

Figure E.6: *Online velocity estimation with measurement loss, Integrated Camera System/INS Algorithm. The linear velocity estimates are expressed in the inertial frame.*