Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Marcus Lundeby Lerfald

Vision-Based Precision Landing of Autonomous Unmanned Aerial Vehicles

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
June 2023

NTNU
Norwegian University of Science and Technology

Aviant

Marcus Lundeby Lerfald

Vision-Based Precision Landing of Autonomous Unmanned Aerial Vehicles

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

The range of potential applications for Unmanned Aerial Vehicles (UAVs) continues to expand due to their enhanced capabilities and decreased costs in recent years. A domain in which they could offer valuable advancements is aerial transportation and logistics. This thesis presents a complete system for performing precision landing of a UAV on a landing platform in the context of autonomous aerial cargo delivery. The proposed method assists the primary autopilot in navigation during the descent phase from an initial position susceptible to noise originating from Global Navigation Satellite Systems (GNSSs). The discrepancy between the global positioning error and the mechanical tolerances of the landing platform is substantial and necessitates corrections. To correct the global positioning error, the designed and implemented system uses visual detections of fiducial markers embedded in the landing platform to estimate the relative position and orientation between the platform and aircraft. The adoption of a recursive arrangement of fiducial markers ensures marker visibility throughout the entire descent. This arrangement effectively maintains horizontal alignment above the center of the landing platform, all while minimizing the required platform footprint.

By employing the onboard camera and Inertial Measurement Unit (IMU), the companion computer of the aircraft provides real-time estimates of the relative pose using an Invariant Extended Kalman Filter (IEKF). The IEKF represents the complete extended pose of the aircraft within a Lie group framework, inherently adhering to geometric constraints associated with attitude representation. This framework guarantees that the filter updates remain confined to the manifold, thereby preserving the integrity of the attitude estimate. For the given system, it is shown that the logarithmic mapping of the estimation error follows a linear differential equation which is independent of the true state of the system, commonly referred to as the log-linear property. This ensures local stability around any trajectory, a rare trait to hold for nonlinear estimators.

The full pose estimate in six degrees of freedom is computed analytically by exploiting geometric properties and the known scale of the fiducial markers in the Infinitesimal Plane-Based Pose Estimation (IPPE) method. The analytical solution to the Perspective-n-Point (PnP) problem is significantly faster than traditional iterative approaches, making it ideal for real-time estimation. The filter parameters in terms of process and measurement noise covariance matrices are tuned using real data obtained from a motion capture environment, and subsequently assessed for consistency and performance. The evaluation metrics are based on Normalized Innovation Squared (NIS), Normalized Estimation Error Squared (NEES) and Root Mean Square Error (RMSE) and are employed on a hold-out dataset.

Finally, a statistical analysis of the full system performance is conducted through experimental trials. The results of this study demonstrate the effectiveness of an inertially-driven IEKF based on pose updates from the IPPE method applied to recursive fiducial markers. The proposed system reliably converges to the center of the landing platform with high precision compared to the platform's tolerances. The IEKF shows potential for further development and application within the domain of inertial navigation.

Sammendrag

De potensielle bruksområdene for ubemannede luftfartøy har i løpet av de siste årene utvidet seg betydelig grunnet deres reduserte kostnad og forbedrede funksjonalitet. Et område hvor de kan tilby verdifulle fremskritt er innen lufttransport og logistikk. I denne masteroppgaven presenteres et komplett system for presisjonslanding av ubemannede luftfartøy på en landingsplattform i kontekst av autonom luftfraktlevering. Systemet gaider den primære autopilotens navigasjon under landingssekvensen fra en initiell posisjon som er utsatt for støy fra satellittbaserte posisjoneringssystemer. Den globale posisjoneringsfeilen er uproporsjonalt høy sammenliknet med plattformens mekaniske toleranser og må derfor korrigeres. For å korrigere den globale posisjoneringsfeilen, baserer presisjonslandingssystemet seg på visuell deteksjon av fiduciale markører fastmontert til landingsplattformen for å estimere relativ posisjon og orientasjon mellom plattformen og luftfartøyet. Bruken av fiduciale markører i en rekursiv ordning sikrer synlighet av markørene gjennom hele nedstigningen. Denne ordningen opprettholder horisontal innretting over midten av landingsplattformen, samtidig som den minimerer nødvendig plassbehov for plattformen.

Ved å benytte seg av et ombordkamera og treghetsmålinger, vil luftfartøyets datamaskin benytte et invariant utvidet Kalman-filter for å gi estimater i sanntid. Det invariante filteret representer posisjon, hastighet og orientasjon for luftfartøyet i et matematisk rammeverk basert på Lie-grupper som har den iboende egenskapen til å overholde geometriske begrensninger knyttet til representasjon av orientasjon. Dette rammeverket garanterer at filterets oppdaterte estimater forblir på manifolden, hvilket ivaretar integriteten til orientasjonsestimatet. Det er vist at logaritmen til estimeringsfeilen for det gitte systemet følger en lineær differensiallikning som er uavhengig av systemets sanne tilstand, ofte referert til som den log-lineære egenskapen. Dette sørger for lokal stabilitet rundt en vilkårlig bane, en sjelden egenskap for ikke-lineære observatører.

Posisjon og orientasjon i seks frihetsgrader er estimert analytisk ved å utnytte geometriske egenskaper og den kjente størrelsen på de fiduciale markørene i Infinitesimal Plane-Based Pose Estimation (IPPE)-metoden. Den analytiske løsningen på Perspective-n-Point (PnP)-problemet er vesentlig raskere enn tradisjonelle iterative tilnærminger, hvilket gjør den ideell for sanntidsestimering. Filterets parametere i form av kovariansmatriser for prosess- og målestøy er funnet ved hjelp av ekte data innhentet med høypresis bevegelsessporing og senere evaluert for kvalitet og ytelse. Den endelige evalueringen er basert på normaliserte innovasjoner og estimeringsfeil kvadrert i tillegg til det kvadratiske gjennomsnittet av estimeringsfeilen og er gjennomført for et separat datasett.

Avslutningsvis er det gjennomført en statistisk analyse basert på eksperimentell data av effektiviteten til systemet som en helhet. Resultatene av dette studiet demonstrerer ytelsen til et treghetsbasert invariant utvidet Kalman-filter basert på orientasjon- og posisjonsoppdateringer. Oppdateringene har opphav i IPPE-metoden anvendt på rekursive fiduciale markører. Systemet konvergerer pålitelig til landingsplattformens senter med høy presisjon sammenliknet med plattformens toleranser. Det invariante utvidede Kalman-filteret viser potensial for videre utvikling og anvendelse som treghetsnavigasjonssystem.

Preface

This thesis concludes my five years at the Norwegian University of Science and Technology in the Department of Engineering Cybernetics to finish a master's degree in Cybernetics and Robotics. The work is conducted in collaboration with Aviant, a company that specializes in the field of autonomous aerial delivery, with a particular emphasis on their existing aircraft platform. The thesis introduces a proposed design and implementation of a vision-based precision landing system using inertial measurements and invariant filtering. This thesis aims to contribute toward the development of a fully autonomous end-to-end aerial delivery service, using existing sensors and hardware to precisely land the vehicle on an automated platform capable of performing cargo loading and battery charging.

First and foremost, I would like to express my gratitude to my thesis supervisor, Prof. Dr. Kostas Alexis, for his invaluable guidance throughout the completion of this research. In addition, I would like to thank the Aviant team for their crucial insights into the company's platform and lending me the hardware required for testing the proposed system. I would also like to extend my appreciation to Mohit Singh and Morten Nissov from the Autonomous Robots Lab for providing me access to calibration targets, a motion capture system and engaging in technical discussions.

The precision landing system proposed in this thesis was developed in Ubuntu 20.04 and ROS Noetic. In addition to the aforementioned, this thesis has benefited significantly from existing open-source libraries and systems. The most notable contributions forming the basis for this work are the *PX4* [1] flight control software, the *MAVLink* [2] messaging protocol, the *OpenCV* [3] computer vision library and the *AprilTag 3* [4] flexible layout for fiducial tags. I am deeply grateful to the developers of the open-source software utilized in this thesis, whose helpful contributions have significantly enhanced the progress and results of this research.

Marcus Lundeby Lerfald

Trondheim, 5 June 2023

Contents

List of Tables

List of Figures

Acronyms

AGL Above Ground Level
ANEES Average Normalized Estimation Error Squared
ANIS Average Normalized Innovation Squared
API Application Programming Interface

BMS Battery Management System
BVLOS Beyond Visual Line Of Sight

CAA Civil Aviation Authority
CDF Cumulative Distribution Function
CEP Circular Error Probability
CI Confidence Interval
CPU Central Processing Unit
CRLB Cramér–Rao Lower Bound

DOF Degrees Of Freedom

EKF Extended Kalman Filter

FOV Field Of View
FPS Frames Per Second

GCS Ground Control Station
GNSS Global Navigation Satellite System
GPIO General-Purpose Input/Output

IAC Image Of The Absolute Conic
IEKF Invariant Extended Kalman Filter
IMU Inertial Measurement Unit
IPPE Infinitesimal Plane-Based Pose Estimation
ISP Image Signal Processor

MAV Micro Air Vehicle
MEKF Multiplicative Extended Kalman Filter
MEMS Microelectromechanical Systems
MTOM Maximum Take-Off Mass

NEES Normalized Estimation Error Squared
NIS Normalized Innovation Squared

PDF Probability Density Function
PnP Perspective-n-Point
PPF Percent Point Function

RAM Random Access Memory
RGB Red Green Blue
RMSE Root Mean Square Error
ROS Robot Operating System
RTK Real-Time Kinematic

SBC Single-Board Computer
SITL Software In The Loop
SLAM Simultaneous Localization And Mapping

TCP Transmission Control Protocol

UART Universal Asynchronous Receiver-Transmitter
UAV Unmanned Aerial Vehicle
UDP User Datagram Protocol
USB Universal Serial Bus
VIO Visual-Inertial Odometry
VTOL Vertical Take-Off And Landing

Chapter 1

Introduction

1.1 Motivation

In the vast landscape of modern robotics, the emergence of Unmanned Aerial Vehicles (UAVs) has ignited a paradigm shift that extends beyond the constraints of terrestrial navigation. As aerial counterparts to their ground-based robotic companions, UAVs have opened up new dimensions for scientific research, industrial applications, and societal advancements. By combining robotics and aviation, UAVs have not only enhanced existing applications but have also unlocked entirely new avenues for exploration and problem-solving. The integration of sophisticated sensors, machine learning algorithms, and precise control systems has enabled UAVs to autonomously navigate complex environments, adapt to changing conditions, and perform intricate tasks with remarkable precision. Over the last two decades, there has been an exponential growth in UAV technology and its integration into everyday life. The miniaturization of sensors, increase in computational power, reduced cost of hardware, advancements in materials, and the widespread adoption of wireless communication systems has accelerated the development of smaller, more agile UAVs. This progress opened up entirely new domains of application, such as search and rescue operations, infrastructure inspections, and delivery services.

Over the past years, Vertical Take-Off And Landing (VTOL) hybrid designs have become increasingly popular. VTOL UAVs combine conventional multirotor and fixed-wing aircraft design in a single, more capable platform. The multirotor capabilities of a VTOL UAV allow it to hover in place or takeoff and land without the need for traditional infrastructure such as runways, launchers or nets. By transitioning to fixed-wing flight, a VTOL UAV can also enjoy the increased energy efficiency and range of conventional winged aircrafts. These properties make VTOL UAVs common for long range aerial delivery services, utilizing the hover capabilities to winch down payloads in cluttered environments from a safe distance. One of the VTOL UAVs used by Aviant is shown in Figure 1.1.

(a) In fixed-wing flight, efficiency is attained through the utilization of forward airspeed and wings to generate lift.

(b) In multirotor flight, the aircraft possesses the capability to achieve vertical takeoff and landing, as well as hover in a stationary position.

Figure 1.1: An instance from Aviant's fleet of VTOL UAVs which is the focus in this thesis. *Photographer: Author.*

The level of autonomy in typical UAV applications range from fully manual remote piloted vehicles, to assisted or even fully autonomous systems. Robotics research continually strives toward achieving higher levels of autonomy and minimizing human intervention, aiming to liberate our time and resources to be applied elsewhere. In the context of autonomous aerial delivery, the need for human intervention to recharge batteries and load new payloads poses a constraint on scalability, thereby impeding the realization of a fully autonomous solution. To reduce the downtime and human intervention in an autonomous delivery service, a landing platform can be designed to perform the required interactions with the UAV. The mechanical tolerances of the landing platform within which the UAV must land are typically much lower than the accuracy of the Global Navigation Satellite System (GNSS) modules used by most aerial delivery UAVs. This motivates the pursuit for a precision landing system, which ensures that the UAV reliably lands within the tolerances of the landing platform.

Aviant has purposefully designed and constructed a prototype landing platform tailored to cater to their fleet of aerial delivery vehicles, taking into consideration this specific use case. The landing platform uses electromechanical actuators to align the UAV, connect to the battery charging interface and load new payloads. The operational vision entails the utilization of multiple landing platforms in daily activities, including the deployment of certain platforms in the field, thereby expanding the delivery range of the system. In order to maintain scalability and preserve the end-to-end nature of the solution while enabling mobility, the landing platform's overall size and footprint must be physically manageable. The platform's footprint is constrained to that of a trailer, not exceeding the dimensions of a standard parking space.

The desired mechanical tolerances of the platform will directly impact several key factors, including its footprint, the dimensions and strength of the actuators, as well as consid-

erations related to weight and cost. Achieving consistent and precise landing with high accuracy is sought after as it enables the reduction of the landing platform's mechanical tolerances and the benefits it entails.

1.2 Problem Formulation and Contributions

As a contribution towards the goal of a fully autonomous end-to-end delivery service, the objective of this thesis is to design, implement and test a precision landing system capable of aiding a UAV's autopilot in reliably descending onto a landing platform within its mechanical tolerances. The system must detect the landing platform from a safe distance and provide accurate real-time estimates of the relative position and orientation between the UAV and the platform in order to provide correctional setpoints for the autopilot during the descent.

In order to remain scalable, the precision landing system should be based on affordable and easily available sensors which are present by default on typical delivery UAVs, such as the autopilot's Inertial Measurement Unit (IMU) and the onboard camera which is enforced by the Norwegian Civil Aviation Authority (CAA) for Beyond Visual Line Of Sight (BVLOS) operations. It is desirable to keep the computational cost low, ensuring minimal constraints on the onboard companion computer.

This thesis focuses on using fiducial markers in a recursive layout in combination with an Invariant Extended Kalman Filter (IEKF) to detect and estimate the relative pose of the landing platform in real time. The estimate is then used for guiding the UAV's autopilot in a decent. The research question on which this thesis is based is the following:

- Is a framework based on pose measurements of recursive visual fiducial markers and an IEKF driven by inertial data a viable approach to reliably guide a UAV from an initial, known position above the ground subject to GNSS noise to a state in which the UAV has touched down and aligned itself according to the mechanical tolerances of its landing platform?

Expanding on the recent research of Lie groups in invariant filtering and the works of [5], [6], [7] and [8] in addition to flexible and efficient fiducial marker design and presented in [4], the main goal of this thesis is to design, implement and experimentally demonstrate the viability of a precision landing system that can be integrated with a broad range of UAV platforms. With the exception of the fiducial markers, the system should utilize sensors and hardware already present on the UAV and an efficient implementation that imposes low computational demands on the companion computer.

Aligned with the primary objective, this thesis will also pursue the following intermediate goals:

- Based on theoretical and experimental results, determine if the use of a recursive fiducial layout and an IEKF can provide some notion of guarantees in terms of convergence, safety and reliability of the system as a whole.
- Determine the consistency of an IEKF used to estimate the extended pose of the

UAV based on IMU data and full six Degrees Of Freedom (DOF) pose measurements.

- In pursuit of reducing the footprint of the landing platform, determine bounds for landing precision and accuracy such that future revisions of the landing platform can decrease its mechanical tolerances.

This thesis makes the following key contributions related to the research question:

- An implementation of the IEKF capable of estimating the orientation, position and velocity of a UAV in addition to IMU biases.
- A tool for logging, tuning and evaluating the filter consistency.
- A proposed design and implementation for a precision landing framework based on visual fiducial marker detections and inertial measurements which can be used to guide the primary autopilot during a decent.
- A physical realization of the fiducial marker layout used as a landing target, which can be embedded on the landing platform.
- Statistical analysis of the integrated system's performance in terms of accuracy and precision, based on real life experiments.
- An interface for autopilot communication based on industry standards, which can be integrated with existing and future compliant autopilots.

1.3 Delimitations, Limitations and Assumptions

Intended to focus the scope of this thesis and facilitate relevant development and progression pertaining to the research questions, the following key delimitations are established:

- The landing platform is subject to ownership and design freedom, including fiducial marker arrangement.
- The detection system must be capable of operating under low-light conditions, although partial illumination can be provided on the landing platform to mitigate absolute darkness.
- The production version of the landing platform will incorporate heating or actuated roofing mechanisms that selectively open during the landing sequence to prevent occlusion caused by the accumulation of snow or other debris.
- Prior information regarding the approximate location of the landing platform is available up to noise inherent to GNSS measurements.
- The landing platform possesses the capability to accurately align the position and heading of the UAV upon landing, provided that it lands within the mechanical tolerances of the platform.

To acknowledge the constraints and potential challenges inherent in this thesis, the following key limitation is recognized:

- In capturing the ground truth of the UAV's pose, the motion capture system is confined to a height limitation lower than the intended altitude encountered during a landing sequence.

In order to establish the fundamental framework for the research conducted in this thesis, specific key assumptions have been defined:

- Within the flight volume encompassing the potential initial positions of the UAV down to the landing platform, it is assumed that there are no obstacles present, ensuring unimpeded navigation.
- During the landing process, the GNSS-based position estimate of the autopilot is anticipated to experience minimal drift, enabling it to maintain position with accuracy, albeit with the possibility of an offset in its estimate relative to the true position.

1.4 Thesis Outline

This thesis is composed of a total of six chapters. Chapter 1 serves as an introduction, outlining the motivation behind the thesis, elucidating the key research question, and emphasizing the contributions made by the thesis. Additionally, it provides an overview of the key delimitations, limitations, and assumptions, which are presented to facilitate the reader's understanding and provide contextual clarity. Chapter 2 comprehensively examines the pertinent theories that underpin all facets of the proposed precision landing system. It also introduces previously conducted research related to the suggested system modules, its auxiliary components and methodology used in this thesis. Chapter 3 presents the framework of the precision landing system, encompassing the entirety of its design and functionality. In contrast, Chapter 4 delves into the technical intricacies and implementation details of pivotal procedures and novel algorithms associated with the system. Chapter 5 is dedicated to the presentation of the testing results, accompanied by a comprehensive evaluation and in-depth discussion. Subsequently, Chapter 6 serves as the culmination of the research effort in this thesis, offering concluding remarks and recommending future avenues of exploration to address the limitations observed in the proposed and implemented precision landing system.

Chapter 2

Background

2.1 Lie Groups

The intention of this section is to provide definitions and basic properties of Lie groups. Lie theory is a vast mathematical field and cannot be fully covered in this contribution alone. The approach is motivated by the problems encountered in the field of robotics and state estimation, choosing a selected subset of the topics from Lie theory. Categorically, this section focuses on groups related to rigid transformation and rotation matrices in three dimensions. According to [5], a Lie group unifies the mathematical concepts of a group and a smooth manifold. The group, denoted $(\mathcal{G}, \circ)$, is defined as a set $\mathcal{G}$ on which the composition operator $\circ$ satisfies the following axioms for the elements $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{G}$ and the identity $\mathcal{E} \in \mathcal{G}$:

1. Closure: $\mathcal{A} \circ \mathcal{B} \in \mathcal{G}$
2. Identity: $\mathcal{E} \circ \mathcal{A} = \mathcal{A} \circ \mathcal{E} = \mathcal{A}$
3. Inverse: $\mathcal{A}^{-1} \circ \mathcal{A} = \mathcal{A} \circ \mathcal{A}^{-1} = \mathcal{E}$
4. Associativity: $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C} = \mathcal{A} \circ (\mathcal{B} \circ \mathcal{C})$

The smooth manifold represents a topological space which is differentiable, implying a uniquely defined tangent space to any point on the manifold. In Lie theory applied to robotics, we can say that our state vector evolves on the surface of the manifold and the linear tangent space can be used to perform calculus. This representation allows constraints imposed on the state vector to be encoded in the manifold. Due to the axiom of closure, we can therefore guarantee that said constraints are satisfied under the composition operator. This is especially useful when representing orientation, which is discussed in Section 2.2.3.

This thesis only considers the matrix Lie groups. A matrix Lie group is a subgroup of

the set of square, invertible $n \times n$ matrices with real entries, $\mathcal{GL}(n, \mathbb{R})$, which follows the axioms above. For matrix Lie groups, the axioms are satisfied by the matrix product, the identity matrix and the matrix inverse.

According to Section 0.2 of [9], any topological property valid at a single point of the manifold can be applied to any other point. This means that all tangent spaces on the manifold are alike, and we can conveniently use the tangent space at the identity to perform calculus. The Lie algebra $\mathfrak{g}$ of a matrix Lie group $\mathcal{G}$ is defined as the tangent space at the identity, $T_\mathcal{E}\mathcal{G}$. $\mathfrak{g}$ is a vector space associated with $\mathcal{G}$, which is a a subspace of all $n \times n$ matrices with real entries. Specifically, the Lie algebra can be related to $\mathbb{R}^m$ due to it being a vector space, where m is the number of DOF for the Lie group [5]. The Lie algebra allows us to join local properties of a smooth, differentiable manifold with global properties such as the composition of distant objects which may be nonlinear. The relation is composed of two isomorphisms, commonly referred to as *hat* and *vee*:

$$\text{Hat}: \quad \mathbb{R}^m \to \mathfrak{g} \quad ; \quad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge \tag{2.1a}$$

$$\text{Vee}: \quad \mathfrak{g} \to \mathbb{R}^m \quad ; \quad \boldsymbol{\tau}^\wedge \mapsto (\boldsymbol{\tau}^\wedge)^\vee = \boldsymbol{\tau} \tag{2.1b}$$

In general, the elements of the Lie algebra can be cumbersome to work with directly as they might be represented with imaginary numbers, quaternions or skew-symmetric matrices. The linear, invertible mapping provides a convenient representation of the Lie algebra in $\mathbb{R}^m$ which can be manipulated with linear algebra. It is common to denote elements $\boldsymbol{\tau}$ in the tangent plane as $\boldsymbol{\tau}^\wedge$ when expressed in the Lie algebra and omitting the hat decorator when $\boldsymbol{\tau}$ is represented as a vector in $\mathbb{R}^m$. A left superscript can be used to explicitly state the exact point in which the tangent space is defined [5]. For example, ${}^\mathcal{X}\boldsymbol{\tau}^\wedge \in T_\mathcal{X}\mathcal{G}$ means the $\boldsymbol{\tau}$ element in the tangent space at $\mathcal{X} \in \mathcal{G}$. The left superscript is commonly omitted for the identity $\mathcal{E}$, i.e. the Lie algebra.

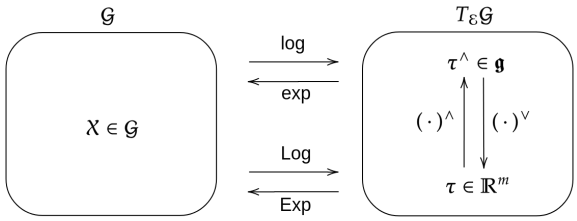


Figure 2.1: Visual representation of the mapping between a Lie group $\mathcal{G}$, its Lie algebra $\mathfrak{g} = T_\mathcal{E}\mathcal{G}$ and the isomorphisms relating the elements of the Lie algebra to their respective Cartesian vector space. Adapted from [5].

The Lie algebra $\mathfrak{g}$ and its Cartesian vector representation can be exactly mapped into the elements of the Lie group $\mathcal{G}$ and back using the exponential map and its inverse, the logarithmic map:

$$\exp: \quad \mathfrak{g} \to \mathcal{G} \quad ; \quad \boldsymbol{\tau}^\wedge \mapsto \mathcal{X} = \exp(\boldsymbol{\tau}^\wedge) \tag{2.2a}$$

$$\log: \quad \mathcal{G} \to \mathfrak{g} \quad ; \quad \mathcal{X} \mapsto \boldsymbol{\tau}^\wedge = \log(\mathcal{X}) \tag{2.2b}$$

For convenience, the capitalized exponential mapping can be defined similarly,

$$\text{Exp}: \quad \mathbb{R}^m \to \mathcal{G} \quad ; \quad \boldsymbol{\tau} \mapsto \mathcal{X} = \text{Exp}(\boldsymbol{\tau}) \tag{2.3a}$$

$$\text{Log}: \quad \mathcal{G} \to \mathbb{R}^m \quad ; \quad \mathcal{X} \mapsto \boldsymbol{\tau} = \text{Log}(\mathcal{X}) \tag{2.3b}$$

Equation 2.3a and 2.3b provide a direct mapping from the from the Cartesian vector space representation to the Lie group. Figure 2.1 provides a visual overview of the mappings between the Lie group, its tangent space and the isomorphism between the Lie algebra and Cartesian vector space representation. A visual representation of the relation between a Lie group's manifold $\mathcal{G}$ and the Lie algebra $T_\mathcal{E}\mathcal{G}$ is shown in Figure 2.2.

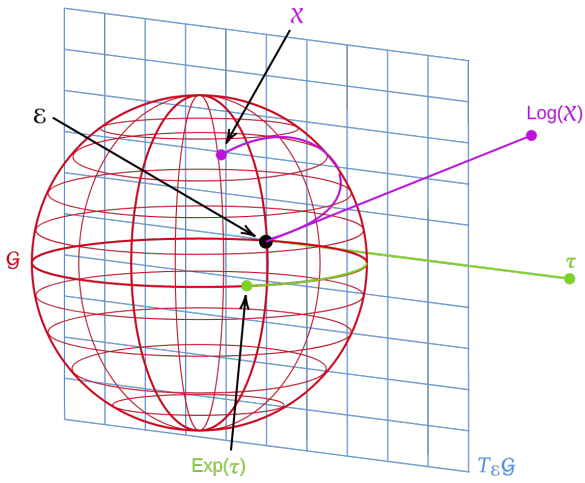


Figure 2.2: Visual representation of how elements in the Lie group's manifold $\mathcal{G}$ relate to the corresponding element in the Lie algebra $T_\mathcal{E}\mathcal{G}$, which is the tangent space to the manifold at the identity. Similarly to how a piece of string can be wrapped around a ball's geodesic[1], all elements in the Lie algebra have an corresponding representation in the Lie group. For instance, an element $\boldsymbol{\tau}$ in the Lie algebra has the equivalent element $\text{Exp}(\boldsymbol{\tau})$ in the Lie group. Inversely, an element $\mathcal{X}$ on the group has the Lie algebra equivalent element $\text{Log}(\mathcal{X})$. Adapted from [5].

It is desirable to be able to express small increments of a manifold in the local tangential vector space. With this in mind, the plus and minus operators are defined by combining a composition with the exponential map or its inverse. Recall that commutativity is not guaranteed for the composition according to the Lie group axioms. For this reason, each operator has two definition; a left and a right formulation:

$$\text{Right-}\oplus: \quad \mathcal{Y} = \mathcal{X} \oplus {}^{\mathcal{X}}\boldsymbol{\tau} = \mathcal{X} \circ \text{Exp}({}^{\mathcal{X}}\boldsymbol{\tau}) \in \mathcal{G} \tag{2.4a}$$

$$\text{Right-}\ominus: \quad {}^{\mathcal{X}}\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} = \text{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in T_\mathcal{X}\mathcal{G} \tag{2.4b}$$

[1] A sphere in $\mathbb{R}^3$ is not a Lie group, but it is easy to visualize and helps build intuition.

Note that this thesis uses the same notation as [5], i.e. ${}^{\mathcal{X}}\boldsymbol{\tau}$ at the right hand side of the composition in Equation 2.4a is said to be expressed in the local frame at $\mathcal{X}$.

$$\text{Left-}\oplus: \quad \mathcal{Y} = {}^{\mathcal{E}}\boldsymbol{\tau} \oplus \mathcal{X} = \text{Exp}({}^{\mathcal{E}}\boldsymbol{\tau}) \circ \mathcal{X} \in \mathcal{G} \tag{2.5a}$$

$$\text{Left-}\ominus: \quad {}^{\mathcal{E}}\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} = \text{Log}(\mathcal{Y} \ominus \mathcal{X}^{-1}) \in T_{\mathcal{E}}\mathcal{G} \tag{2.5b}$$

Similarly, we now define ${}^{\mathcal{E}}\boldsymbol{\tau}$ on the left hand side of Equation 2.5 to be expressed in the global frame. Since the $\ominus$ operator is ambiguous, this thesis uses the right formulation by default. From Equations 2.4 and 2.5 a relation between the local and global elements is observed. The adjoint operator is used to exactly transform tangent vectors between tangent spaces by a linear operation:

$$\text{Ad}_{\mathcal{X}}: \quad \mathfrak{g} \to \mathfrak{g} \quad ; \quad \boldsymbol{\tau}^{\wedge} \mapsto \text{Ad}_{\mathcal{X}}(\boldsymbol{\tau}^{\wedge}) = \mathcal{X}(\boldsymbol{\tau}^{\wedge})\mathcal{X}^{-1} \tag{2.6}$$

The adjoint is commonly used to transform vectors in the tangent space at $\mathcal{X} \in \mathcal{G}$, $T_{\mathcal{X}}\mathcal{G}$ to the Lie algebra, ${}^{\mathcal{E}}\boldsymbol{\tau}^{\wedge} = \text{Ad}_{\mathcal{X}}({}^{\mathcal{X}}\boldsymbol{\tau}^{\wedge})$. Since the transformation is linear, it can also be performed with a matrix operation. The adjoint matrix can be calculated by applying the vee operator to Equation 2.6,

$$\mathbf{Ad}_{\mathcal{X}}\boldsymbol{\tau} = (\mathcal{X}(\boldsymbol{\tau}^{\wedge})\mathcal{X}^{-1})^{\vee} \tag{2.7}$$

An important characteristic of the adjoint matrix is its property of $\mathbf{Ad}_{\mathcal{X}}^{-1} = \mathbf{Ad}_{\mathcal{X}^{-1}}$ due to the lower computational cost associated with the right-hand side compared to the left-hand side [5].

2.1.1 The Special Orthogonal Group, $SO(3)$

One of the most common Lie groups in robotics is $SO(3)$, the group of 3D rotations. $SO(3)$ is used to represent the orientation of a rigid body with a 3×3 rotation matrix $\mathbf{R}$, subject to the following constraints:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} \quad | \quad \mathbf{R}^{\top}\mathbf{R} = \mathbf{I}, \quad \det \mathbf{R} = 1\} \tag{2.8}$$

$SO(3)$ is not commutative and and the tangent space around the identity is found from the orthogonality condition, $\mathbf{R}^{\top}\mathbf{R} = \mathbf{I}$, by differentiating both sides with respect to time:

$$\mathbf{R}^{\top}\dot{\mathbf{R}} + \dot{\mathbf{R}}^{\top}\mathbf{R} = \mathbf{0} \implies \mathbf{R}^{\top}\dot{\mathbf{R}} = -\dot{\mathbf{R}}^{\top}\mathbf{R} = -(\mathbf{R}^{\top}\dot{\mathbf{R}})^{\top} \tag{2.9}$$

This means that $\mathbf{R}^{\top}\dot{\mathbf{R}}$ is skew-symmetric. At the identity, $\mathbf{R} = \mathbf{I}$ and we get

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]_{\times} \tag{2.10}$$

where $[\mathbf{a}]_{\times}$ is used to obtain the cross-product matrix,

$$[\mathbf{a}]_{\times} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{2.11}$$

The Lie algebra of $SO(3)$ in which $[\boldsymbol{\omega}]_\times$ lies is called $\mathfrak{so}(3)$ and is the space of skew-symmetric matrices in 3D. Since the skew-symmetric matrices are isomorphic to $\mathbb{R}^3$, $\boldsymbol{\omega}^\wedge = [\boldsymbol{\omega}]_\times$ and $([\boldsymbol{\omega}]_\times)^\vee = \boldsymbol{\omega}$.

Now, consider the angle-axis rotation $\mathbf{u}\theta = \boldsymbol{\omega} t \in \mathbb{R}^3$ which is the rotation about a vector $\mathbf{u}$ of unit length with angle θ obtained by a constant $\boldsymbol{\omega}$ over a time t. The exponential map is given by the Rodrigues rotation formula,

$$\exp([\mathbf{u}\theta]_\times) = \mathbf{I} + \sin\theta[\mathbf{u}]_\times + (1-\cos\theta)[\mathbf{u}]_\times^2 \tag{2.12}$$

as shown in Example 4 of [5].The capitalized exponential map is $\mathrm{Exp}(\mathbf{u}\theta) = \exp([\mathbf{u}\theta]_\times)$ as defined in Equation 2.3a. The logarithmic map is given by

$$\theta = \arccos\left(\frac{\operatorname{Tr}\mathbf{R} - 1}{2}\right) \tag{2.13a}$$

$$\log\mathbf{R} = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\top) \tag{2.13b}$$

with the capitalized logarithmic map defined as in Equation 2.3b. The adjoint matrix is simply $\mathbf{Ad}_\mathbf{R} = \mathbf{R}$.

The left and right Jacobians of $SO(3)$ are given in Appendix A.1.

2.1.2 The Special Euclidean Group, $SE(3)$

The direct spatial isometries denoted $SE(3)$ are useful for representing orientation-position pairs in three dimensions and can be considered an expansion of $SO(3)$ for rigid motion transforms. Such pairs are often referred to as poses and can be embedded in a single matrix,

$$SE(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \quad | \quad (\mathbf{R},\mathbf{T}) \in SO(3)\times\mathbb{R}^3 \right\} \tag{2.14}$$

Here, $\mathbf{T} \in \mathbb{R}^3$ represents a translation vector and $\mathbf{R} \in SO(3)$ is a 3D rotation matrix as defined in Equation 2.8. The bottom row of the matrix in Equation 2.14 makes composition straightforward as it can be performed with the matrix product,

$$\begin{bmatrix} \mathbf{R}_a & \mathbf{T}_a \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_b & \mathbf{T}_b \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b & \mathbf{R}_a\mathbf{T}_b + \mathbf{T}_a \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{2.15}$$

The inverse is then

$$\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{2.16}$$

because

$$\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{R}^\top & -\mathbf{R}\mathbf{R}^\top\mathbf{T} + \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \mathbf{I} \tag{2.17}$$

The Lie algebra, $\mathfrak{se}(3)$, is formed from the tangent space around the identity and is of the type

$$\boldsymbol{\tau}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathfrak{se}(3) \quad | \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^6 \tag{2.18}$$

in which $\boldsymbol{\theta} = \mathbf{u}\theta$ as defined in Equation 2.12. The capitalized exponential and logarithmic maps are given as

$$\mathbf{M} = \mathrm{Exp}(\boldsymbol{\tau}) = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\theta}) & \mathbf{V}(\boldsymbol{\theta})\boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{2.19a}$$

$$\boldsymbol{\tau} = \mathrm{Log}(\mathbf{M}) = \begin{bmatrix} \mathbf{V}^{-1}(\boldsymbol{\theta})\mathbf{T} \\ \mathrm{Log}(\mathbf{R}) \end{bmatrix} \tag{2.19b}$$

where $\mathbf{V}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2$. The derivation is not recited in this thesis, but the reader is referred to Appendix D of [5] for more details. In short, the expression is obtained by expanding the power series of the exponential, grouping the odd and even factors and identifying the coefficients from common Taylor expansions.

Consider the group element $\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix}$, its Lie algebra $\boldsymbol{\tau}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix}$ and vector $\begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^6$. Inspired by Example 6 in [5], we can expand Equation 2.6 for the adjoint matrix,

$$\begin{aligned} \mathbf{Ad}_\mathbf{M}\boldsymbol{\tau} = (\mathbf{M}\boldsymbol{\tau}^\wedge\mathbf{M}^{-1})^\vee &= \begin{bmatrix} \mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{R}^\top & -\mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{R}^\top\mathbf{T} + \mathbf{R}\boldsymbol{\rho} \\ \mathbf{0}^\top & \mathbf{0}^\top \end{bmatrix}^\vee \\ &= \begin{bmatrix} [\mathbf{R}\boldsymbol{\theta}]_\times & -[\mathbf{R}\boldsymbol{\theta}]_\times\mathbf{T} + \mathbf{R}\boldsymbol{\rho} \\ \mathbf{0}^\top & \mathbf{0}^\top \end{bmatrix}^\vee = \begin{bmatrix} [\mathbf{R}\boldsymbol{\theta}]_\times & [\mathbf{T}]_\times\mathbf{R}\boldsymbol{\theta} + \mathbf{R}\boldsymbol{\rho} \\ \mathbf{0}^\top & \mathbf{0}^\top \end{bmatrix}^\vee \\ &= \begin{bmatrix} [\mathbf{T}]_\times\mathbf{R}\boldsymbol{\theta} + \mathbf{R}\boldsymbol{\rho} \\ \mathbf{R}\boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & [\mathbf{T}]_\times\mathbf{R} \\ \mathbf{0}_{3\times3} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \end{aligned} \tag{2.20}$$

which in turns implies that the adjoint matrix is given by

$$\mathbf{Ad}_\mathbf{M} = \begin{bmatrix} \mathbf{R} & [\mathbf{T}]_\times\mathbf{R} \\ \mathbf{0}_{3\times3} & \mathbf{R} \end{bmatrix} \tag{2.21}$$

where the properties $[\mathbf{Ra}]_\times = \mathbf{R}[\mathbf{a}]_\times\mathbf{R}^\top$ and $[\mathbf{b}]_\times\mathbf{c} = -[\mathbf{c}]_\times\mathbf{b}$ were used to obtain the second and third matrix expansions of Equation 2.20.

The left and right Jacobians of $SE(3)$ are given in Appendix A.2.

2.1.3 The Group of Double Direct Spatial Isometries, $SE_2(3)$

Similarly to how $SE(3)$ can be considered an expansion of $SO(3)$, the group of double direct spatial isometries can be considered an expansion of $SE(3)$. $SE_2(3)$ can be used to describe extended poses, which include a velocity vector component in addition to position and orientation [6]. Similarly to $SE(3)$, this triplet can conveniently be embedded in a single matrix,

$$SE_2(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{v} & \mathbf{T} \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \in \mathbb{R}^{5\times5} \quad | \quad (\mathbf{R}, \mathbf{v}, \mathbf{T}) \in SO(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \right\} \tag{2.22}$$

The composition of two extended poses follows a similar structure as the $SE(3)$ case and is performed by matrix multiplication,

$$\begin{bmatrix} \mathbf{R}_a & \mathbf{v}_a & \mathbf{T}_a \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_b & \mathbf{v}_b & \mathbf{T}_b \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b & \mathbf{R}_a\mathbf{v}_b + \mathbf{v}_a & \mathbf{R}_a\mathbf{T}_b + \mathbf{T}_a \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \tag{2.23}$$

According to [6], the Lie algebra $\mathfrak{se}_2(3)$ is defined similarly to $\mathfrak{se}(3)$ and is of the type,

$$\boldsymbol{\tau}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\nu} & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \in \mathfrak{se}_2(3) \quad | \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\nu} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^9 \tag{2.24}$$

The capitalized exponential and logarithmic maps from $SE(3)$ as defined in Equations 2.19 can be directly expanded to $SE_2(3)$, in which the position and velocity components obey the same structure. According to [6], the respective mappings are defined as

$$\mathbf{M} = \mathrm{Exp}(\boldsymbol{\tau}) = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\theta}) & \mathbf{V}(\boldsymbol{\theta})\boldsymbol{\nu} & \mathbf{V}(\boldsymbol{\theta})\boldsymbol{\rho} \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \tag{2.25a}$$

$$\boldsymbol{\tau} = \mathrm{Log}(\mathbf{M}) = \begin{bmatrix} \mathbf{V}^{-1}(\boldsymbol{\theta})\mathbf{T} \\ \mathbf{V}^{-1}(\boldsymbol{\theta})\mathbf{v} \\ \mathrm{Log}(\mathbf{R}) \end{bmatrix} \tag{2.25b}$$

The adjoint matrix follows as similar pattern. Using the same derivation as in Equation 2.20 expanded to $SE_2(3)$, one arrives at

$$\mathbf{Ad}_\mathbf{M} = \begin{bmatrix} \mathbf{R} & [\mathbf{v}]_\times\mathbf{R} & [\mathbf{T}]_\times\mathbf{R} \\ \mathbf{0}_{3\times3} & \mathbf{R} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{R} \end{bmatrix} \tag{2.26}$$

The left and right Jacobians of $SE_2(3)$ are given in Appendix A.3.

2.2 The Kalman Filter

In this section, the linear Gaussian Kalman filter and its default extension to nonlinear systems are presented as contextual knowledge for the reader, albeit with the assumption of prior familiarity. Their properties are used to motivate a modification of the Extended Kalman Filter (EKF) for invariant updates, referred to as the IEKF. The IEKF addresses the nonlinear nature of motion models used in navigation, particularly how one can conform to the geometric constraints related to representing attitude by containing the state in a Lie group framework.

2.2.1 The Linear Gaussian Kalman Filter

In the domain of control theory and state estimation, it is often often desirable to estimate the state variables as a joint probability distribution rather than basing it on single measurements. The most popular approach is arguably the Kalman filter [10], which has been used extensively since the 1960s. The Kalman filter recursively estimates the true state using measurements and inputs to the system over multiple time steps. By utilizing prior knowledge of the process and measurement models, the Kalman filter will output an optimal estimate. The prediction and update steps are typically performed at a low computational cost, making the filter ideal for real-time estimation.

Consider the linear discrete-time system defined as

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \qquad k = 1...K \tag{2.27a}$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{n}_k, \qquad k = 0...K \tag{2.27b}$$

where k is the discrete-time index and

- $\mathbf{x}_k \in \mathbb{R}^N$ is the system state
- $\mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0)$ is the initial state
- $\mathbf{v}_k = \mathbf{B}_k \mathbf{u}_k \in \mathbb{R}^N$ is the system input
- $\mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ is the process noise
- $\mathbf{y}_k \in \mathbb{R}^M$ is the measurement
- $\mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise

Matrix $\mathbf{A}_k \in \mathbb{R}^{N \times N}$ is the transition matrix and $\mathbf{C}_k \in \mathbb{R}^{M \times N}$ is the observation matrix, which respectively define the system dynamics and the measurement model at time step k. It is assumed that $\mathbf{x}_0$, $\mathbf{w}_k$ and $\mathbf{n}_k$ are uncorrelated to each other and themselves at different time steps. Using a Bayesian inference approach [11], the prior estimate at $k-1$ is

$$p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}) \tag{2.28}$$

Using $\mathbf{v}_k$, the prediction step gives us the prior at time step k,

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_k, \check{\mathbf{P}}_k) \tag{2.29}$$

with mean and covariance

$$\check{\mathbf{x}}_k = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k \tag{2.30a}$$

$$\check{\mathbf{P}}_k = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^\top + \mathbf{Q}_k \tag{2.30b}$$

using the linear motion model. In general, there is no closed-form solution to the prediction step, i.e. the Chapman-Kolmogorov equation which is introduced below in Equation 2.35. When Bayes' formula is applied in the update step, there are no guarantees for the expression to even be a valid Probability Density Function (PDF) [12]. The linear Gaussian Kalman filter is a special case in which the closed-form solution exists under

the assumption of a Gaussian initial PDF in addition to the Markov model and likelihood being linear and Gaussian.

In the Markov model, the underlying assumption implies that the PDF of future states, conditioned on the current state, is solely influenced by the present state and not influenced by past states. At time k, the joint density of the state and latest measurement is

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) &= \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{C}_k \check{\mathbf{x}}_k \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}}_k & \check{\mathbf{P}}_k \mathbf{C}_k^\top \\ \mathbf{C}_k \check{\mathbf{P}}_k & \mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^\top + \mathbf{R}_k \end{bmatrix} \right) \\ &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix} \right) \end{aligned} \tag{2.31}$$

The conditional density for the current state at time step k, often referred to as the posterior, is then given by Bayesian inference theory

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_y), \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}) \tag{2.32}$$

This leads to the the Kalman filter equations,

$$\textbf{Predictor} \begin{cases} \check{\mathbf{P}}_k = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^\top + \mathbf{Q}_k \\ \check{\mathbf{x}}_k = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k \end{cases} \tag{2.33a}$$

$$\textbf{Kalman gain} \left\{ \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{C}_k^\top (\mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^\top + \mathbf{R}_k)^{-1} \right. \tag{2.33b}$$

$$\textbf{Corrector} \begin{cases} \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k \\ \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_k) \end{cases} \tag{2.33c}$$

Here, the notation $\check{(\cdot)}$ is used to denote a prior estimate and $\hat{(\cdot)}$ is used for the posterior estimates. The term $\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_k$ is often refereed to as the innovation, and indicates how much the prior estimated measurement differs from the actual measurement. Together with the Kalman gain, it is used to correct the estimate when measurements become available. The Kalman gain effectively assigns appropriate weights to the contribution of the innovation. Other variations of the Kalman filter follow the same structure of a predict-update scheme, using prior knowledge about the motion model to propagate the estimate and uncertainty until a measurement is available and the estimate is corrected.

Under the assumption of Gaussian noise and prior with linear motion and measurement models, the posterior will remain a Gaussian. In this case, the mean coincides with the mode. The Kalman filter is optimal in the sense that it is unbiased and the covariance of the filter is exactly at the Cramér–Rao Lower Bound (CRLB), meaning we cannot be more certain of the estimate. From Section 3.3 of [11], we know that the covariance of the error is perfectly modeled by the estimated covariance and we should expect the error to decay to zero after an infinite number of trials. The filter is therefore refereed to as the best linear unbiased estimator.

2.2.2 The Extended Kalman Filter

The Kalman filter introduced in Section 2.2.1 and its optimality proofs can only be applied for linear and Gaussian systems, but nonlinear and non-Gaussian derivations exist and are

widely used in academia and industry. The EKF is considered the de facto nonlinear state estimator in many applications. The EKF uses Taylor series expansion to obtain a linearized model around a working point. This approach can be highly effective for non-Gaussian systems with mild nonlinearities according to [11]. Unfortunately, there are no optimality guarantees for the EKF. The main weakness of the EKF is the propagation of the estimation error through a first-order linearization. The linearization is performed around the estimated trajectory, which may differ significantly from the true state. If the estimate is too far away from the true state, the linearization does not hold and may lead to a positive feedback loop in the error dynamics, which causes the filter to diverge. For stronger nonlinearities this distance is effectively reduced, rendering the filter unstable.

Consider the motion and measurement models of the nonlinear form

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k), \qquad k = 1...K \tag{2.34a}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k), \qquad k = 0...K \tag{2.34b}$$

where k is the discrete-time index of which K is the maximum, $\mathbf{f}$ and $\mathbf{g}$ are the nonlinear motion model and nonlinear observation model, respectively. Unlike the linear case, we will not make any assumptions on the random variables. The derivation of the EKF is made by invoking the assumption of a Markov process. By combining the Markov property with the introduction of the hidden state $\mathbf{x}_{k-1}$, we arrive at the Chapman-Kolmogorov integral of the Bayes filter:

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \eta p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1} \tag{2.35}$$

For more details, the reader is pointed to Section 4.2.2 of [11]. The Bayes filter adheres to the predict-update scheme, but cannot be implemented for two important reasons: The PDFs are continuous functions and require an infinite amount of memory to be represented. Therefore, only an approximation of the belief can be made. This is done e.g. through an approximate Gaussian representation or a finite number of samples from the PDF. Second, the integral part is intractable, except for the linear Gaussian case which has a closed-form solution. This integral too must be approximated, either by Monte Carlo integration or a linearization of the motion and observation models. As Monte Carlo integration is computationally expensive, the latter is often chosen. Significant resources are put into handling the two issues, and only a subset of them are discussed in this thesis.

The equations for the EKF are obtained by constraining the noise and belief to be Gaussian,

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k) \tag{2.36}$$

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q_k}) \tag{2.37}$$

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R_k}) \tag{2.38}$$

Since the Gaussian PDFs of the noise may become non-Gaussian when transformed through a nonlinearity, we cannot assume they are additive and instead represent them through linearization of the motion and observation models. The linearized motion and observation

model around the state estimate prior becomes,

$$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{w}'_k \tag{2.39a}$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k) \approx \check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{n}'_k \tag{2.39b}$$

in which

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}) \tag{2.40a}$$

$$\mathbf{F}_{k-1} = \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}}\Big|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}} \tag{2.40b}$$

$$\mathbf{w}'_k = \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{w}_k}\Big|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, \mathbf{0}} \mathbf{w}_k \tag{2.40c}$$

$$\check{\mathbf{y}}_k = \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}) \tag{2.40d}$$

$$\mathbf{G}_k = \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\Big|_{\check{\mathbf{x}}_k, \mathbf{0}} \tag{2.40e}$$

$$\mathbf{n}'_k = \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k}\Big|_{\check{\mathbf{x}}_k, \mathbf{0}} \mathbf{n}_k \tag{2.40f}$$

The equations for the classic recursive update EKF are obtained by inserting Equations 2.39 into 2.35. The result is similar in structure to the linear Gaussian Kalman filter described in Equations 2.33,

$$\textbf{Predictor} \begin{cases} \check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^\top + \mathbf{Q}'_k \\ \check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) \end{cases} \tag{2.41a}$$

$$\textbf{Kalman gain} \left\{ \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{G}_k^\top (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^\top + \mathbf{R}'_k)^{-1} \right. \tag{2.41b}$$

$$\textbf{Corrector} \begin{cases} \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\check{\mathbf{P}}_k \\ \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0})) \end{cases} \tag{2.41c}$$

where $\mathbf{Q}'_k = \mathbb{E}[\mathbf{w}'_k \mathbf{w}'^\top_k]$ and $\mathbf{R}'_k = \mathbb{E}[\mathbf{n}'_k \mathbf{n}'^\top_k]$ are the noise covariances in which the motion and observation model Jacobians are embedded. Another distinct difference from the linear Gaussian filter is the nonlinear functions used to propagate the estimate mean. There are little to no guarantees for the optimality or efficiency of the EKF for a given system. The Kalman gain for the EKF is computed under the assumption that the estimation error is small and can be propagated through a first-order linearization of the dynamics. Since the operating point is located at the mean of the estimated trajectory, it may differ from the true state by a significant amount. If the estimate differs significantly from the true trajectory, the linearization does not hold and the filter might become inconsistent or biased and in the worst case diverge due to amplification of the error.

2.2.3 The Invariant Extended Kalman Filter

Proving stability for the EKF is nontrivial, even locally [13]. Convergence can be proved for certain classes of nonlinear systems [14], but few practical examples hold such properties. This motivates the use of an IEKF, a variant of the EKF modified to operate on

Lie group state spaces. The IEKF can be proven to converge locally around any trajectory for a broad class of systems implemented as a Lie group in the matrix form. To show this property, consider the continuous time process model with state $\mathcal{X}_t \in \mathcal{G}$ at time t evolving on a matrix Lie Group $\mathcal{G} \subset \mathbb{R}^{N\times N}$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{X}_t = \mathbf{f}(\mathcal{X}_t, \mathbf{u}_t) \tag{2.42}$$

Consider the state estimate $\hat{\mathcal{X}}_t$. The left and right invariant errors between the estimated and true state are

$$\text{Right invariant}: \quad \eta_t^r = \hat{\mathcal{X}}_t \mathcal{X}_t^{-1} \tag{2.43a}$$

$$\text{Left invariant}: \quad \eta_t^l = \mathcal{X}_t^{-1} \hat{\mathcal{X}}_t \tag{2.43b}$$

In the case that the two trajectories are equal, the errors are reduced to the identity. According to Theorem 1 in [6], a system is group affine if

$$\mathbf{f}(\mathcal{X}_1\mathcal{X}_2, \mathbf{u}_t) = \mathbf{f}(\mathcal{X}_1, \mathbf{u}_t)\mathcal{X}_2 + \mathcal{X}_1\mathbf{f}(\mathcal{X}_2, \mathbf{u}_t) - \mathcal{X}_1\mathbf{f}(\mathbf{I}, \mathbf{u}_t)\mathcal{X}_2 \tag{2.44}$$

is satisfied for all positive t and $\mathcal{X}_1, \mathcal{X}_2 \in \mathcal{G}$ where $\mathbf{I} \in \mathcal{G}$ is the identity matrix of the group. If this condition is satisfied, the right and left invariant errors satisfy

$$\frac{\mathrm{d}}{\mathrm{d}t}\eta_t^r = \mathbf{g}^r(\eta_t^r, \mathbf{u}_t) = \mathbf{f}(\eta_t^r, \mathbf{u}_t) - \eta_t^r\mathbf{f}(\mathbf{I}, \mathbf{u}_t) \tag{2.45a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\eta_t^l = \mathbf{g}^l(\eta_t^l, \mathbf{u}_t) = \mathbf{f}(\eta_t^l, \mathbf{u}_t) - \mathbf{f}(\mathbf{I}, \mathbf{u}_t)\eta_t^l \tag{2.45b}$$

Note that the error dynamics in this case are independent of the true state trajectory, which is referred to as autonomous error evolution. Consider the Lie algebra of $\mathcal{G}$ denoted $\mathfrak{g}$ and $\xi \in \mathbb{R}^{\dim \mathfrak{g}}$. We can then define the matrix $\mathbf{A}_t^i$ to satisfy

$$\mathbf{g}^i(\exp(\xi^\wedge), \mathbf{u}_t) = (\mathbf{A}_t^i\xi)^\wedge + \mathcal{O}(\|\xi\|^2) \tag{2.46}$$

such that for all $t > 0$, ξ_t is defined by the linear differential equation in $\mathbb{R}^{\dim \mathfrak{g}}$

$$\frac{\mathrm{d}}{\mathrm{d}t}\xi_t^i = \mathbf{A}_t^i\xi_t^i, \quad i \in \{l, r\} \tag{2.47}$$

According to Theorem 2 of [6], if $\exp(\xi_0^i) = \eta_0^i$ the following mapping between ξ_t^i and the true nonlinear error η_t^i will hold for all errors at any $t > 0$:

$$\eta_t^i = \exp(\xi_t^i) \tag{2.48}$$

The result is commonly referred to as the log-linear property in the literature and means that the logarithm of the IEKF's linearized error dynamics are independent of the true state under the assumption of affinity. As shown in Theorem 4 of [6], the estimate from the IEKF is an asymptotically stable observer around any trajectory.

Consider the noisy continuous time system, which obeys the group-affine dynamics,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{X}_t = \mathbf{f}(\mathcal{X}_t, \mathbf{u}_t) - \mathcal{X}_t \mathbf{w}_t^{\wedge}, \qquad t \in [0, T] \tag{2.49a}$$

$$\text{Left invariant :} \quad \mathbf{y}_t = \mathcal{X}_t \mathbf{d} + \mathbf{n}_t \tag{2.49b}$$

$$\text{Right invariant :} \quad \mathbf{y}_t = \mathcal{X}_t^{-1} \mathbf{d} + \mathbf{n}_t \tag{2.49c}$$

in which $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ are the process and measurement noises, respectively. The vector $\mathbf{d} \in \mathbb{R}^{\dim \mathcal{X}}$ is known for the given measurement and relates the state matrix to the measurement. There are now two equivariant linearized measurement models, one for left and one for right invariant observations. The nonlinear case is further discussed in Section 3.3.3. The left invariant observations are usually associated with measurements of state variables in a global frame, whereas the right invariant observations are commonly used for measurements in the local frame. The IEKF follows a predict-update scheme like the EKF, but the formulation of the innovation terms is altered such that it can be composed with the Lie group representation of the state estimate. Using the same definitions as in Equation 2.40 adapted to the new observation models we get the left IEKF equations,

$$\textbf{Predictor} \begin{cases} \check{\mathbf{P}}_k = \mathbf{\Phi}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{\Phi}_{k-1}^{\top} + \mathbf{\Phi}_{k-1} \mathbf{Q}_k \mathbf{\Phi}_{k-1}^{\top} h \\ \check{\mathcal{X}}_k = \mathbf{f}(\hat{\mathcal{X}}_{k-1}, \mathbf{u}_k) \end{cases} \tag{2.50a}$$

$$\textbf{Kalman gain} \begin{cases} \mathbf{S}_k = (\mathbf{H}_k \check{\mathbf{P}}_k \mathbf{H}_k^{\top} + \hat{\mathbf{R}}_k)^{-1} \\ \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}_k^{\top} \mathbf{S}_k \end{cases} \tag{2.50b}$$

$$\textbf{Corrector} \begin{cases} \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \check{\mathbf{P}}_k \\ \hat{\mathcal{X}}_k = \check{\mathcal{X}}_k \exp(\mathbf{K}_k (\check{\mathcal{X}}_k^{-1} \mathbf{y}_k - \mathbf{d})) \end{cases} \tag{2.50c}$$

where the left invariant observation model of Equation 2.49b has been left multiplied with $\hat{\mathcal{X}}_k^{-1}$ to obtain $\hat{\mathcal{X}}_k^{-1} \mathbf{y}_k = \eta_k^{-1} \mathbf{d} + \hat{\mathcal{X}}_k^{-1} \mathbf{n}_k$ such that a conventional Kalman gain can be applied due to the first-order approximation $\exp(\xi) \approx \mathbf{I} + \xi^{\wedge}$. The $\mathbf{G}_k$ matrix from the EKF Equations 2.41 is then replaced by $\mathbf{H}_k$ which is implicitly defined through $\mathbf{H}_k \xi = \xi^{\wedge} \mathbf{d}$. The discrete time transition matrix at time step k is here denoted $\mathbf{\Phi}_k = \exp(\mathbf{A}_t h)$, where h is the time duration of a time step. The introduction of $\hat{\mathbf{R}}_k$ is to reflect the modified measurement noise, $\hat{\mathbf{n}}_k = \hat{\mathcal{X}}_k^{-1} \mathbf{n}_k$. The right IEKF are almost identical, but reflects the use of a right invariant error and observation model,

$$\mathbf{H}_k \xi = -\xi^{\wedge} \mathbf{d} \tag{2.51a}$$

$$\hat{\mathcal{X}}_k = \exp(\mathbf{K}_k (\check{\mathcal{X}}_k \mathbf{y}_k - \mathbf{d})) \check{\mathcal{X}}_k \tag{2.51b}$$

$$\mathbf{Q}_k \leftarrow \mathbf{Ad}_{\mathbf{f}(\hat{\mathcal{X}}_k, \mathbf{u}_k)}^{\top} \mathbf{Q}_k \mathbf{Ad}_{\mathbf{f}(\hat{\mathcal{X}}_k, \mathbf{u}_k)} \tag{2.51c}$$

in addition to the modified measurement noise now being $\hat{\mathbf{n}}_k = \hat{\mathcal{X}}_k \mathbf{n}_k$.

For applications were the system state fits into a Lie group framework, the filter will outperform the EKF because the error dynamics of the IEKF are independent of the true

state. In state representations containing rotations, the linearization of the EKF fails to consider the geometric constraints of the attitude representation. The linear update of the EKF attempts to linearize the nonlinear space representing the orientation. By doing so, it introduces estimation error and the new estimate will likely not be a valid rotation. Tricks are commonly applied to ensure that the new attitude representation is valid, but the IEKF linearizes on the Lie algebra which is a linear space to begin with. Using the exponential mapping, the IEKF update ensures that the estimate moves along the manifold. The same goes for the covariance, which is expressed in the Lie algebra. The EKF on the other hand, would move tangentially to the manifold in the direction of the covariance and thus leave the manifold altogether. The difference in update for the EKF and IEKF is illustrated in Figure 2.3.

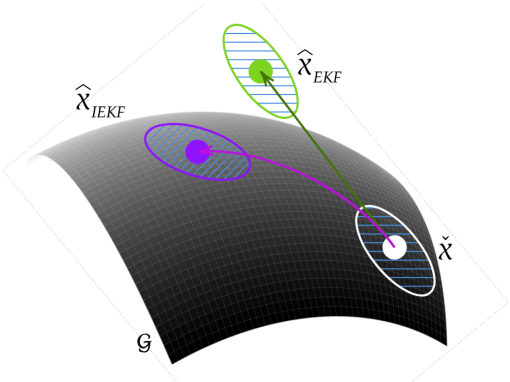


Figure 2.3: Visualization of differences in the EKF and IEKF updates. The prior estimate $\check{\mathcal{X}}$ is consistent with the manifold $\mathcal{G}$ and has a covariance represented by the shaded region. The updated estimates for the EKF will then leave the manifold, whereas the IEKF update remains consistent with the group $\mathcal{G}$. Adapted from [7].

For non-zero rotations, the linearization error of the EKF will increase and potentially cause estimator inconsistencies or divergence in the extreme cases. For high-accuracy navigation purposes, the IEKF improves consistency and performance compared to the regular EKF. Not only the conventional EKF is outperformed by the IEKF. For an estimate evolving on a subgroup of $\mathcal{G}$, the IEKF will remain on the subgroup through the exponential map relating the manifold to its Lie algebra on which the IEKF linearizes. Other means of attitude estimation, such as the Multiplicative Extended Kalman Filter (MEKF) are not guaranteed to remain on the subgroup despite ensuring consistency with $\mathcal{G}$. According to [7], the IEKF can be seen as an extension of the MEKF for more general state spaces, taking advantage of the group affine dynamics and group actions as updates to obtain the autonomous error evolution. A full comparison between the MEKF and the IEKF is outside the scope of this thesis, but the reader is pointed to [15], [16] and [17] for more details.

2.3 Camera Geometry

The intention of this section is to introduce basic camera geometry the concept of geometric intrinsic camera calibration, which aims to identify the optical parameters of a camera relating a 3D point in the world scene to 2D points in the camera's image plane. The output of this process is the intrinsic camera matrix, which is used in many computer vision tasks such as pose estimation, augmented reality and 3D scene reconstruction. Section 2.4 discusses the utilization of intrinsic parameters and the model used to delineate image formation, in the application of ascertaining the pose of a camera based on known feature points and their corresponding representations in the image.

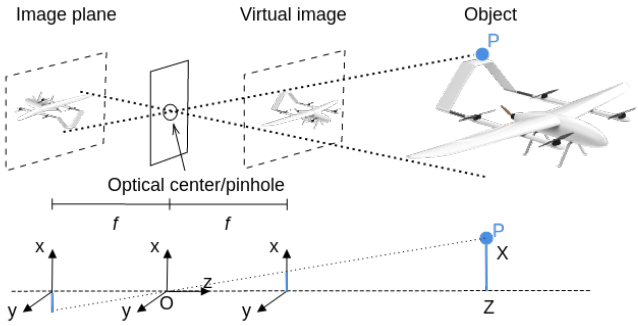


Figure 2.4: Geometry of the pinhole camera model.

The geometry of the pinhole camera model is shown in Figure 2.4. The optical center is the origin for the 3D world coordinates, denoted by X, Y, Z. The virtual image plane is introduced to remove the need for image rotation when performing computations. Both the image plane and the virtual image plane are located a distance f away from the optical center in the z-direction. The distance f is commonly known as the focal distance. The geometric properties of similar triangles can be exploited to arrive at the following:

$$\frac{x}{f} = \frac{X}{Z}, \frac{y}{f} = \frac{Y}{Z} \implies x = f\frac{X}{Z}, y = f\frac{Y}{Z} \tag{2.52}$$

In practical applications, lenses are used to focus parallel light rays from the 3D scene to the focal point. The lens introduces additional issues such as focus, vignetting, exposure, aberration and distortion. These issues are discussed here, but the reader is referred to Section 2.2.3 of [18] for more details on the topic of optics. Equation 2.52 is the first step in deriving the ideal transform from world coordinates to pixel coordinates. Before deriving the transform, three coordinate frames are introduced:

- The world frame (W).
- The camera frame (C).
- The image frame (O).

The origin of the camera frame is placed in the optical center. The camera frame is related to the world frame which is arbitrarily defined through an rotation $\mathbf{R}$ and a transform $\mathbf{T}$. Finally, the image frame is located in the virtual image plane, with the distance of one focal length from the camera frame in the z-direction. The frames are illustrated in Figure 2.5.

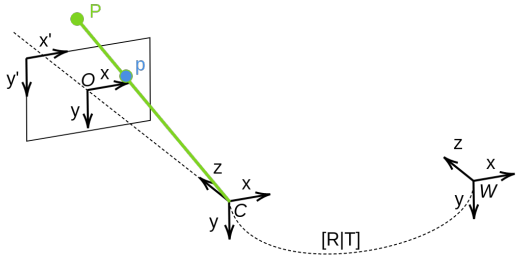


Figure 2.5: World, camera and image coordinate frames.

In practical applications, it is inconvenient to work directly on the image frame, due to pixel coordinates potentially being negative. This motivates the introduction of pixel coordinates. In Figure 2.5, the two-dimensional pixel coordinates (x', y') are shown in the top left of the virtual image plane. They represent the discrete index of the pixels spanning the image plane: $(x', y') \in [0, W) \times [0, H)$ where W and H is the image width and height in number of pixels.

In general, when transforming a 3D point $\mathbf{X}_w$ in the world frame to pixel coordinates, the following transforms are applied:

1. Using $\mathbf{R}$ and $\mathbf{T}$, apply a rigid body transformation to obtain $\mathbf{X}_w$ in camera coordinates, denoted $\mathbf{X}_c$.
2. Perform a perspective projection of the 3D point using the pinhole camera model to the 2D image plane. Denote the point as (x, y).
3. Transform the image coordinates to discrete pixel coordinates, (x', y').

In some applications, only the geometric relation of the world point to the camera is of interest. In this case, the first step can be omitted. Before discussing the details of the steps above, some additional geometric primitives will be introduced. When working with 2D points, such as projections of 3D points to an image, the intuitive representation is a two-dimensional representation such as $\mathbf{x} = (x, y) \in \mathbb{R}^2$. Alternatively, one could use homogeneous coordinates $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}^2$. The 2D projective space is defined such that $\mathcal{P}^2 = \mathbb{R}^3 - (0, 0, 0)$. For consistency, the same notation as [18] is used. The points in which $\tilde{w} = 0$ represent the ideal points, which are located at infinity. Such points do not have an inhomogeneous representation. For all other points in $\mathcal{P}^2$, it is straightforward to convert back to inhomogeneous coordinates by dividing with $\tilde{w}$: $\bar{\mathbf{x}} = (x, y, 1) = (\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}, \frac{\tilde{w}}{\tilde{w}}) = (\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}, 1)$. The vector $\bar{\mathbf{x}}$ is commonly referred to as the augmented vector. The projective space extends beyond the two-dimensional case. In

general, any point in a projective space $\mathcal{P}^n$ is described by a vector of $n+1$ components. A key property of the projective space, is that any vector in said space is only defined up to a scaling factor. Intuitively, this makes sense in the process of image formation. All 3D points along the line casted from a point in the image plane will project to the same point, e.g. all points along the green line in Figure 2.5. The loss of information when going from 3D to 2D space means that only the direction of the homogeneous vector is of importance, not the scale. Specifically, all points $\lambda\tilde{\mathbf{x}}$ describe the same point $\tilde{\mathbf{x}} \in \mathcal{P}^n \; \forall\lambda \in \mathbb{R} \setminus \{0\}$

An additional benefit of homogeneous coordinates written in the augmented form, is the more compact notation. For example, consider the 2D rigid body motion $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{T}$ where $\mathbf{R}$ is a 2×2 orthonormal rotation matrix and $\mathbf{T}$ is a two-dimensional translation vector. Using matrix form, one could then write $\mathbf{x}' = [\mathbf{R}\ \mathbf{T}]\bar{\mathbf{x}}$. Alternatively, one could use a full-rank 3×3 matrix which preserves the homogeneous form:

$$\bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \bar{\mathbf{x}} \tag{2.53}$$

where $\mathbf{0}$ is the two-dimensional zero vector. The example generalized to higher dimensionalities and is common for representing rigid body motion in $SE(3)$. By appending the bottom row, the full-rank matrix can be used to calculate the inverse transform or chaining matrix multiplication. This is the basis for creating the transform between a 3D point and its corresponding pixel coordinates. Recall the first step in this process, which is to obtain the camera coordinates $\mathbf{X}_c = (X, Y, Z)$ from the world coordinates $\mathbf{X}_w = (X_0, Y_0, Z_0)$. Using homogeneous coordinates, the following transform is applied:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{2.54}$$

Using Equation 2.52, step two can be performed the following way:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.55}$$

Using matrix notation in homogeneous coordinates, this can be rewritten as

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.56}$$

in which the rightmost 3×4 matrix performs a projection and the leftmost 3×3 matrix scales the output. For compactness, it is common to merge the two matrices intro one. The leftmost Z is often rewritten to λ, a scaling factor representing the scale ambiguity of the perspective. Equations 2.54 - 2.56 represent the transforms from a 3D point in the world to 2D image coordinates. The geometric model is referred to as the ideal perspective projection in [19] because the only points contributing to irradiance at point $\mathbf{p}$ in the image

plane are those located on the line between $\mathbf{p}$ and the 3D point $\mathbf{P}$, as shown in Figure 2.5. The ideal pinhole model is an paradigmatic geometric construct, which only approximates a well-focused camera. Effects such as distortion, diffraction and blurring breaks these assumptions, but the model is a good starting point for many computer vision applications. Recall that the origin of the image frame is located at the optical center as shown in Figure 2.4. It is usually preferred to place the origin of the pixel coordinates away from the optical axis. A common choice is the top-left corner as seen in Figure 2.5. In this step, there are two considerations; the scale factors and the offsets between image and pixel coordinates in each direction. The scaling part of the image transformation is performed as follows:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & s_\theta \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.57}$$

The translation to the top-left corner can then be applied with

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2.58}$$

Similarly to Equation 2.56, the two transforms are usually merged into one matrix operation using homogeneous coordinates for a more compact notation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.59}$$

Combining the results from Equations 2.56 and 2.59, one arrives at the following projective equation:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & s_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.60}$$

The leftmost matrix of Equation 2.60 is commonly referred to as the intrinsic camera matrix, $\mathbf{K}$, which is followed by the perspective projection matrix. The elements of $\mathbf{K}$ represent physical properties of the camera:

- fs_x and fs_y represents the focal lengths in X and Y-direction. For a true pinhole camera, these should be equal. In practice, they may differ due to manufacturing flaws, lens distortion or other imperfections.
- The ratio $\frac{fs_x}{fs_y}$ is often called the aspect ratio.
- c_x and c_y denotes the distance to the principal point located at the image center in pixel coordinates.
- s_θ refers to the pixel skew, which may occur if the pixels are not perfectly rectangular or if the sensor is not perpendicular to the optical axis. In practice, this value is often close to zero.

A more common representation, where the focal length is embedded in the adjacent s-terms is denoted by

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.61}$$

2.3.1 Determining the Intrinsic Matrix

As discussed, the pinhole camera model is a geometric construct and does not fully encapsulate all aspects of image formation. In most applications it is sufficiently accurate in describing the geometric relation between the pixel coordinate of a point in an image to the 3D coordinates in the camera frame up to scale. A good approximation of the intrinsic calibration parameters will therefore greatly improve the quality of perception. The intrinsic matrix is often assumed to be constant for a camera, unless it is subject to excessive vibrations or external forces. This means that the calibration can be performed once in a controlled environment and the parameters can be saved for future use.

The process of determining the intrinsic parameters is often referred to as intrinsic calibration. According to [18], the classical approach to intrinsic calibration involves estimating the internal camera parameters while simultaneously estimating the extrinsic pose of the camera with respect to some known calibration pattern. There are numerous ways to perform the intrinsic calibration, adhering to a trade-off between accuracy and complexity in calibration rig and setup. Within the domain of mobile robotics, one of the most common approaches is to move a planar calibration pattern in the camera's field of view and saving a picture from each location. In this approach, the location of the calibration pattern does not need to be known in advance. This method requires that the pose of the calibration target is calculated together with the intrinsics, which in general is less accurate than the N-planes calibration approach where the location of the calibration boards is known. However, it is sufficiently accurate for most applications and is less cumbersome than the alternative.

The calibration pattern can take many shapes and display various visual features. One of the most common target is the checkerboard pattern. The accuracy of the calibration depends on the tolerances of the calibration target manufacturing. As a rule of thumb, the tolerances of the manufacturing should be at least one order of magnitude lower than the desired accuracy of the calibration. After collecting the images of the calibration target, features from the calibration pattern are extracted for each image. Using the prior information of the planar calibration pattern's geometry, a homography matrix describing the projective transform between the calibration points of image i in the world frame, $(X_i, Y_i, Z_i, 1)$ and the respective pixel coordinates (x_i', y_i') can be computed.

Only the relative pose between the camera and the calibration target is of interest during the calibration. This means that without loss of generality, one can assume that the calibration target is located in $Z_0 = 0$ of the world frame. The following simplification then holds

true according to [20]:

$$\begin{aligned}\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} fs_x & fs_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ 0 \\ 1 \end{bmatrix} \\ &= \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ 1 \end{bmatrix}\end{aligned} \tag{2.62}$$

$\mathbf{r}_i$ denotes the i-th column of $\mathbf{R}$. The 3×3 homography $\tilde{\mathbf{H}}$ can now be defined up to scale:

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{T} \end{bmatrix} \tag{2.63}$$

The matrix $\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}$ is known as the Image Of The Absolute Conic (IAC), which is useful for representing orthogonality in an image and it frequently occurs in projective geometry. Since its elements are composed of intrinsic parameters only, it is commonly used in camera calibration. According to Section 8.5 of [21] and Appendix A of [20], Equation 2.63 can be rewritten using the orthonormal properties of the columns in $\mathbf{R}$:

$$\mathbf{h}_1^\top \mathbf{B} \mathbf{h}_2 = 0 \tag{2.64a}$$

$$\mathbf{h}_1^\top \mathbf{B} \mathbf{h}_1 = \mathbf{h}_2^\top \mathbf{B} \mathbf{h}_2 \tag{2.64b}$$

There are in total six DOF related to the extrinsic parameters, three for rotation and three for translation. The homography matrix is composed of nine elements, but is only defined up to scale. Therefore, the homography only has eight DOF. The result is only two constraints on the intrinsic parameters from a single homography. Consider the matrix $\mathbf{B}$ on the form

$$\begin{aligned}\mathbf{B} &= \mathbf{K}^{-T}\mathbf{K}^{-1} \\ &= \begin{bmatrix} \frac{1}{f_x^2} & -\frac{s}{f_x^2 f_y} & \frac{c_y s - c_x f_y}{f_x^2 f_y} \\ -\frac{s}{f_x^2 f_y} & \frac{s^2}{f_x^2 f_y^2} + \frac{1}{f_y^2} & -\frac{s(c_y s - c_x f_y)}{f_x^2 f_y^2} - \frac{c_y}{f_y^2} \\ \frac{c_y s - c_x f_y}{f_x^2 f_y} & -\frac{s(c_y s - c_x f_y)}{f_x^2 f_y^2} - \frac{c_y}{f_y^2} & \frac{(c_y s - c_x f_y)^2}{f_x^2 f_y^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix} \\ &= \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}\end{aligned} \tag{2.65}$$

The explicit form of the IAC reveals a symmetric structure.
Let $\mathbf{b} = [B_{11}\ B_{12}\ B_{22}\ B_{13}\ B_{23}\ B_{33}]^\top$ hold the nonduplicate elements of $\mathbf{B}$. The fundamental intrinsic constraints of Equations 2.64 can therefore be rewritten as

$$\begin{bmatrix} \mathbf{v}_{12}^\top \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^\top \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{2.66}$$

in which

$$\mathbf{v}_{ij} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix} \tag{2.67}$$

letting h_{ij} denoting the j-th element of the i-th column in $\tilde{\mathbf{H}}$. In general for a system of n images, the homogeneous representation of the fundamental intrinsic constraints can be rewritten as

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{2.68}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix. Assuming the number of images $n \geq 3$ which is usually the case during image calibration, there is in general a unique solution for $\mathbf{b}$ defined up to scale[2]. The solution of Equation 2.68 can be retrieved as the eigenvector associated with the smallest eigenvalue of $\mathbf{V}^\top\mathbf{V}$. One can compute the parameters of $\mathbf{K}$ from $\mathbf{b}$ as follows:

$$c_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \tag{2.69a}$$

$$\lambda = B_{33} - \frac{B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \tag{2.69b}$$

$$f_x = \sqrt{\frac{\lambda}{B_{11}}} \tag{2.69c}$$

$$f_y = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}} \tag{2.69d}$$

$$s = -\frac{B_{12}f_x^2 f_y}{\lambda} \tag{2.69e}$$

$$c_x = \frac{sc_y}{f_y} - \frac{B_{13}f_x^2}{\lambda} \tag{2.69f}$$

2.4 Pose Estimation

This section aims to introduce the main concepts of conventional iterative extrinsic pose estimation and introduce the Infinitesimal Plane-Based Pose Estimation (IPPE) as an analytical and fast alternative solution when the structure of the feature points adhere to a certain structure. By continuing on the procedure from Section 2.3, we can now estimate

[2]There exists some degenerate configurations, such as parallel target planes. The reader is referred to [20] for more details.

the camera's pose under the assumption that $\mathbf{K}$ is known. The pose can be retrieved by

$$\mathbf{r}_1 = \lambda \mathbf{K}^{-1}\mathbf{h}_1 \tag{2.70a}$$
$$\mathbf{r}_2 = \lambda \mathbf{K}^{-1}\mathbf{h}_2 \tag{2.70b}$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{2.70c}$$
$$\mathbf{T} = \lambda \mathbf{K}^{-1}\mathbf{h}_3 \tag{2.70d}$$

using the property of orthonormality in $\mathbf{R}$ to compute $\mathbf{r}_3$ via the cross-product and defining $\lambda = \frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|}$. In general, there is some degree of noise involved in the procedure. As a result, $\mathbf{R}$ will likely not satisfy the properties of a proper orthogonal rotation matrix, namely $\mathbf{R}^\top = \mathbf{R}^{-1}$ and $\det \mathbf{R} = 1$. The procedure of finding the best rotation matrix $\mathbf{R}$ to a given 3×3 matrix $\mathbf{Q}$ can be done by minimizing the Frobenius norm of $\mathbf{R} - \mathbf{Q}$. More precisely, the problem is defined as follows:

$$\min_{\mathbf{R}} \|\mathbf{R} - \mathbf{Q}\|_F^2 \quad s.t. \quad \mathbf{R}^\top \mathbf{R} = \mathbf{I} \tag{2.71}$$

where $\mathbf{I}$ is the 3×3 identity matrix. We can rewrite $\|\mathbf{R} - \mathbf{Q}\|_F^2 = \text{trace}((\mathbf{R} - \mathbf{Q})^\top (\mathbf{R} - \mathbf{Q})) = \text{trace}(\mathbf{R}^\top\mathbf{R} - \mathbf{R}^\top\mathbf{Q} - \mathbf{Q}^\top\mathbf{R} + \mathbf{Q}^\top\mathbf{Q}) = 3 + \text{trace}(\mathbf{Q}^\top\mathbf{Q}) - 2\text{trace}(\mathbf{R}^\top\mathbf{Q})$ using the definition of Frobenius norm from Equation 2.3.1 in [22]. Equivalently to the problem defined in Equation 2.71, one can maximize the trace of $\mathbf{R}^\top\mathbf{Q}$. Following the procedure in Appendix C of [20], let the singular value decomposition of $\mathbf{Q}$ be $\mathbf{USV}^\top$ s.t. $\mathbf{S} = diag(\sigma_1, \sigma_2, \sigma_3)$. By defining an orthonormal matrix $\mathbf{Z} = \mathbf{V}^\top\mathbf{R}^\top\mathbf{U}$, we can write

$$\begin{aligned} \text{trace}(\mathbf{R}^\top\mathbf{Q}) = \text{trace}(\mathbf{R}^\top\mathbf{USV}^\top) &= \text{trace}(\mathbf{V}^\top\mathbf{R}^\top\mathbf{US})^3 \\ &= \text{trace}(\mathbf{ZS}) = \sum_{i=1}^{3} z_{ii}\sigma_i \leq \sum_{i=1}^{3} \sigma_i \end{aligned} \tag{2.72}$$

The property of $\text{trace}(\mathbf{ABC}) = \text{trace}(\mathbf{BCA}) = \text{trace}(\mathbf{CAB})$ is used in the second equality. The last inequality follows from the assumption of orthonormality in $\mathbf{Z}$. The maximum is obtained when $\mathbf{Z} = \mathbf{I}$, e.g. by setting $\mathbf{R} = \mathbf{UV}^\top$.

The solution to Equation 2.68 is generally subject to noise. Under the assumption that this noise is independent and identically distributed, one can refine the estimate with maximum likelihood estimation. By iterating over all points j in image i, the following functional can be used to obtain an accurate estimate:

$$\sum_i \sum_j \|\mathbf{p_{ij}} - \hat{\boldsymbol{\pi}}(\mathbf{K}, \mathbf{R}_i, \mathbf{T}_i, \mathbf{P}_j)\|^2 \tag{2.73}$$

Here, $\mathbf{p}_{ij}$ is point j of image i and $\hat{\boldsymbol{\pi}}$ is the estimated projection of the world point $\mathbf{P}_j$ according to the projection model from Equation 2.60. The functional is the sum of all reprojection errors squared. The reprojection error is a common benchmark for assessing the quality of camera calibration and pose estimation. The problem is nonlinear and can e.g. be solved with Levenberg-Marquardt using the solution from Equation 2.68 as an initial guess. The presence of lens distortion has not been accounted for, and a linear projection model has been assumed in the above derivations. In practice, straight lines in real

life may appear curved due to radial distortion of wide-angle lenses. In practice, various distortion models using polynomial approximations can be applied during the calibration process to estimate this effect and adjust for it as shown in Section 2.1.5 of [18]. This is discussed in Section 3.3.2.

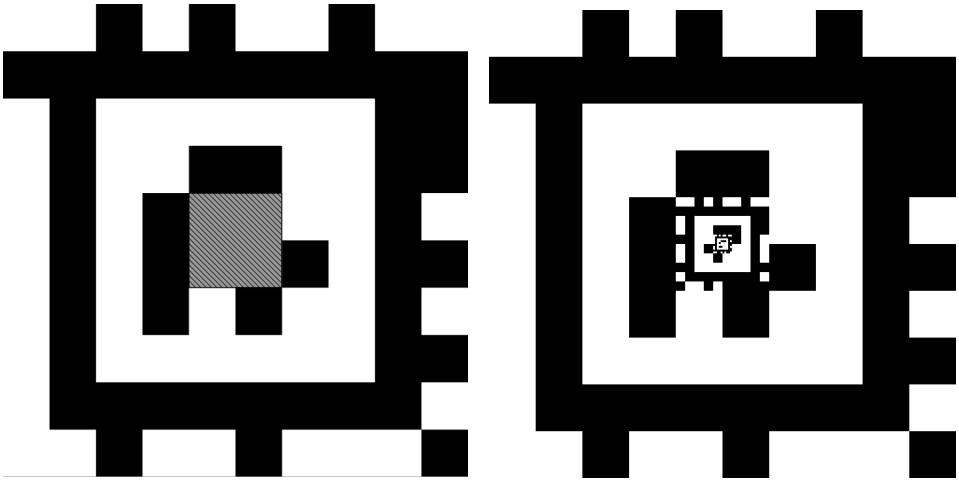
The pose estimation can of course be performed independently from the calibration process, which is often the case in practical applications. The problem of estimating a calibrated camera's pose given a set of 3D points in a scene and the coordinates of their correspondences in the 2D image plane is commonly referred to as the Perspective-n-Point (PnP) problem. Prior information about the geometry of the visual target can in many cases be exploited to make the pose estimation more accurate, reliable and efficient. Consider a flat, square target with a point of interest in each of the four corners such that the points are coplanar and zero centered. This problem solved completely analytically using the IPPE algorithm [23]. Under these assumptions, IPPE is faster and more accurate than traditional iterative PnP methods by exploiting redundancy in the coefficients of the homography. These properties make IPPE useful for real-time applications with a limited computational budget. The IPPE method is further discussed in Section 3.3.3.

2.4.1 AprilTag 3 Fiducial Markers

Fiducial markers are artificial visual features placed in a camera's frame to provide spatial context and are commonly used to determine the scale of a scene. If placed in a known location, they can also be used to determine the camera's position. AprilTag 3 [4] is designed to be faster, more accurate and reliable than its predecessor, while introducing several new tag families.

By allowing data bits outside of the separation border, more data bits can be encoded with a higher Hamming distance between tags. Families in AprilTag 3 can be detected with higher recall and precision at higher Frames Per Second (FPS) than alternative fiducial marker frameworks such as AprilTag 2 and ArUco 3 [4]. The tag families are parameterized by the number of bits, n, in the embedded codeword and the minimum Hamming distance, d, between each codeword. For example, the *custom48h12* family has 48 variable bits in each codeword and a minimum Hamming distance of 12 between each codeword. There are $42,211$ unique tag codewords in this family.

AprilTag codewords are designed such that a rotation by $90°$, $180°$ or $270°$ still maintains the minimum Hamming distance from any other codeword. For the purpose of robust localization, AprilTags are designed to be easily distinguishable from naturally occurring features in the environment. In general, the candidate codewords are considered in lexicographic order. The implementation guarantees the ability to detect $\frac{d}{2}$ erroneous bits and correct up to $\lfloor \frac{d-2}{2} \rfloor$ of them. AprilTag 3 introduces new tag families which are designed to be flexible and customizable, offering circular layouts or recursive ones as shown in Figure 2.6. Other tag families exist, all with different key properties. This thesis focuses on the custom48h12 family due to its recursive structure. The markers used in this system are designed such that each individual marker in the recursion has a unique codeword, which is used to query the marker size. When the size of the currently tracker marker is known, the PnP problem can be solved for the relative pose. As discussed, the IPPE method is



(a) An instance of the AprilTag custom48h12 family. The shaded area in the center can be filled arbitrarily.

(b) A recursive AprilTag, using a depth of three tags from the custom48h12 family.

Figure 2.6: Examples of the AprilTag custom48h12 and how they can be used recursively.

highly efficient for pose estimates of planar, zero centered feature points. For this reason, it is a popular choice for pose estimation based on fiducial markers.

Chapter 3

Method

3.1 Solution Concept and System Overview

The purpose of the vision-based precision landing system is to aid a UAV during a landing sequence in a reliable and safe manner with sufficiently high precision to descend onto a landing platform within its mechanical tolerances. The system is intended for onboard companion computers with a limited computational budget, using affordable sensors which are standard issue for most aerial systems. There are three main components of the proposed system:

1. **Landing target detector:** Process the image stream from the onboard camera and detect visual markers. Using prior information about the landing target configuration, the size of the marker is determined from its encoded bit arrangement and the PnP problem is solved for the UAV's pose relative to the landing target.
2. **Filtering:** The vision-based pose measurement and the UAV's high rate IMU readings are used to estimate the realtive pose in an IEKF.
3. **Autopilot interface:** This module is responsible for reading IMU data and the state of the vehicle to initialize the filter when a landing sequence is started. When initialized, setpoints are generated and forwarded to the UAV based on the filter output. Sanity checks are applied to ensure consistency and the module is designed to be integratable with popular autopilots such as PX4 [1] or ArduPilot [24].

In addition to the main components, the proposal features a custom marker design, a simulation environment to test system modules and integration in addition to a tool for logging and evaluating filter performance. A high-level visualization of the proposed system can be seen in Figure 3.1.

The proposed solution has been devised within the context of the concluding phases of round trip aerial delivery. Specifically, it addresses the scenario in which the UAV has

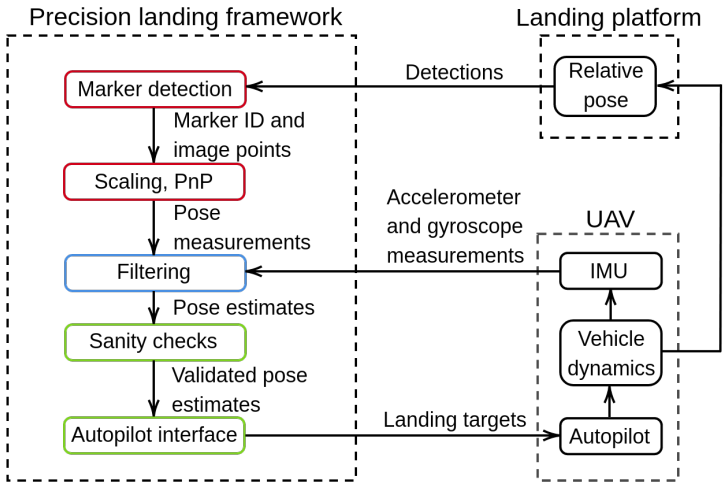


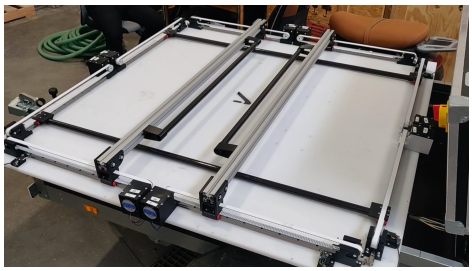
Figure 3.1: Visualization of system overview in the proposed solution. The system consists of three main modules indicated by different colors; landing target detector, extended pose estimation and autopilot interface. The system is designed to be reliable, efficient and deployable to affordable and computationally limited hardware.

successfully executed the delivery and subsequently returned to the operational base. At the base, the UAV is required to land for battery recharging and cargo reloading purposes. The system is based on a VTOL UAV, as shown in Figure 1.1. It possesses a wingspan of $250\,\mathrm{cm}$ and a Maximum Take-Off Mass (MTOM) of $16\,\mathrm{kg}$. These specifications enable it to transport cargo weighing up to $1.2\,\mathrm{kg}$ with a maximum range of $120\,\mathrm{km}$. The proposed solution aims to achieve centimeter-level landing precision using only a single downward-facing camera, the IMU of the autopilot, a companion computer and a solitary external visual target consisting of recursive fiducial markers. Alternative methods include the utilization of Real-Time Kinematic (RTK) GNSS, necessitating a compatible receiver for each UAV, along with the use of either an external or self-hosted service for streaming correction data. In this case, the location of the platform must also be measured precisely, introducing some additional complexity when quickly deploying platforms in the field or to a customer. External services often incur significant expenses, particularly when dealing with a fleet of multiple UAVs, and neither service can ensure uninterrupted availability. Similarly, Ultra-wideband positioning beacons could also be used for this purpose, but that would also require additional hardware on both the platform and UAV.

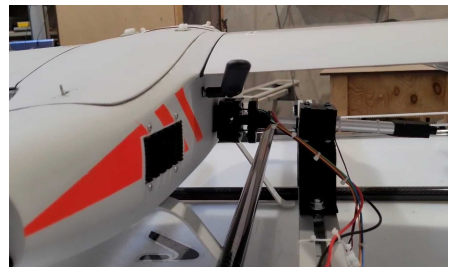
3.2 Geometry of the Landing Platform

The intended use case assumes design freedom and ownership of the landing platform, notably that the platform can be equipped with visual markers of choice and that the approximate location of the landing platform is known. Aviant has developed a landing

platform intended for such use, which is embedded in a car trailer and thus mobile. The landing platform is shown in Figure 3.2.



(a) The automatic alignment system.



(b) The charging interface.

Figure 3.2: Images displaying the landing platform along with its automatic mechanical alignment and charging interface. The visual marker, which aids in precision landing, can be integrated into the floor of the platform. The mechanism for cargo loading is not depicted in the image. *Photographer: Author.*

The landing platform is equipped with guiding cross bars which can be used to align the UAV after it has landed, both in terms of yaw and horizontal position. The mechanical alignment can successfully position and orient the UAV if the initial position is anywhere in the platform's interior and the yaw angle alignment between the UAV and the platform differs less than $35°$. After alignment, a linear actuator connects the onboard Battery Management System (BMS) to external power to charge the batteries. It is assumed that future iterations of the platform will be equipped with lights and heating elements or roofing to ensure visibility and prevent snow from covering the platform. Due to the mobile nature of the landing platform, it can also be placed at strategic locations in the operational area to serve as range extenders on which the UAV can land and recharge before continuing its mission. Despite the platform being mobile, practical applications would greatly benefit from increasing the landing accuracy and precision, thus reducing the required footprint of the platform.

When in the open position, the interior geometry of the cross bars is $102\,\mathrm{cm} \times 103\,\mathrm{cm}$ and the visual markers are to be placed inside this area. The charging interface protrudes $23\,\mathrm{cm}$ vertically from the plane of the landing target, which potentially obstructs the view of the visual markers if viewed from an angle. This is expected to happen, because the initial approach before precision landing is started is limited to the precision of the GNSS module of the UAV.

By reducing the dimensions of the visual marker, this occlusion can be alleviated. Consequently, the altitude at which the onboard camera of the UAV can accurately detect the marker is also decreased. A side view of the geometry is shown in Figure 3.3. The geometric constraints of the marker size and placement are driven by three factors:

1. The error of horizontal position measurements before the precision landing is started.
2. The minimum altitude at which the UAV is considered safe to fly in the landing area, assuming a horizontal position error governed by the GNSS accuracy.

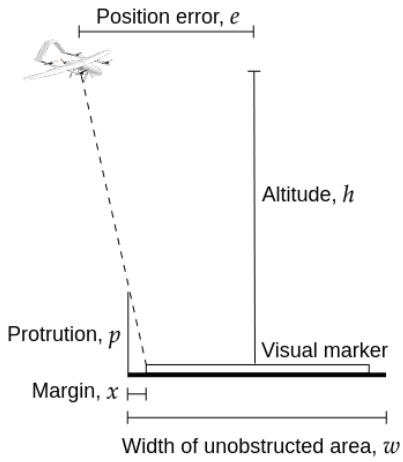


Figure 3.3: Visualization of landing target geometry. The figure is not to scale.

3. The maximum distance to the visual marker from which it reliably can be detected.

The UAV is to initiate the precision landing from no less than $10\,\mathrm{m}$ Above Ground Level (AGL) for safety purposes. The estimated altitude AGL is accurately known down to centimeter precision due to an onboard laser altimeter. For applications with less constraints on the initial altitude AGL for precision landing initialization, the altitude can also be determined from the fiducial marker detection as the UAV descends over the prior belief of the landing platform's location. According to the data sheet of the u-blox M8P-2 GNSS module used [25], the standalone horizontal position accuracy is $2.5\,\mathrm{m}$ Circular Error Probability (CEP). That is, the measured horizontal position will be within $2.5\,\mathrm{m}$ of the true horizontal position $50\,\%$ of the time. Using the geometry of Figure 3.3 and the triangle similarities, the margin from the edge of the visual marker to the cross bars is identified as $x = p\frac{e-\frac{w}{2}}{h-p}$. Since the uncertainty of the landing platform's horizontal position is comparable to that of the UAV, a horizontal position error of $e = 5\,\mathrm{m}$ from the platform center is assumed. Using $h = 10\,\mathrm{m}$, $w = 102\,\mathrm{cm}$ and $p = 23\,\mathrm{cm}$, the desired margin is found to be $10.6\,\mathrm{cm}$. The available area for visual markers was therefore limited to a $90\,\mathrm{cm} \times 90\,\mathrm{cm}$ square, centered on the landing platform.

In addition to occlusion by the platform itself, the landing target must be visible from $10\,\mathrm{m}$ AGL considering the initial horizontal displacement due to erroneous GNSS measurements and the camera's Field Of View (FOV). Similarly to the derivation of the marker size, the geometry shown in Figure 3.4 is used to determine the minimum FOV of the camera. The trigonometric relations reveal that the vertical FOV needs to be $F_y \geq 2\arctan\frac{e+\frac{m}{2}}{h}$. Using $e = 5\,\mathrm{m}$, $m = 90\,\mathrm{cm}$ and $h = 10\,\mathrm{m}$, one obtains $F_y \geq 57.2°$. The calculations were made for the vertical FOV, which is typically the smallest of the two. This is congruent with the $64°$ vertical and $80°$ horizontal FOV of the camera used.

Considering its wings, the aircraft has a higher propensity to encounter significant pertur-

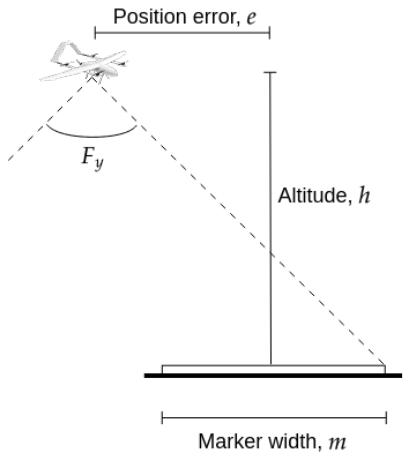


Figure 3.4: Visualization of landing target visibility based on camera FOV. The figure is not to scale.

bations in roll compared to pitch. Therefore, the camera is positioned in a manner that accommodates this behavior. The camera is mounted such that the larger horizontal FOV compensates for a rolling motion, thereby improving visibility of the marker.

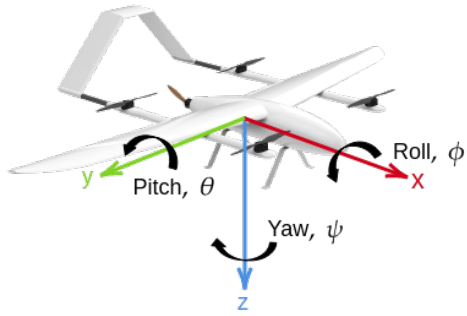
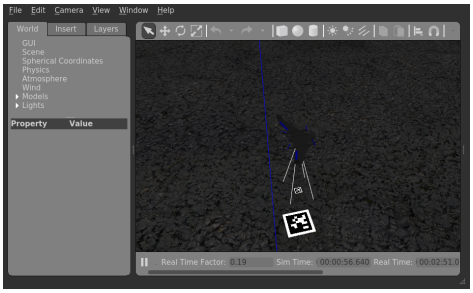


Figure 3.5: Visualization of axes x, y and z of the UAV's body frame and the respective axes for roll, pitch and yaw movement.

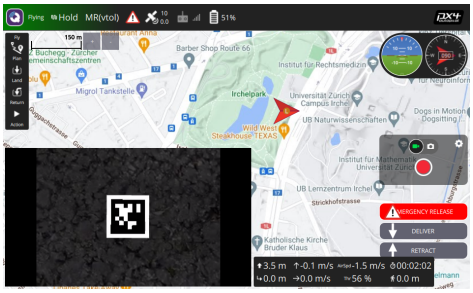
For a camera affixed in a rigid manner to the ventral section of the UAV, facing down along the z-axis as shown in Figure 3.5 and oriented such that the vertical image axis aligns with the x-axis passing through the aircraft's nose, the resulting values leave a $6.8°$ margin in pitch and a $22.8°$ margin in roll. For a stable hover, this coincides with historical attitude data for the UAV platform in windy conditions.

To rapidly prototype and test different marker configurations, a custom VTOL quadplane equipped with a camera was added to PX4's Gazebo based Software In The Loop (SITL) simulator. The model streamed the camera feed over User Datagram Protocol (UDP) to a

receiving node, e.g. the Ground Control Station (GCS). The simulator environment was modified to display visual fiducial markers in an otherwise empty world. The camera of the model was setup with horizontal and vertical FOV to match the onboard camera used in real flights. The ground control station QGroundControl [26] was used to view the image and send commands to the simulated vehicle. The setup can be seen in Figure 3.6.



(a) Gazebo world with camera-equipped UAV and a visual marker.



(b) Simulated camera feed and vehicle controls in QGroundControl.

Figure 3.6: Image of test setup to determine visual marker configuration.

Prior to testing, it was clear that no single marker could be used for tracking at a distance and in proximity to the platform. At close range, larger markers would exceed the camera's FOV and smaller markers would not be detectable at range. The originally envisioned approach considered multiple adjacent markers placed in the unobstructed area of the landing platform. The initial simulator testing revealed three main flaws in this approach:

1. Multiple markers of different sizes placed adjacently are suboptimal for utilizing the available space of the landing platform.

2. The UAV must move horizontally as it switches from tracking one marker to the next. During this phase, it is more likely to lose track due to the roll and pitch motions causing the rigidly mounted camera to point away from the target.

3. The smallest marker used for close range tracking could not be located at the center of the platform, because the largest marker occupied a majority of the space. Effectively, this reduced the tolerances for landing precision because the UAV was forced to land closer to the perimeter of the platform.

To solve these issues, the third generation of AprilTag [4] with its custom and flexible layouts was used. The localization properties of the AprilTag markers are discussed in Section 3.3.1, but the geometry of the custom markers is relevant for the problem at hand. The custom48h12 family of tags have an empty space in the middle which does not affect detection and can be filled arbitrarily. This makes it possible to design recursive tags, as shown in Figure 2.6. The configuration solves all three problems stated above; it utilizes the entire available area of the landing platform, the UAV can track the center of the platform at all times and it is now possible to land at the platform center for maximum tolerance in positioning error.

3.3 Landing Target Detector

3.3.1 Fiducial Marker Geometry

The main task of the landing target detector is to process the raw image stream from the onboard camera and detect fiducial markers. The custom48h12 tags are square with a side length of ten bits. The innermost four bits of the marker are omitted and can be used for recursion. This means that the side length of each layer is five times smaller than the previous one. Considering a three layer design with an outter layer of $90\,\mathrm{cm}$ side length, the two next layers have side lengths of $18\,\mathrm{cm}$ and $3.6\,\mathrm{cm}$ respectively.

The maximum detection range of a tag, z_{max}, is given by

$$z_{max} = \frac{t}{2\tan\frac{F_x bp}{2W}} \tag{3.1}$$

where t is the tag width in meters, F_x is the horizontal FOV, b is the tag width in bits, W is the horizontal image size in pixels and p denotes the number of image pixels needed per tag bit for detection. p therefore serves as a tuning parameter which determines the robustness of detections. In cases of high viewing angles and low illumination, a higher value of p is needed. $p = 5$ is considered good for robust detection and $p = 2$ corresponds to the Nyquist frequency.

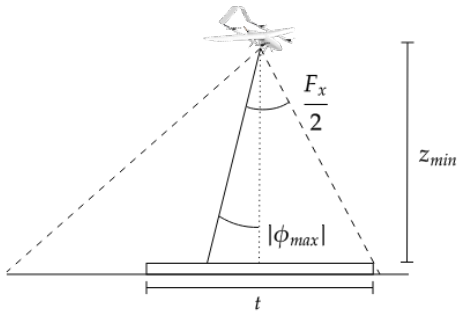


Figure 3.7: Visualization of the geometry for the minimum detection distance calculation, shown in a side view for perturbations in roll. The figure is not to scale.

An approximation of the minimum detection range of a tag, z_{min} was derived by the camera's FOV and expected perturbations in orientation of the UAV during the descent as shown in Figure 3.7:

$$z_{min} = \frac{t}{2\tan F'} \tag{3.2a}$$

$$F' = \min\left(\frac{F_x}{2} - |\phi_{max}|, \frac{F_y}{2} - |\theta_{max}|\right) \tag{3.2b}$$

The expression for F' given by Equation 3.2b is based on the minimum of the horizontal and vertical FOVs to ensure that z_{min} is not underestimated. The FOVs are adjusted for

the expected maximum roll and pitch movement expected during a descent, $\phi_{max} = 20°$ and $\theta_{max} = 12°$. The maximum perturbations in orientation are based on historical data for the UAV during multirotor descents in strong wind. The formula assumes the precision landing module is active and the camera is centered above visual marker with no positional offset in the horizontal plane. In practice there will always be some offset, but the formula is useful for building intuition for overlap of the detection distances. Additionally, vertical movement is not requested until sufficient horizontal alignment is achieved. This is further discussed in Section 3.5. Since the altitude of initialization is above the minimum detection distance, this assumption is considered fair.

	Maximum detection range, z_{max}				
Marker size, t	p=5	p=4	p=3	p=2	Minimum detection range, z_{min}
90 cm	10.30 m	12.89 m	17.18 m	25.78 m	1.24 m
18 cm	2.06 m	2.58 m	3.44 m	5.16 m	0.25 m
3.6 cm	0.41 m	0.52 m	0.69 m	1.03 m	0.05 m

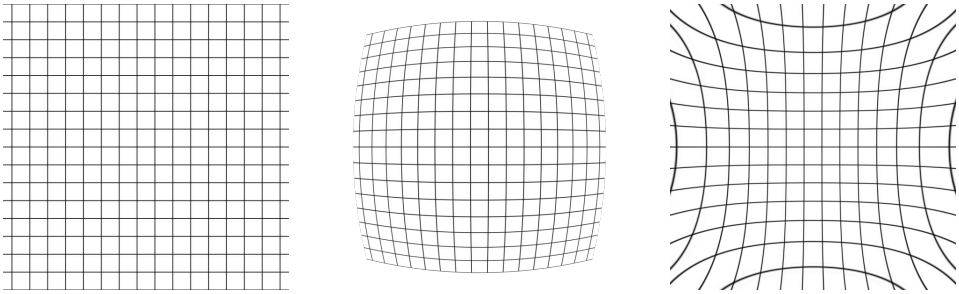
Table 3.1: Maximum and minimum detection ranges for different marker sizes and pixel per bit ratios. The values are based on Equations 3.1 and 3.2.

The UAV is equipped with landing legs, making the vertical distance from the onboard camera to the landing target 14 cm while grounded. As shown in Table 3.1, there is overlap of the detection distances for the three markers during the entire descent. Even in the most conservative case, the system is expected to perform contiguous detections all the way from 10 m AGL until the UAV is grounded. In practice, the overlap and maximum detection ranges are likely higher for two reasons; the UAV is equipped with a low-light camera which requires little illumination and the viewing angle is expected to approach zero as the UAV centers horizontally over the landing platform. Real-life tests were conducted to address these claims, which is discussed in Chapter 5.

3.3.2 Camera Calibration

In addition to the pixel coordinates obtained from detections of the fiducial markers, the camera intrinsics are necessary to calculate the pose. The intrinsic matrix is obtained obtained from a calibration scheme as discussed in Section 2.3.

The calibration was performed using the pinhole camera model. In reality, the camera aperture cannot be infinitely small as that would prevent any light from reaching the sensor. A lens is required to focus the light onto the sensor, which in turn contributes to some distortion effects. The distortion effect can be assumed to have a radial symmetry, meaning its value only depends on the distance from the principle point, i.e. $r(x', y') = \sqrt{(x' - c_x)^2 + (y' - c_y)^2}$. The distortion can be pointed outwards or inwards, which is referred to as barrel and pincushion distortion respectively. A visualization of this effect is shown in Figure 3.8. The effect of the distortion can be modelled in numerous ways. Brown's even ordered radial distortion model [27] is among the more commonly used distortion models.



(a) Reference orthogonal grid pattern.

(b) Visualization of barrel distortion.

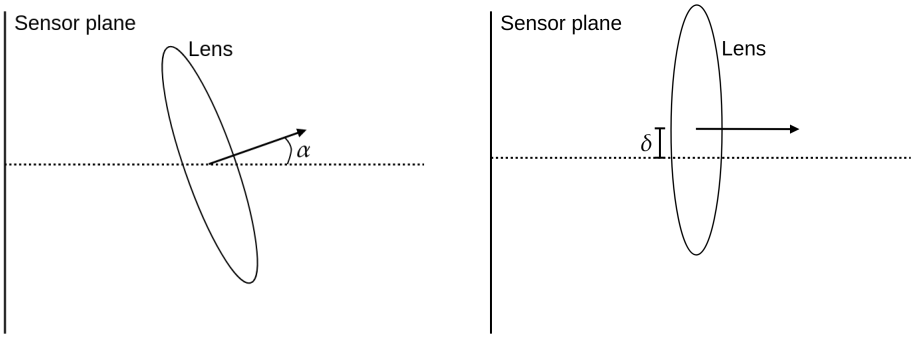
(c) Visualization of pincushion distortion.

Figure 3.8: Comparison between barrel and pincushion radial distortion effects of an orthogonal grid pattern. The distortion is exaggerated compared to expected results for illustrative purposes.

In this case, the two first radial distortion parameters were used:

$$\mathbf{x}'_{ru} = (1 + k_1 r^2 + k_2 r^4)\mathbf{x}'_{rd} \tag{3.3}$$

Where $\mathbf{x}'_{ru}$ are the radially undistorted pixel coordinates and $\mathbf{x}'_{rd}$ is the pixel coordinates subject to radial distortion. The distortion effect is usually more prevalent towards the edges of the image, which follows from Equation 3.3. For high-quality cameras with a low FOV, the distortion can sometimes be negligible. In this project, this is not the case as the camera lens provides a wide-angle view. This motivates the use of a radial distortion model, because it accommodates more accurate detections and pose estimates of targets in the camera's peripheral view.



(a) Illustration of camera lens not being parallel to image plane.

(b) Illustration of camera lens being offset relative to image plane center.

Figure 3.9: Illustration of a skewed and offset lens which introduces tangential distortion.

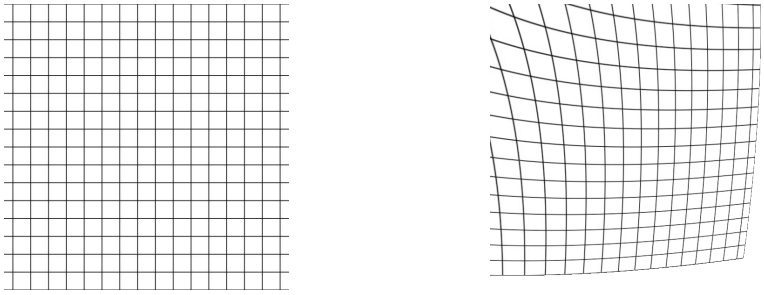
In addition to the radial distortion caused by lens imperfections, misalignment of the lens during manufacturing will also lead to distortion as shown in Figure 3.9. This type of tangential distortion occurs if the image plane is not parallel to the lens or if the lens is offset

from the image plane center. Even high-end cameras often have some degree of tangential distortion, but the effect is more visible in lower-end hardware. For these reasons, both the radial and tangential distortion models were used in this project to address some of the weaknesses for the pinhole camera model.

Tangential distortion differs from the radial distortion by not being symmetric and can be described by the following model [27]:

$$\mathbf{x}'_{tu} = \begin{bmatrix} 2p_1x'y' + p_2(r^2 + 2x'^2) \\ p_1(r^2 + 2y'^2) + 2p_2x'y' \end{bmatrix} \odot \mathbf{x}'_{td} \tag{3.4}$$

in which $\odot$ denotes the Hadamard product, $\mathbf{x}'_{tu}$ is the tangentially undistorted pixel coordinates and $\mathbf{x}'_{td}$ is the pixel coordinates subject to tangential distortion. The radial and tangential distortion can be introduced to the projection model in Equation 2.60 and their parameters can be estimated similarly to the scheme outlined in Section 2.3 given a sufficient number of calibration images to account for the extra DOF introduced by the distortion parameters.



(a) Reference orthogonal grid pattern. **(b)** Visualization of tangential distortion.

Figure 3.10: Visualization of tangential distortion of a reference orthogonal grid pattern. Unlike the radial distortion, the effect is not radially symmetric. The distortion is exaggerated compared to expected results for illustrative purposes.

The effect of tangential distortion on an image is shown in Figure 3.10. A second order approximation of the radial and tangential models were used, as the provided accuracy is sufficiently high for this application. For computer vision applications with very high accuracy requirements, higher degree polynomials or a different model entirely may be required.

3.3.3 Relative Pose Measurement

The UAV is equipped with a downward-facing low-light camera, which is used to detect the fiducial markers. To obtain detections reliably in various lighting conditions, an increased pixel size is desirable for high quantum efficiency. In this thesis, the Sony Exmor IMX323 based camera from Blue Robotics [28] was used. It has a pixel size of $2.8\,\mu\mathrm{m}$ which is relatively high for a $2\,\mathrm{Mpx}$ camera. The camera is light weight at only $17\,\mathrm{g}$, but can provide a high definition image with good low-light performance at a low cost.

To maximize low-light performance, the camera was setup to use binning; an artificial increase in pixel size by merging sensor data from adjacent pixels. This decreases the overall image size in terms of pixels, reducing the maximum detection range. The calculations from Table 3.1 were computed for the camera in binning configuration, meaning it does not compromise on the detection range over the intended flight pattern. The decreased image size reduces the computational load for the onboard companion computer per image, allowing the detections to be performed at a higher rate. This is beneficial for consistency, which is discussed in Section 3.4.

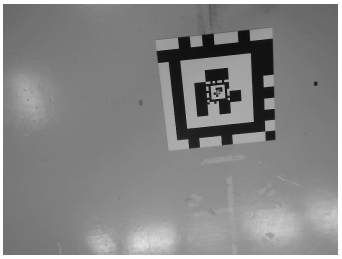
The corners of the fiducial marker are detected as a multi-stage, sequential process [4]:

1. The input Red Green Blue (RGB) image is converted to grayscale.
2. The image is downsampled, while maintaining the original FOV. This step is optional, but can be done to reduce the computational load and speed up the detection.
3. The image is thresholded.
4. Pixel unions are computed with a union-find algorithm based on gradient magnitude and direction for each pixel. The output is used to determine connected components in the thresholded image.
5. The boundaries between dominating black and white components are found by segmentation, similar to the graph-based method described in [29]. The outputs serve as candidates for the tag contours. In general, this is the slowest part of the detection algorithm.
6. The boundaries are searched for quadrilateral shapes which are fitted to the contours.
7. A perspective correction is applied. The homography is estimated using an approach similar to the one discussed in Section 2.3.1.
8. Sharpening and decoding of data bits.

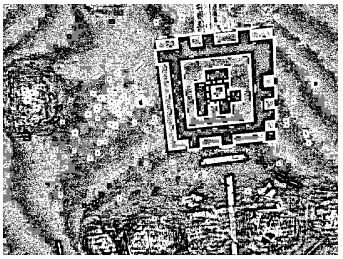
The output of individual detection steps are shown in Figure 3.11 for a sample image of the landing target during a test flight. In this example, the two outermost markers of the recursive target are correctly identified. The innermost tag is only populated by 10×10 pixels, which corresponds to $p = 1$ in Equation 3.1 since the marker is only 10 bits wide. This is below the minimum value for reliable detection and explains why only the two larger tags are found.

The detection corners define the edges between the black and the white borders inside the tag as shown in Figure 3.12. Using the codeword of the detected tag's data bits, its scale is determined based on the known physical configuration. Since the points are coplanar and the detection corners are zero centered, the detected points can the be used to obtain a pose measurement using the IPPE algorithm given the camera intrinsics. In IPPE, a transform is applied around an infinitesimally small region on the target's surface to determine its pose.

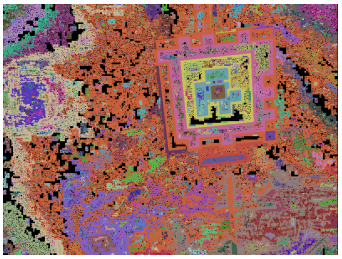
The following derivation aims to outline the steps of the IPPE method and how they result in a completely analytical expression for the relative pose. Further details of the individual



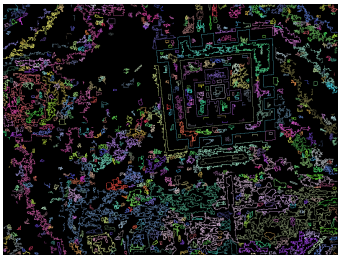
(a) Grayscale conversion of input image and decimation.



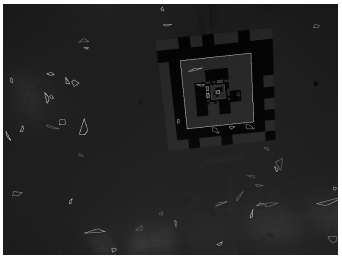
(b) Individual pixels are thresholded by value.



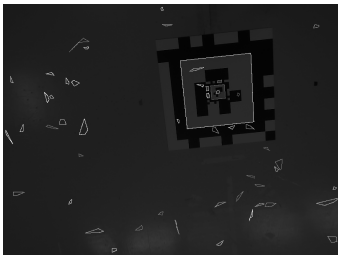
(c) Connected components in the image are identified and grouped.



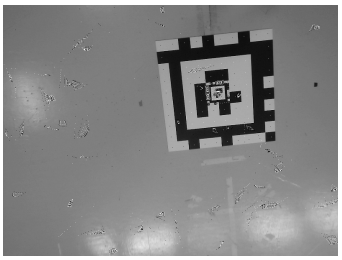
(d) Segmented areas are used to define contours between dark and light areas.



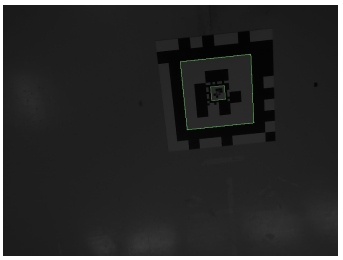
(e) Tag candidates are identified based on contours.



(f) Quadrilaterals are fitted to the tag candidates.



(g) Perspective correction and sampling of data bits.



(h) Output of the tags detected in the image.

Figure 3.11: Visualization of the AprilTag 3 detection steps.

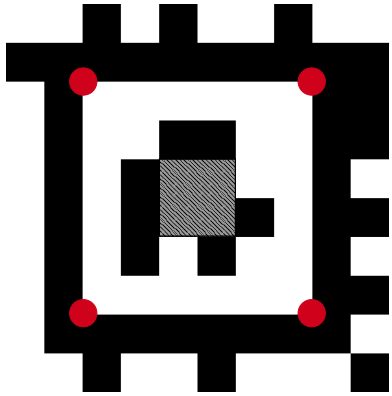


Figure 3.12: Detection corners visualized in red on an instance of the AprilTag custom48h12 family.

steps are provided by [23]. Consider a set of n points in the model plane of the fiducial marker $\mathbf{u}_k \in \mathbb{R}^2,\ k \in [1, n]$. The points are zero centered, $\sum_1^n \mathbf{u}_k = \mathbf{0}$. Using the fiducial marker shown in Figure 3.12, the model plane points span a square with $n = 4$. Each point in the model plane has a known corresponding point $\mathbf{q}_k \in \mathbb{R}^2$ in the camera's image. The camera intrinsic matrix $\mathbf{K}$ is known.

The normalized image coordinates are obtained by $[\tilde{\mathbf{q}}_k^\top, 1]^\top = \mathbf{K}^{-1}[\mathbf{q}_k^\top, 1]^\top$. The best fitting homography $\mathbf{H}$ up to noise is then computed between $\{\tilde{\mathbf{q}}_k\}$ and $\{\mathbf{u}_k\}$ and $\mathbf{H}$ is rescaled to ensure that $H_{33} = 1$. The Jacobian of the plane-to-image function $\boldsymbol{\pi}(\cdot)$ is then calculated at $\mathbf{u} = \mathbf{0}$,

$$\mathbf{J} = \begin{bmatrix} H_{11} - H_{31}H_{13} & H_{12} - H_{32}H_{13} \\ H_{21} - H_{31}H_{23} & H_{22} - H_{32}H_{23} \end{bmatrix} \tag{3.5}$$

We then let $\mathbf{v} = \boldsymbol{\pi}(\mathbf{H}[\mathbf{0}^\top, \mathbf{1}]^\top) = [H_{13}, H_{23}]^\top$, i.e. the 2D point in the image where the marker center is located, expressed in normalized coordinates. By defining

$$\cos\theta = \frac{1}{\|[\mathbf{v}^\top, 1]^\top\|_2} \tag{3.6a}$$

$$\sin\theta = \sqrt{1 - \frac{1}{\|[\mathbf{v}^\top, 1]^\top\|_2^2}} \tag{3.6b}$$

$$[\mathbf{k}]_\times = \frac{1}{\|\mathbf{v}\|_2} \begin{bmatrix} \mathbf{0} & \mathbf{v} \\ -\mathbf{v}^\top & 0 \end{bmatrix} \tag{3.6c}$$

we can use Rodrigues formula to compute $\mathbf{R}_v = \mathbf{I} + \sin\theta[\mathbf{k}]_\times + (1 - \cos\theta)[\mathbf{k}]_\times^2$, which is the smallest possible rotation aligning $[\mathbf{v}^\top 1]$ with the z-axis. Given $\mathbf{v}$ and $\mathbf{R}_v$, we can now compute $[\mathbf{B}|\mathbf{0}] = [\mathbf{I}| - \mathbf{v}]\mathbf{R}_v$ and $\mathbf{A} = \mathbf{B}^{-1}\mathbf{J}$.

Let

$$\mathbf{A}\mathbf{A}^\top = \begin{bmatrix} a_1 & a_2 \\ a_2 & a_3 \end{bmatrix} \tag{3.7}$$

such that

$$\gamma = \sigma_1^A = \sqrt{\frac{1}{2}(a_1 + a_3 + \sqrt{(a_1 - a_3)^2 + 4a_2^2})} \tag{3.8}$$

is the largest singular value of $\mathbf{A}$. There are two possible solutions $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$ for the rotational part of the relative pose between the marker and the camera,

$$\mathbf{R}_1 = \mathbf{R}_v \tilde{\mathbf{R}}_1 \tag{3.9a}$$

$$\mathbf{R}_2 = \mathbf{R}_v \tilde{\mathbf{R}}_2 \tag{3.9b}$$

The matrices $\tilde{\mathbf{R}}_1$ and $\tilde{\mathbf{R}}_2$ can be found by

$$\tilde{\mathbf{R}}_1 = \begin{bmatrix} \tilde{\mathbf{R}}_{22} & \mathbf{c} \\ \mathbf{b}^\top & a \end{bmatrix} \tag{3.10a}$$

$$\tilde{\mathbf{R}}_2 = \begin{bmatrix} \tilde{\mathbf{R}}_{22} & -\mathbf{c} \\ -\mathbf{b}^\top & a \end{bmatrix} \tag{3.10b}$$

where

$$\tilde{\mathbf{R}}_{22} = \frac{1}{\gamma}\mathbf{A} \tag{3.11a}$$

$$\mathbf{b}^\top \mathbf{b} = \tilde{\mathbf{R}}_{22}^\top \tilde{\mathbf{R}}_{22} \tag{3.11b}$$

$$\begin{bmatrix} \mathbf{c} \\ a \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{R}}_{22} \\ \mathbf{b}^\top \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} \tilde{\mathbf{R}}_{22} \\ \mathbf{b}^\top \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.11c}$$

In Equation 3.11, $\mathbf{b}$ can be obtained by the rank-1 decomposition of $\mathbf{I} - \tilde{\mathbf{R}}_{22}^\top \tilde{\mathbf{R}}_{22}$.

In theory, the translational part of the relative pose could be obtained through $\mathbf{T}_i = \frac{1}{\gamma}\begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} - \mathbf{R}_i \begin{bmatrix} \mathbf{u}_0 \\ 0 \end{bmatrix}, i \in \{1, 2\}$. Since $\mathbf{v}$ was computed at $\mathbf{u}_0 = \mathbf{0}$, only the first term is used. However, $\mathbf{v}$ was also computed using the homography $\mathbf{H}$ which is subject to noise. Therefore, a more accurate estimate is found by using the solution for $\mathbf{R}$ and linear least squares regression with the cost function to be minimized,

$$\sum_{i=1}^{n} \left\| \mathbf{R}_{22}\mathbf{u}_i + \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - (\mathbf{s}_3 \begin{bmatrix} \mathbf{u}_i \\ 0 \end{bmatrix} + T_3)\tilde{\mathbf{q}}_i \right\|_2^2 \tag{3.12}$$

in which $\mathbf{s}_3$ is the third row in $\mathbf{R}$. Since the error is minimized in the 3D camera space and not the 2D image space, the problem is convex and the solution $\mathbf{T}$ will thus be a global minimum. This can be solved very efficiently, by rewriting the functional to be minimized on the form

$$\|\mathbf{W}_j \mathbf{T}_j - \mathbf{b}_j\|_2^2, \quad j \in 1, 2 \tag{3.13}$$

such that the minima is located at $\mathbf{T}_j^\star = (\mathbf{W}_j^\top \mathbf{W}_j)^{-1}\mathbf{W}_j \mathbf{b}_j$. The solution is unique because $\mathbf{W}_j$ has rank 3. Since $\mathbf{W}_j$ is $2n \times 3$, $\mathbf{W}_j^\top \mathbf{W}_j$ is 3×3 and its inverse can be computed with low computational cost. For a coplanar square object defined by its four corners, the homography $\mathbf{H}$ is computed analytically.

Looking at the steps above, the IPPE method as a hole will then compute the relative pose completely analytically, which makes it extremely fast. This is further discussed in Section 5.2. The underlying geometry of this problem makes it prone to ambiguity, which cannot be accounted for by any PnP method alone. In general, IPPE returns two solutions. The solution associated with the lowest reprojection error can be chosen in most cases. In some configurations, the reprojection error is an ambiguous metric due to noise, especially when viewed from a distance or if the marker is small. In this scenario, the projection of the marker appears affine and the ambiguity arises due to symmetry around the axis from the camera's center to the marker's center [23]. Both of these conditions may apply for the intended trajectory, meaning other measures must be taken to resolve the ambiguity. In the proposed system, pose measurements inconsistent with the current pose estimate or too different from previous measurements are discarded.

3.4 Invariant Filtering

The pose measurements discussed in Section 3.3.3 are used to determine relative height, yaw alignment and positional errors in the horizontal plane during the descent, which the autopilot can use to align the UAV with the center of the landing platform. The raw pose measurements are subject to noise and would result in a trajectory with high-frequency components the vehicle is unable to follow. Additionally, raw measurements are prone to dropouts and ambiguity with no sense of uncertainty associated to the believed pose. To address this, the UAV's pose is estimated with an IEKF. In addition to effectively smoothing the trajectory, the filter can predict the UAV pose in the event of missing detections for a short time period and the filter's innovation and estimated state along with their respective covariances can be used to assess consistency and quality of the estimates in real time.

3.4.1 Bias-Free Motion Model

First, we consider the motion model based on a bias-free IMU. In this case, the IEKF represents the state space as an extended pose, using the $SE_2(3)$ double direct spatial isometries. The IMU measurements of acceleration and angular velocity are assumed to be corrupted with zero-mean Gaussian noise,

$$\tilde{\mathbf{a}}_t = \mathbf{a}_t + \mathbf{a}_{nt}, \qquad \mathbf{a}_{nt} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_a) \tag{3.14a}$$

$$\tilde{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t + \boldsymbol{\omega}_{nt}, \qquad \boldsymbol{\omega}_{nt} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\omega) \tag{3.14b}$$

where $(\cdot)_t$ denotes the true value, $(\cdot)_{nt}$ denotes the noise and $\tilde{(\cdot)}_t$ denotes the measured value at time t. Using Equation 3.14, the system dynamics in continuous time are

$$\dot{\mathbf{R}}_t = \mathbf{R}_t[\tilde{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_{nt}]_\times \tag{3.15a}$$

$$\dot{\mathbf{v}}_t = \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{a}_{nt}) + \mathbf{g} \tag{3.15b}$$

$$\dot{\mathbf{p}}_t = \mathbf{v}_t \tag{3.15c}$$

in which $\mathbf{g}$ is the gravity vector, which is assumed to be known [6]. In Equations 3.15, the IMU measurements are used as input, $\mathbf{u}_t$, to the system. The $SE_2(3)$ Lie group is

particularly useful in inertially driven estimation problems, because it involves using sensor data from gyroscopes and accelerometers to evolve the orientation and velocity states. This will in turn propagate into the position component. By using the IMU measurements as input to the system, one avoids the need to model the dynamics of the acceleration, i.e. the jerk of the system. The matrix representation of the dynamics as an $SE_2(3)$ extended pose is then

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{X}_t &= \begin{bmatrix}\mathbf{R}_t[\tilde{\boldsymbol{\omega}}_t]_\times & \mathbf{R}_t\tilde{\mathbf{a}}_t + \mathbf{g} & \mathbf{v}_t \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0\end{bmatrix} \\ &\quad - \begin{bmatrix}\mathbf{R}_t & \mathbf{v}_t & \mathbf{T}_t \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1\end{bmatrix}\begin{bmatrix}[\boldsymbol{\omega}_{nt}]_\times & \mathbf{a}_{nt} & 0 \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0\end{bmatrix} \\ &= \mathbf{f}(\mathcal{X}_t, \mathbf{u}_t) - \mathcal{X}_t\mathbf{w}_t^\wedge\end{aligned} \tag{3.16}$$

where the process noise $\mathbf{w}_t = \begin{bmatrix}\boldsymbol{\omega}_{nt}^\top & \mathbf{a}_{nt}^\top & \mathbf{0}^\top\end{bmatrix}^\top$ has covariance $\mathbf{Q}$ with diagonal elements $\boldsymbol{\Sigma}_\omega, \boldsymbol{\Sigma}_a, \mathbf{0}_{3\times 3}$. Together with the discrete time transition matrix $\boldsymbol{\Phi}_k$, the covariance is propagated as described in Equations 2.50 and 2.51.

To prove the group affine property, consider

$$\mathcal{X}_a = \begin{bmatrix}\mathbf{R}_a & \mathbf{v}_a & \mathbf{T}_a \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1\end{bmatrix} \tag{3.17a}$$

$$\mathcal{X}_b = \begin{bmatrix}\mathbf{R}_b & \mathbf{v}_b & \mathbf{T}_b \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1\end{bmatrix} \tag{3.17b}$$

such that

$$\mathbf{f}(\mathcal{X}_a, \mathbf{u}) = \begin{bmatrix}\mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{R}_a\tilde{\mathbf{a}} + \mathbf{g} & \mathbf{v}_a \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0\end{bmatrix} \tag{3.18a}$$

$$\mathbf{f}(\mathcal{X}_b, \mathbf{u}) = \begin{bmatrix}\mathbf{R}_b[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{R}_b\tilde{\mathbf{a}} + \mathbf{g} & \mathbf{v}_b \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0\end{bmatrix} \tag{3.18b}$$

The identity function is then

$$\mathbf{f}(\mathbf{I}, \mathbf{u}) = \begin{bmatrix}\mathbf{I}[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{I}\tilde{\mathbf{a}} + \mathbf{g} & \mathbf{0}^\top \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0\end{bmatrix} \tag{3.19}$$

Next, we compute $\mathbf{f}(\mathcal{X}_a\mathcal{X}_b, \mathbf{u}), \mathcal{X}_a\mathbf{f}(\mathcal{X}_b, \mathbf{u}), \mathbf{f}(\mathcal{X}_a, \mathbf{u})\mathcal{X}_b$ and $\mathcal{X}_a\mathbf{f}(\mathbf{I}, \mathbf{u})\mathcal{X}_b$,

$$\mathcal{X}_a\mathcal{X}_b = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b & \mathbf{R}_a\mathbf{v}_b + \mathbf{v}_a & \mathbf{R}_a\mathbf{T}_b + \mathbf{T}_a \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \tag{3.20a}$$

$$\mathbf{f}(\mathcal{X}_a\mathcal{X}_b, \mathbf{u}) = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{R}_a\mathbf{R}_b\tilde{\mathbf{a}} + \mathbf{g} & \mathbf{R}_a\mathbf{v}_b + \mathbf{v}_a \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \tag{3.20b}$$

$$\mathcal{X}_a\mathbf{f}(\mathcal{X}_b, \mathbf{u}) = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{R}_a\mathbf{R}_b\tilde{\mathbf{a}} + \mathbf{R}_a\mathbf{g} & \mathbf{R}_a\mathbf{v}_b \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \tag{3.20c}$$

$$\mathbf{f}(\mathcal{X}_a, \mathbf{u})\mathcal{X}_b = \begin{bmatrix} \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{R}_b & \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{v}_b + \mathbf{R}_a\tilde{\mathbf{a}} + \mathbf{g} & \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{T}_b + \mathbf{v}_a \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \tag{3.20d}$$

$$\begin{aligned} \mathcal{X}_a\mathbf{f}(\mathbf{I}, \mathbf{u})\mathcal{X}_b &= \begin{bmatrix} \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times & \mathbf{R}_a\tilde{\mathbf{a}} + \mathbf{R}_a\mathbf{g} & \mathbf{0}^\top \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_b & \mathbf{v}_b & \mathbf{T}_b \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{R}_b & \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{v}_b + \mathbf{R}_a\tilde{\mathbf{a}} + \mathbf{R}_a\mathbf{g} & \mathbf{R}_a[\tilde{\boldsymbol{\omega}}]_\times\mathbf{T}_b + \mathbf{v}_a \\ \mathbf{0}^\top & 0 & 0 \\ \mathbf{0}^\top & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.20e}$$

From the Equations 3.20, it is clear that $\mathbf{f}(\mathcal{X}_a\mathcal{X}_b, \mathbf{u}) = \mathcal{X}_a\mathbf{f}(\mathcal{X}_b, \mathbf{u}) + \mathbf{f}(\mathcal{X}_a, \mathbf{u})\mathcal{X}_b - \mathcal{X}_a\mathbf{f}(\mathbf{I}, \mathbf{u})\mathcal{X}_b$ and the group affine property defined in Equation 2.44 is satisfied. The dynamics in Equation 3.16 can therefore be shown to have state independent error trajectories with provable local stability around any trajectory as discussed in Section 2.2.3.

The expression for η_t can be approximated as

$$\eta_t = \exp(\xi_t) \approx \mathbf{I} + \xi_t^\wedge \tag{3.21}$$

meaning the right invariant formulation[1] of 2.45b can be rewritten to

$$\begin{aligned} \mathbf{g}(\mathbf{I} + \xi_t^\wedge, \mathbf{u}_t) &= (\mathbf{A}_t\xi_t)^\wedge \\ \mathbf{A}_t &= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ [\mathbf{g}]_x & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I} & \mathbf{0}_{3\times3} \end{bmatrix} \end{aligned} \tag{3.22}$$

according to [30]. In the bias-free model, the matrix $\mathbf{A}_t$ is constant for a larger set of trajectories in the right IEKF unlike the left invariant formulation which can be derived similarly. This is a benefit when performing predictions at a high rate, e.g. when using an IMU because a recalculation at each prediction increases the computational load. In

[1] $\mathbf{g}$ denotes the gravitational vector in $\mathbf{A}_t$.

practice, the adjoint matrix can be utilized to transform between the two representations,

$$\xi^r = \mathbf{Ad}_{\hat{\mathcal{X}}}\xi^l \tag{3.23a}$$
$$\mathbf{P}^r = \mathbf{Ad}_{\hat{\mathcal{X}}}^{\top}\mathbf{P}^l\mathbf{Ad}_{\hat{\mathcal{X}}} \tag{3.23b}$$
$$\tag{3.23c}$$

For this reason, the right invariant formulation of the IEKF is used unless stated otherwise.

3.4.2 Estimating IMU Biases

In practice, the biases of the IMU must be estimated to perform reliable tracking of the true trajectory. Unfortunately, the bias does not fit into the framework of Lie Groups like the extended pose. It is therefore included as a state augmentation, resulting in an imperfect IEKF. The log-linear property has not been demonstrated in this case, meaning convergence around any trajectory cannot be proven like in the conventional IEKF formulation. The biases are usually represented with Brownian Motion, for which the dynamics are modelled as a random walk. Since the bias dynamics are slow compared to the vehicle dynamics, the coupling between the augmented states and the extended pose is in general weak. Although the attractive mathematical proofs no longer hold, the imperfect IEKF still outperforms the EKF in practice.

The IMU measurements and biases are now described by

$$\tilde{\mathbf{a}}_t = \mathbf{a}_t + \mathbf{b}_t^a + \mathbf{a}_{nt}, \qquad \mathbf{a}_{nt} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_a) \tag{3.24a}$$
$$\tilde{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t + \mathbf{b}_t^\omega + \boldsymbol{\omega}_{nt}, \qquad \boldsymbol{\omega}_{nt} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\omega) \tag{3.24b}$$
$$\dot{\mathbf{b}}_t^a = \mathbf{b}_{nt}^a, \qquad \mathbf{b}_{nt}^a \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{ba}) \tag{3.24c}$$
$$\dot{\mathbf{b}}_t^\omega = \mathbf{b}_{nt}^\omega, \qquad \mathbf{b}_{nt}^\omega \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{b\omega}) \tag{3.24d}$$

The continuous-time process noise with the augmented state is $\mathbf{w}_t = \begin{bmatrix}\boldsymbol{\omega}_{nt}^\top & \mathbf{a}_{nt}^\top & \mathbf{0}^\top & \mathbf{b}_{nt}^{\omega\top} & \mathbf{b}_{nt}^{a\top}\end{bmatrix}^\top$ using the bias-enabled IMU dynamics, with the diagonal elements of the covariance matrix $\mathbf{Q}$ following the same augmented structure.

By inserting Equations 3.24 into the dynamics of Equation 3.16 and linearizing using standard Euler integration, the estimated discrete-time dynamics for the augmented extended pose can be derived,

$$\hat{\mathbf{R}}_k = \hat{\mathbf{R}}_{k-1}\exp((\tilde{\boldsymbol{\omega}}_{k-1} - \hat{\mathbf{b}}_{k-1}^\omega)h) \tag{3.25a}$$
$$\hat{\mathbf{v}}_k = \hat{\mathbf{v}}_{k-1} + \hat{\mathbf{R}}_{k-1}(\tilde{\mathbf{a}}_{k-1} - \hat{\mathbf{b}}_{k-1}^a)h + \mathbf{g}h \tag{3.25b}$$
$$\hat{\mathbf{p}}_k = \hat{\mathbf{p}}_{k-1} + \hat{\mathbf{v}}_{k-1}h + \frac{1}{2}\hat{\mathbf{R}}_{k-1}(\tilde{\mathbf{a}}_{k-1} - \hat{\mathbf{b}}_{k-1}^a)h^2 + \frac{1}{2}\mathbf{g}h^2 \tag{3.25c}$$
$$\hat{\mathbf{b}}_k^a = \hat{\mathbf{b}}_{k-1}^a \tag{3.25d}$$
$$\hat{\mathbf{b}}_k^\omega = \hat{\mathbf{b}}_{k-1}^\omega \tag{3.25e}$$

in which h is the time step duration. The linearization is performed under the assumption of constant IMU measurements over the sample time. For this assumption to remain valid,

the IMU's sample rate must be sufficiently high relative to the dynamics of the vehicle. To arrive at the matrix $\mathbf{A}_t$, the same approach shown in Equation 3.22 of the bias-free motion model can be applied.

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\hat{\mathbf{R}}_t & \mathbf{0}_{3\times3} \\ [\mathbf{g}]_x & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -[\hat{\mathbf{v}}]_x\hat{\mathbf{R}}_t & -\hat{\mathbf{R}}_t \\ \mathbf{0}_{3\times3} & \mathbf{I} & \mathbf{0}_{3\times3} & -[\hat{\mathbf{T}}]_x\hat{\mathbf{R}}_t & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{3.26}$$

Due to the addition of biases as a state augmentation, the IEKF correction applies the Kalman gain in two steps; for an invariant error defined on the Lie Group and a bias error. The invariant errors use the definitions from Equation 2.43 with the exponential mapping between errors defined in Equation 2.48. The error is used to update the state estimate using the matrix exponential and the associated Kalman gain is denoted $\mathbf{K}_\xi$. In the second case, the biases are updated using a linear sum of the vectors like in a conventional EKF with Kalman gain $\mathbf{K}_\zeta$ such that $\mathbf{K} = [\mathbf{K}_\xi \quad \mathbf{K}_\zeta]^\top$. The adjoint matrix is also modified to represent the augmented states: $\mathbf{Ad}_\mathcal{X} = \text{diag}(\mathbf{Ad}_{SE_2(3)}, \mathbf{I}, \mathbf{I})$.

3.4.3 Observation Models

The IEKF uses the measurements of the marker relative to the UAV to perform a full six DOF pose update. In principle, one could also fuse the GNSS measurements in the IEKF. This was not implemented because the precision landing is only concerned with local positioning, and the added complexity would likely not be justifiable considering the high uncertainty of GNSS compared to the fiducial marker-based pose measurements. Additionally, the proposed architecture is more flexible in terms of future use cases as it does not rely on external positioning services and could e.g. be used in GNSS-denied environments. When estimating the relative pose, the yaw of the UAV is estimated with higher certainty than e.g. a magnetometer which the autopilot uses, improving the alignment of heading between platform and UAV during the landing sequence.

By tracking the landing platform center using the recursive arrangement as shown in Figure 2.6b instead of adjacent markers, the marker center is always aligned with the platform center. This removes the need to perform a rigid body transform to the platform center depending in which marker currently being detected. Therefore, the update model is simplified and consistent for all three markers. First, consider the simplified IEKF update for position only. For a given position $\mathbf{p}_t$, the noisy measured position will be of the form $\mathbf{y}_k^p = \mathbf{p}_k + \mathbf{n}_k^p$ which intrinsically is a left invariant observation. The update can be performed according to Equation 2.49b with $\mathbf{d}^p = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^\top$ corresponding to the last column in the unaugmented state matrix. For reasons discussed in Section 3.4.1, it is desirable to use the right invariant formulation. This can be achieved by transforming the resulting measurement matrix using the adjoint; $\mathbf{H}_p^r = \mathbf{H}_p^l \mathbf{Ad}_{\hat{\mathcal{X}}}^{-\top}$.

For the full pose update, the measurement matrix is computed according to the following

ideal measurement model of the unaugmented state $\mathcal{X}_k \in SE_2(3)$:

$$\mathcal{Y}_k = \Phi(\mathcal{X}_k)\mathbf{d} \tag{3.27}$$

In this derivation, $\mathcal{Y}_k, \mathbf{d} \in SE(3)$ because the linearized model used for pure position updates falls short when dealing with the rotational component of the pose measurement. The group homomorphism Φ is given by

$$\Phi: \quad SE_2(3) \to SE(3) \quad ; \quad \begin{bmatrix} \mathbf{R} & \mathbf{v} & \mathbf{T} \\ \mathbf{0}^\top & 1 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.28}$$

According to [31], Equation 3.27 with the addition of noise can be rewritten as a group action

$$\mathcal{Y}_k = (\mathcal{X}_k \cdot \mathbf{d}) \oplus \mathbf{n}_k \tag{3.29}$$

where $\mathbf{n}_k \in \mathbb{R}^6$ is additive Gaussian noise with covariance $\mathbf{R}_k$. We can now define the innovation as

$$\nu_k = (\hat{\mathcal{X}}_k^{-1}\mathcal{Y}_k) \ominus \mathbf{d} \tag{3.30}$$

Recall from Equation 2.4, that the result of a $\ominus$-composition is an element in $\mathbb{R}^6$ in the case of $SE(3)$ composition. Thus, the expression of Equation 3.30 can replace the term multiplied with the Kalman gain in Equation 2.50c. In the case of ideal measurements, subject to no noise, $\hat{\mathcal{X}}_k^{-1}\mathcal{Y}_k = \hat{\mathcal{X}}_k^{-1}\mathcal{X}_k\mathbf{d} = \eta_k\mathbf{d}$. In the event that $\xi_k = \mathbf{0}$, $\eta_k = \exp(\xi_k)$ is at the identity and $\hat{\mathcal{X}}_k^{-1}\mathcal{Y}_k = \mathbf{d}$. The composition of $\ominus\mathbf{d}$ in Equation 3.30 will then ensure that the innovations are of zero mean. The Jacobian $\frac{\mathrm{D}\Phi(\eta_k)}{\mathrm{D}\eta_k}$ is constant,

$$\frac{\mathrm{D}\Phi(\eta_k)}{\mathrm{D}\eta_k} = \frac{\mathrm{D}\Phi(\eta)}{\mathrm{D}\eta} = \begin{bmatrix} \mathbf{I} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I} \end{bmatrix} \tag{3.31}$$

such that

$$\begin{aligned} \frac{\mathrm{D}\eta_k \cdot \mathbf{d}}{\mathrm{D}\eta_k} &= \frac{\mathrm{D}\Phi(\eta_k)\mathbf{d}}{\mathrm{D}\eta_k} \\ &= \frac{\mathrm{D}\Phi(\eta_k)\mathbf{d}}{\mathrm{D}\Phi(\eta_k)} \frac{\mathrm{D}\Phi(\eta)}{\mathrm{D}\eta} = \mathbf{Ad}_{\mathbf{d}^{-1}} \frac{\mathrm{D}\Phi(\eta)}{\mathrm{D}\eta} \end{aligned} \tag{3.32}$$

using the chain rule and the following property for composition of Jacobians[2]:

$$\begin{aligned} \frac{\mathrm{D}\mathcal{X}\cdot\mathcal{Y}}{\mathrm{D}\mathcal{X}} &= \lim_{\boldsymbol{\tau}\to\mathbf{0}} \frac{\mathrm{Log}((\mathcal{X}\mathcal{Y})^{-1}(\mathcal{X}\mathrm{Exp}(\boldsymbol{\tau})\mathcal{Y}))}{\boldsymbol{\tau}} \\ &= \lim_{\boldsymbol{\tau}\to\mathbf{0}} \frac{\mathcal{Y}^{-1}\mathrm{Exp}(\boldsymbol{\tau})\mathcal{Y}}{\boldsymbol{\tau}} \\ &= \lim_{\boldsymbol{\tau}\to\mathbf{0}} \frac{(\mathcal{Y}^{-1}\boldsymbol{\tau}^\wedge\mathcal{Y})^\vee}{\boldsymbol{\tau}} \\ &= \mathbf{Ad}_{\mathcal{Y}}^{-1} = \mathbf{Ad}_{\mathcal{Y}^{-1}} \end{aligned} \tag{3.33}$$

[2]For more comprehensive details on the steps in this derivation, the reader is referred to [5].

The measurement matrix for the full six DOF pose update is then

$$\mathbf{H}_k = \mathbf{J}_r^{-1}(\mathbf{z}_k)\frac{\mathrm{D}\eta \cdot \mathbf{d}}{\mathrm{D}\eta} \tag{3.34}$$

where $\mathbf{J}_r$ is the inverse right Jacobian as defined in Appendix A.2.

3.5 Autopilot Interface

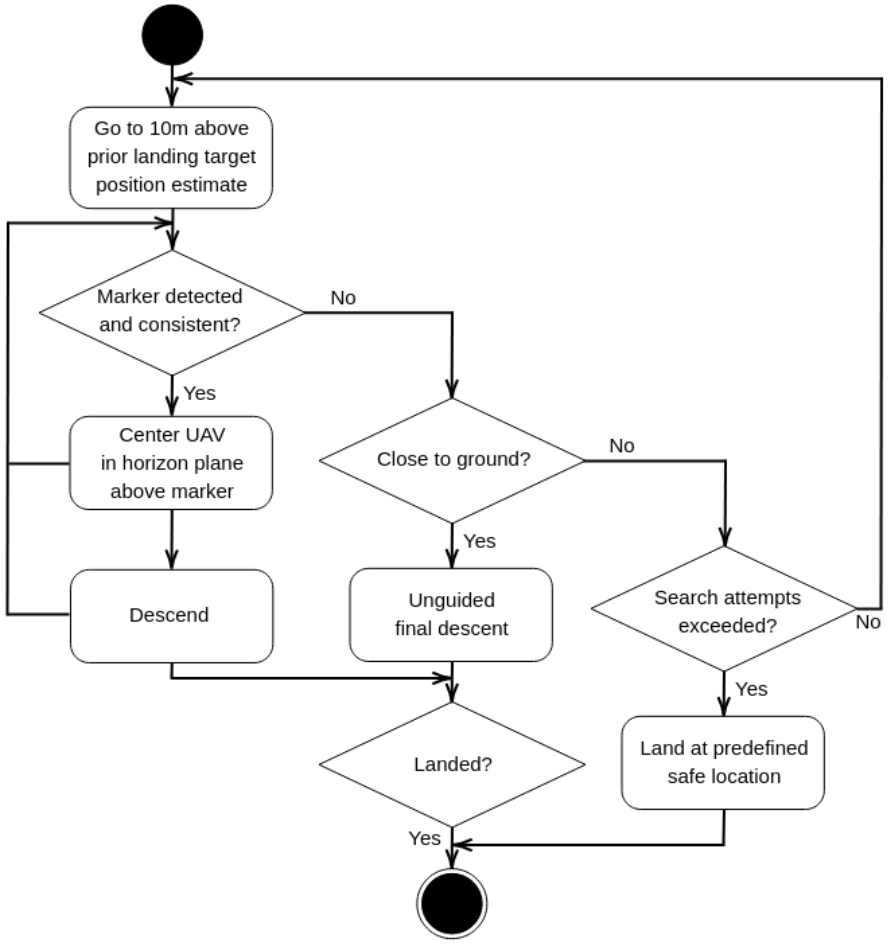


Figure 3.13: High-level flow diagram of autopilot interface finite-state machine.

The estimation of relative pose between UAV and landing target is performed separately from the autopilot's primary estimator used for navigation and control. This is partly done because the precision landing module is designed to be flexible and agnostic of autopilot

choice, but it also makes it easier to configure safety measures to prevent the landing target estimator from causing potentially dangerous behaviour of the UAV. The autopilot interface is based on a finite-state machine, of which the logic flow is shown in Figure 3.13.

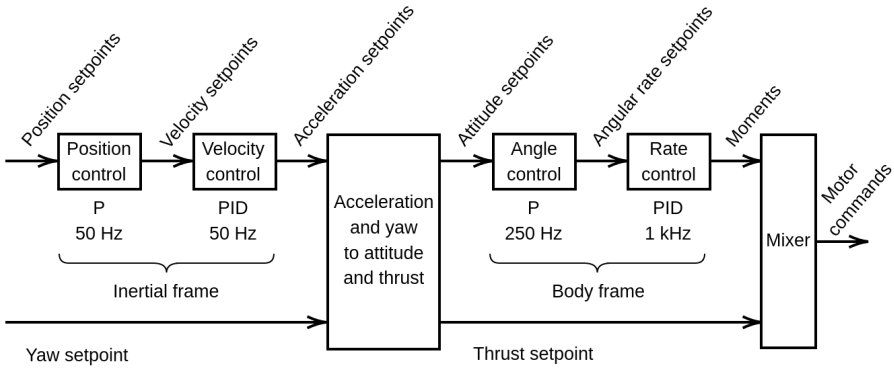


Figure 3.14: Overview of the cascaded control architecture used by the multirotor position controller in PX4. Adapted from [1].

Initially, the UAV moves to a predefined location where it is expected to be able to visually detect the landing target. The invariant filter then initializes and waits for convergence, receiving IMU messages from the autopilot and images from the onboard camera. The UAV is then commanded to move horizontally at the current altitude towards the center of the platform before it starts to descend using the cascaded control architecture shown in Figure 3.14. The precision landing system, distinct from the navigation and control system of the primary autopilot, is focused on guiding the aircraft during the landing process. In essence, the standalone module of the precision landing system can be implemented across various autopilots, regardless of the specific techniques employed for navigation and control.

To make sure the filter is consistent during operation, the landing sequence is delayed from the initial detection to allow the filter to converge. The log-linear property of the IEKF contributes to a fast initial convergence for any trajectory, which is important for repeatability and it reduces already limited hover time requirements. The IEKF used for precision landing uses the same IMU as the autopilot, but runs as a standalone system. In theory, one could use the IMU bias estimates of the autopilot's EKF to hot start the invariant filter. However, due to the already fast convergence of the IEKF this was not found to be necessary.

During the landing sequence, different means of sanity checks are applied to ensure consistency. The raw pose measurements are filtered based on the camera intrinsics and extrinsics, discarding any measurements which are infeasible in terms of the current UAV pose, the FOV and distance to the ground. Most importantly, the consistency of the IEKF is used to assess whether the current position estimate is usable or not and the covariance in position and yaw are be compared to the acceptable tolerances of the landing platform.

If the filter becomes inconsistent, means are taken to improve the estimate or abort the landing procedure. The metrics used to assess filter consistency are further discussed in Section 3.6.

The behavior shown in Figure 3.13 is repeated until the UAV has landed. If the estimator becomes inconsistent or fails a sanity check, the UAV will move back to the initial predefined location. If the filter reacquires track on the way, the precision landing module continues from the current location. In the event that track is lost close to the target, the UAV will land at the current location. This is expected if the camera is mounted in such a way that the marker becomes unobservable as the vehicle is about to land or because the rotational blur when in proximity to the landing target becomes too excessive for reliable detection. The specific altitude at which this happens serves as a tuning parameter and depends on the specific vehicle type and the tolerances of the landing platform. If the pose estimate remains inconsistent or the marker is not detected over multiple precision landing attempts, the UAV will land at the current location or at a nearby safe landing site of choice, depending on the autopilot settings. The failsafe can also trigger from battery depletion.

3.6 Filter Tuning

In state estimation, it is important to assess the consistency of the estimate to ensure optimal performance. Before using the estimator in a production setting, one should make sure the estimate is as close to the true state as possible with an uncertainty reflecting the quality of the estimate. The notion of filter consistency is based on the criteria in Section 5.4.2 of [32], which are recited for convenience:

1. The estimation errors have mean zero (i.e., the estimates are unbiased).
2. The estimation errors have covariance matrix as calculated by the filter.
3. The state errors should be acceptable as zero mean and have magnitude commensurate with the state covariance as yielded by the filter.
4. The innovations should also have the same property.
5. The innovations should be acceptable as white.

To assess the quality of the estimate, several metrics exist. In this thesis, the Normalized Estimation Error Squared (NEES) and Normalized Innovation Squared (NIS) metrics are used to tune the IEKF and determine consistency.

For an estimation error $\xi_k = \mathcal{X}_k \ominus \hat{\mathcal{X}}_k$ at a time step k with estimator covariance $\hat{\mathbf{P}}_k$, the NEES, $\epsilon_{\mathcal{X},k}$, is defined as

$$\epsilon_{\mathcal{X},k} = \xi_k^\top \hat{\mathbf{P}}_k^{-1} \xi_k \tag{3.35}$$

To fit into the Lie group framework, the error ξ_k must be defined according to the composition in Equation 2.4 for a right invariant error and analogously for the left invariant case. Specifically, the capitalized logarithmic mapping of the invariant errors in Equation 2.43 is applied.

The sum of squared normally distributed random variables follows a chi-square distribution. As a result, $\epsilon_{\mathcal{X},k}$ is chi-square distributed if the consistency conditions are met [32]. One can then use the property $\mathbb{E}[\epsilon_{\mathcal{X},k}] = \dim(\xi_k)$ which corresponds to the DOF of the state vector. This can be used to formulate a hypothesis test,

- H_0: $\hat{\mathcal{X}}_k$ with covariance $\hat{\mathbf{P}}_k$ is consistent.

Given a desired Type I error probability α, the filter is consistent with probability $1 - \alpha$ if $\epsilon_{\mathcal{X},k} \in [l_{\mathcal{X}}(\alpha), u_{\mathcal{X}}(\alpha)]$ for a given time k. The bounds for the Confidence Interval (CI) are calculated from Percent Point Function (PPF)[3] for the chi-square distribution, F^{-1}. The CI is then given by

$$l_{\mathcal{X}}(\alpha) = F^{-1}(\frac{\alpha}{2}, \dim(\xi_k)) \tag{3.36a}$$

$$u_{\mathcal{X}}(\alpha) = F^{-1}(1 - \frac{\alpha}{2}, \dim(\xi_k)) \tag{3.36b}$$

The innovation is defined as the measurement residual given in Equation 2.51b and 2.50c for linear right and left invariant updates, respectively. The generalized case for a nonlinear update of the full six DOF pose update is shown in equation 3.30. The innovation covariance matrix $\mathbf{S}_k$ is defined according to Equation 2.50b. Similarly to the NEES defined in Equation 3.35, the NIS is defined as

$$\epsilon_{\mathcal{Y},k} = \nu_k^\top \mathbf{S}_k^{-1} \nu_k \tag{3.37}$$

Using the same approach by formulating a hypothesis test, the bounds for the corresponding NIS CI are then given by

$$l_{\mathcal{Y}}(\alpha) = F^{-1}(\frac{\alpha}{2}, \dim(\mathcal{Y})) \tag{3.38a}$$

$$u_{\mathcal{Y}}(\alpha) = F^{-1}(1 - \frac{\alpha}{2}, \dim(\mathcal{Y})) \tag{3.38b}$$

If the NIS or NEES lies outside the CI, the filter is believed to be inconsistent. There are multiple approaches to assessing filter consistency based on NIS and NEES given values for a set of $k = 1...K$. The most straightforward approach is to count the percentage of values within their respective CI. For a well-tuned filter, $100(1 - \alpha)\%$ of the values should be between the lower and upper bounds depending on the desired α. By adding together all $\epsilon_{\mathcal{X},k}$ or $\epsilon_{\mathcal{Y},k}$ for the K time steps, the resulting random variable is chi-squared distributed with $\dim(\xi_k)K$ or $\dim(\mathcal{Y})K$ DOF [32]. If scaled by the number of samples K, the Average Normalized Innovation Squared (ANIS) and Average Normalized Estimation

[3] The PPF is the inverse of the Cumulative Distribution Function (CDF)

Error Squared (ANEES) can be determined for a higher statistical significance,

$$\bar{\epsilon}_{\mathcal{X},k} = \frac{1}{K}\sum_{i=1}^{K}\epsilon_{\mathcal{X},k} \tag{3.39a}$$

$$\bar{\epsilon}_{\mathcal{Y},k} = \frac{1}{K}\sum_{i=1}^{K}\epsilon_{\mathcal{Y},k} \tag{3.39b}$$

The normalized CIs are

$$\textbf{ANEES}\begin{cases} \bar{l}_{\mathcal{X}}(\alpha) = \frac{1}{K}F^{-1}(\frac{\alpha}{2}, \dim(\xi_k)K) \\ \bar{u}_{\mathcal{X}}(\alpha) = \frac{1}{K}F^{-1}(1-\frac{\alpha}{2}, \dim(\xi_k)K) \end{cases} \tag{3.40a}$$

$$\textbf{ANIS}\begin{cases} \bar{l}_{\mathcal{Y}}(\alpha) = \frac{1}{K}F^{-1}(\frac{\alpha}{2}, \dim(\mathcal{Y})K) \\ \bar{u}_{\mathcal{Y}}(\alpha) = \frac{1}{K}F^{-1}(1-\frac{\alpha}{2}, \dim(\mathcal{Y})K) \end{cases} \tag{3.40b}$$

In the case of inconsistency, the filter is said to be underconfident in terms of ANIS or ANEES if their respective values are below the CI bounds. If the values are above the CI bounds, the filter is deemed overconfident. For example, if the ANEES is too high, the elements of $\mathbf{Q}$ are likely too small given that $\mathbf{R}$ is set correctly. The covariance matrix $\mathbf{R}$ is in general easier identify, because the measurement noise is often given by the datasheet of the sensor used or by experimental results. If the filter is not tuned to consistency it will provide suboptimal estimates. However, overconfidence is considered more dangerous than underconfidence because too small elements in $\hat{\mathbf{P}}_k$ are more likely to cause divergence for most nonlinear filters.

Both the NIS and NEES metrics are useful for tuning, and are the standard tools for checking the third and fourth criteria of consistency. The NEES necessitates knowledge of the true state, which can only be accurately obtained through simulations. For this reason, it is not possible to fully guarantee filter consistency on real data. In real-life experiments, motion capture systems or RTK GNSS can be used to obtain high accuracy measurements serving as ground truth approximations. In this case, the ANEES can be used to check the two first consistency criteria. However, estimates such as the IMU biases are in practice not measurable. This thesis is most concerned with consistency of the position estimate, followed by orientation. The yaw part of the orientation estimate is particularly important for alignment upon touchdown.

The NIS limits the consistency check to the measurement space of the system, but does not require any ground truth data unlike the NEES. Tuning by NIS alone can is the only option in some cases, but it can be very effective due to the innovation covariance directly contributing to the Kalman gain and thus filter efficiency. To check the last consistency condition a sample autocorrelation metric can be applied, but this was not further pursued in this thesis. To assess the filter consistency in terms of NEES, a Qualisys motion capture system was used to serve as ground truth for the pose estimation. With submillimeter

accuracy, the motion capture is deemed sufficiently accurate to for this project. The QTM Qualisys Track Manager was used to record the six DOF pose of the UAV during the experiments. Due to height limitations of the motion capture environment, experiments of the full $10\,\mathrm{m}$ landing sequence could not be conducted. The absolute positioning error is most important in the final approach of the landing sequence, because of the limited landing platform tolerances for horizontal position and yaw angle displacements.

Chapter 4

Implementation

4.1 Companion Computer Software

The vision-based precision landing system proposed in this thesis is intended to be run on the UAV's onboard companion computer. Flying robots usually have a strict weight and power budget and therefore use smaller, lighter companion computers with limited computational power. The proposed solution is intended to be computationally feasible for most Single-Board Computers (SBCs), enhancing its accessibility for lighter and more resource-constrained platforms. The proposed system utilizes several free and open-source software libraries and middleware components. Their intended function and necessity are discussed in this section.

In this project, the Raspberry Pi 4B with 8 GB Random Access Memory (RAM) was used with Ubuntu 20.04 Server as operating system. The Raspberry Pi was used due to its flexibility in terms of General-Purpose Input/Output (GPIO) pins, video encoding, hardware accessories such as modems, low cost and computational power compared to its small size. In addition to the functionality discussed in this chapter, the companion computer is responsible for other tasks such as video streaming for BVLOS flight, telemetry and control links, winching and payload delivery, system monitoring and more. Ubuntu 20.04 Server was chosen for its direct compatibility with the Robot Operating System (ROS) Noetic [33] middleware and because it is less resource demanding than the full Desktop variant.

4.1.1 ROS

ROS is a middleware framework designed to facilitate communication among a distributed network of nodes. Contrary to its name, ROS is not an independent operating system but rather a collection of software frameworks widely employed in the field of robotics. Within ROS, the nodes have the capability to communicate in an asynchronous manner through message passing, utilizing a publish-subscribe model across various topics. Com-

munication can also occur through service calls. The standardized communication allows nodes to be written in different programming languages such as C++ or Python, using Application Programming Interface (API) calls to the underlying framework. The composition of nodes with well-defined inputs and outputs facilitate modular designs of more complex systems and scalability. Topics may have multiple subscribers and publishers, which makes data sharing easier. When published, each message asynchronously triggers a callback function in the subscriber node. The topics have standardized message definitions, with the possibility of creating custom messages. Examples of message definitions include image and pointcloud data, frame transforms and pose with corresponding covariance. The message flow between publishers and subscribers is controller by the `roscore`, which also handles parameter sharing and logging.

The utilization of ROS entails a certain computational overhead, along with dependencies on the underlying operating system. Consequently, the proposed system is not strictly reliant on ROS for its operation. The main reason for the inclusion of ROS was the developer tools during prototyping and tuning. In particular, the objective was to leverage an existing framework capable of visualizing, recording, and replaying data streams. `RViz` was utilized for this purpose, which enables subscription to topics and visualization of data to the user. Spatial information such as poses and trajectories can be displayed in 3D in addition to the support for camera feeds. The second and most important reason was the data collection, commonly referred to as bagging. Bagging allows data from all topics to be recorded and saved to a `rosbag`, which can be replayed later. This is useful when tuning or modifying the system, because it allows for rapid testing on data collected from previous real-life or simulated experiments, ensuring data consistency between trials.

4.1.2 MAVLink

MAVLink is a messaging protocol designed for communication between onboard components of a Micro Air Vehicle (MAV) and as well as the off-board communication between the MAV and a GCS. The MAVLink library is header-only and designed to be lightweight and highly efficient in terms overhead per sent package, making it ideal for platforms with limited computational budget and communication bandwidth. MAVLink supports a wide variety of software and hardware solutions and is widely adopted in research and industry. Similar to ROS, MAVLink works by sending data as predefined messages with standardized definitions [2]. For the isolated purpose of autopilot communication, the advantage of MAVLink compared to larger frameworks like ROS is the reduced computational overhead, simplicity and wider support in terms of hardware platforms and programming languages. MAVLink has official support for bindings in 13 programming languages, with numerous other independent contributions. It can be used directly on ARM7, ATMega and STM32 based microcontrollers or full-fledged operating systems such as Linux, Windows, MacOS, iOS or Android.

In this project, the MAVLink router [34] project was used to read data from the autopilot's IMU and publish the position of the landing target as a setpoint. It is mainly written in C++ and is based on the MAVLink C library. MAVLink router is designed to transmit MAVLink messages to multiple connections, including the GCS. Similar functionality can be performed using `mavros`, a ROS wrapper for MAVLink messaging, but using

MAVLink directly imposes less constraints on the hardware and software platforms required. The MAVLink router enables multiple interfaces, including Transmission Control Protocol (TCP), UDP, Universal Serial Bus (USB) and Universal Asynchronous Receiver-Transmitter (UART). In this project, the latter option was used to communicate with the autopilot.

The most relevant message definitions are listed in Appendix B. The message definitions exhibit non-minimal characteristics, implying that certain fields may be disregarded based on the compatibility and requirements of the receiving system. The heartbeat message definition shown in Appendix B.1 is required by the standard for all components, indicating that it is present and responding. Recall that there are multiple IMUs present on the autopilot. Consequently, in addition to the acceleration and angular speed fields depicted in Appendix B.2, the identification field was utilized to account for the presence of multiple IMUs. In principle, data from all IMUs could be used to reduce noise in the measured angular velocity and accelerometer data in addition to improve the respective bias estimates. To account for autopilots constrained to a single IMU, the optional functionality discussed here was left as a proposal for future work. Within this thesis, the relative pose was employed for the purpose of aiding corrections, thereby utilizing solely the MAVLink 2 extension fields denoted by blue text in Appendix B.3.

4.1.3 AprilTag

The official AprilTag detector based on [4] was used in this project. AprilTag is a C library with minimal dependencies designed for fast detections and is used in many applications including calibration and real-time localization. In this thesis, the AprilTag library was used to detect markers in the raw image stream. The third generation of AprilTag is supported in the official library, meaning flexible tags such as the custom48h12 family is supported with a detector more than two times faster than the previous generation. For the user's convenience, a Python wrapper for running the C code is generated upon compilation. The core functionality of the AprilTag library is accessed through the `apriltag` object, on which the `detect()` function can be invoked:

- `apriltag(family, Nthreads, max_hamming, decimate, blur, refine_edges, debug)`: Constructor for the AprilTag detector object. The `family` argument is the only required argument and set to "tagCustom48h12", corresponding to the marker shown in Figure 2.6a. `Nthreads` is the number of threads used by the detector. The Raspberry Pi 4B used in this project has no simultaneous multithreading hardware, but four Central Processing Unit (CPU) cores capable of running one thread at a time. This value was accordingly set to 4. The value of the `max_hamming` argument corresponds to the number of erroneous bits the detector can correct for. Since larger values require exponentially more RAM and increased false positive rates, this value was set to 1, but other configurations were also tested as discussed in Section 5.2. The `decimation` argument can be used to downsample the image for faster detections at the cost of pose accuracy. Since pixel binning was already applied, this effect was disabled. The other arguments were kept at the recommended defaults.

- `detect(image)`: This function takes a grayscale image as the only argument and returns the detections within the image as a tuple. The tuple consists of individual detections represented as a keyed dictionary. The `id` is an integer codeword used to identify tag instance within a family. This value is used to query the tag size from the design specifications in order to determine scale. The `center` represents pixel coordinates of the tag center. The `lb-rb-rt-lt` value is the four corners of the tag in pixel coordinates. The pixel coordinates of the marker corners combined with the prior information of marker scale and squareness are used to determine the relative pose by solving the PnP problem with the IPPE method. The `hamming` value represents the number of corrected bits. In this case, it would be at most one. The `margin` measures difference in decision threshold and data bit intensity. A higher margin indicates a higher quality decoding process. In practice, this could be used for pre-filtering purposes, but the metric is only valid for very small tags.

4.1.4 OpenCV

OpenCV [3] is an open source library for computer vision, primarily written in C++. The library is optimized and designed for real-time applications and widely adopted in research, industry and among hobbyists. OpenCV is actively maintained and contributed to, aiming to deliver a comprehensive set of efficient implementations for state-of-the-art algorithms related to computer vision. Numerous methods for applications such as photogrammetry, pose estimation, object detection, classification and tracking are readily available from OpenCV. In total, over 2500 optimized algorithms are provided cross-platform for C++, Python, Java and MATLAB interfaces. Instead of downloading prebuilt packages, OpenCV was built from source with support for GStreamer [35]. This is due to the onboard camera using an embedded Image Signal Processor (ISP) to compress and transmit the video. By using GStreamer, OpenCV can utilize hardware video decoding to reduce latency of the video feed and offload the companion computer's CPU to reduce the overall computational load. The OpenCV functions relevant to this thesis are:

- `VideoCapture(index, api_reference)`: Constructor for a `VideoCapture` object using for capturing image sequences. The `index` specifies which video device to open. The `api_reference` is used to specify the specific implementation used in the backend. In this case, it was set to `CAP_GSTREAMER`.
- `set(property, value)`: Used to interface the capture device in order to specify capture settings such as resolution, FPS and exposure.
- `cvtColor(image, conversion_code)`: Returns a representation of the input image in an different color space. This function was used to convert the frames of the RGB video stream to grayscale, which is expected by the AprilTag detector.
- `solvePnP(object_points, image_points, camera_matrix, distortion_coefficients, method)`: Returns the translation and rotation vector representing the object pose based on 3D-2D point correspondences. The `object_points` are the world coordinates of the four corners of the square AprilTag relative to the tag center. The scale of this configuration is determined based on tag codeword returned by the AprilTag detector. The `image_points` are

the four corners' pixel coordinates returned by the detector. Their sequence matters, as it defines the yaw component of the marker orientation. The `camera_matrix` and `distortion_coefficients` are obtained from the camera calibration discussed in Section 3.3.2. Finally, the `method` argument is set to `SOLVEPNP_IPPE_SQUARE` which is suitable for fiducial markers in a square, planar configuration as discussed in Section 3.3.3.

The recursive marker configuration contributes to easier implementation compared to multiple adjacent markers, because the need for a rigid transformation from the detected marker to the platform center was removed. The `CvBridge` ROS package was used to convert between ROS and OpenCV images. This functionality was only implemented for bagging purposes, as it allowed image sequences from experiments to be saved an replayed using the rosbag workflow mentioned in Section 4.1.1. By using the square IPPE method, the object-to-image homography and relative pose can be solved completely analytically. This makes the pose estimation 50 to 80 times faster than the default PnP solver in OpenCV [23].

4.1.5 InEKF

The InEKF C++ project [36] is a modular and flexible library for Lie group operations and invariant filter design. Using static types from Eigen [37] at its core, the library is designed to be efficient and offer good performance. The library features Lie groups such as $SO(2)$, $SO(3)$, $SE(2)$ and $SE(3)$, in addition to relevant group methods like logarithmic and exponential mappings, inversion, adjoint and multiplication. InEKF also allows additional columns to be added to the special Euclidean group and accessory Euclidean states. This means that the double direct spatial isometries of $SE_2(3)$ can be created with the addition of a velocity state to $SE(3)$. The full state can be created by adding augmented states representing IMU biases. The library can also be used with dynamic Eigen types in order to add or remove columns of a special Euclidean group on the fly, which can be useful in applications such as Simultaneous Localization And Mapping (SLAM) where the size of the state may vary. The InEKF library provides Python bindings to the core C++ implementation in terms of prebuilt packages. However, these are only available for the x86-64 architecture. To use the library on the ARM based architecture of the Raspberry Pi 4B companion computer, it was built from source using pybind11 [38] to create the Python bindings.

In this thesis, the classes defined by the constructors mentioned below, along with their member functions from the InEKF library, are utilized:

- `InertialProcess()`: The constructor for a `InertialProcess` class uses a `ProcessModel` base class with the augmented $SE_2(3)$ Lie group. The public function `f(u,dt,state)` representing the nonlinear dynamics used to propagate the state one time step is overridden with the IMU-driven dynamics of Equation 3.16. The constructor overrides the `makePhi(u,dt,state)` function for creating the discrete time transition matrix used to propagate the covariance. The `u` argument is the angular velocity and linear acceleration measurements from the IMU and `dt` is the duration of the time step. The `InertialProcess` has setter functions

to specify the accelerometer and gyroscope noise and the noise of their respective bias dynamics which builds the $\mathbf{Q}$ matrix as discussed in Section 3.4.2. The standard deviations are obtained by multiplying the noise density from Table 4.1 with the square root of the sampling rate.

- `MeasureModel(d, R, error_type)`: The constructor for a `MeasureModel` class. The `d` argument represents the linearized measurement model vector relating the measurement to the state matrix as discussed in Section 2.2.3. For a full six DOF pose update, the measurement matrix `H` is set manually as discussed in Section 3.3.3. The `R` argument represents the measurement covariance matrix and the `error_type` denotes whether it is a right or left invariant measurement. The innovation and its inverted covariance matrix can be calculated by the `calcSInverse(state)` and `calcV(y, state)`.

- `InEKF(process_model, x0, error_type)`: The constructor for a IEKF class. The `process_model` argument is the `InertialProcess` type above. The `x0` argument is the initial state of the filter, a `LieGroup` type which also holds uncertainty. The initial state is obtained from the first valid landing target pose measurement and the initial covariance is set conservatively high relative to the covariance matrices for process and measurement noise. The `error_type` can be either left or right invariant. The right invariant formulation was used in this thesis. The `addMeasureModel(name, measurement_model)` public function can be used to add the measurement models introduced above. The `name` is a string used to identify the correct measurement model object when updating. The public interface of its realization is used to perform the `predict(u, dt)` and `update(name, measurement)`. The current estimated state and covariance can be retrieved by the `state` attribute.

For efficiency, the measurement and process model Jacobians are both calculated analytically for the `InertialProcess` with `MeasureModel` corresponding to pose updates. The functions provided for accessing the current estimate and computing innovations with their corresponding inverse covariances can be used to assess filter consistency during operation or compute NIS and NEES for the purpose of filter tuning as discussed in Section 3.6. Naturally, only NIS can be determined during production flights because no ground truth is available.

In the case of inconsistency in-flight, filter outputs are not published and the UAV behavior is determined by the flow diagram shown in Figure 3.13. The filter inconsistency threshold depends on the nature of the operation and acceptable risk levels. The Type I probability α can in this context serve as configurable parameter in addition to the time duration of which the filter must remain inconsistent before contingency measures are taken.

4.2 Auxiliary Components

4.2.1 Camera Calibration

Besides the software and middleware components employed by the companion computer during its operation, the integration of supplementary hardware and software components is necessary either prior to or during the flight. The calibration procedure is included in this category, as the resulting intrinsics are required when computing the pose, but the routine itself only needs to be performed once and can be conducted on a more powerful desktop computer. The calibration procedure was conducted with a 7×10 grid of AprilTags of size $800\,\mathrm{mm} \times 600\,\mathrm{mm} \times 6\,\mathrm{mm}$ obtained from Calib.io [39]. The targets are made from aluminum and low density polyethylene in order to obtain a rigid structure with a flat surface and low thermal expansion. The individual markers are printed using ultra violet light, which gives a very precise end result and a matte finish which limits calibration noise through reflection. A white border is present around calibration target to provide a contrast to the grid elements. This increases the probability of detection and makes it easier for the user to hold the target during calibration without occluding the elements. The manufacturing tolerances aims to be as low as possible, because they in general need to be an order of magnitude lower than the desired calibration accuracy as discussed in Section 2.3.1. The targets from Calib.io have feature point tolerances of $0.1\,\mathrm{mm}$. The pattern of the calibration target can be seen in Figure 4.1.

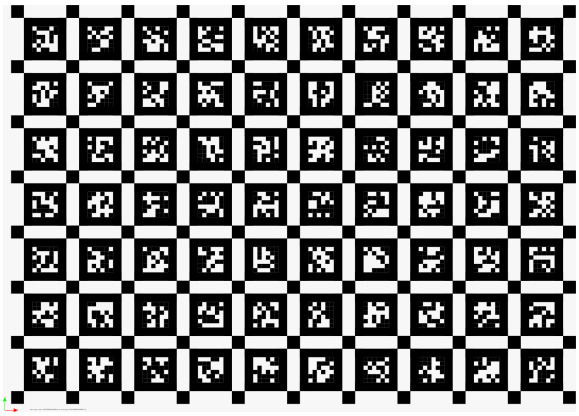


Figure 4.1: Target used for camera calibration.

The benefit of using AprilTags instead of the conventional checkerboard target is to detect partially visible components of the calibration target. To get the best calibration results possible, it is important to ensure that the calibration target fills the entire camera FOV over the union of image frames used for calibration. When using a checkerboard target, the individual squares cannot be accurately detected when partially visible. In practice, this means that there is some padding around the image border in which the calibration routine has little to no information. This impairs the accuracy of the lens distortion parameter estimation. The use of AprilTags during calibration removes ambiguity related to

calibration target orientation. When using AprilTag feature points, the data collection is simplified and the expected result improves.

The free and open-source calibration toolbox Kalibr [40] was used to determine the intrinsic matrix in addition to the IMU-to-camera transform and time shift. Kalibr supports calibration with the pinhole camera model, using radial and tangential distortion as discussed in Section 3.3.2. The second order models were used for both the radial and tangential distortion models.

4.2.2 Autopilot

In this thesis, the PX4 flight control software was used on a CubePilot Cube Orange autopilot. PX4 is a free and open source flight control software, which is extensively used in research, consumer drones and industry. The use cases are numerous, ranging from UAVs to submersibles and rovers. PX4 is designed to be modular and flexible, allowing user to customize it to their needs. Although PX4 has been used for countless experimental platforms, the most common airframes include multicopters, fixed wings and VTOLs. In addition to the flight control software, PX4 provides drivers and middleware for an extensive set of sensors and hardware. PX4 is actively maintained and supported by contributors from industry, manufacturers, researchers and hobbyists. The communication between the autopilot and the precision landing system is limited to reading IMU data and publishing precision landing setpoints. The precision landing module in this thesis is written according to the Landing Target MAVLink Protocol, which in principle is agnostic of autopilot choice. The functionality is tested and confirmed to be working as expected with PX4. In principle, it should be compatible with other MAVLink compliant systems such as ArduPilot, but this is not tested. The integrated IMU of the Cube Orange was used as input to the invariant filter, although an external IMU could also be used. The CubePilot Cube Orange features three IMUs from TDK InvenSense. Their key properties are listed in Table 4.1 and the full accelerometer and gyroscope characteristics of the datasheets are shown in Appendix C.

IMU	Gyroscope noise	Accelerometer noise	Temperature controlled	Mounting
ICM20649[1]	$17.5\frac{mdps}{\sqrt{Hz}}$	$285\frac{\mu g}{\sqrt{Hz}}$	Yes	Fixed
ICM20602[2]	$4\frac{mdps}{\sqrt{Hz}}$	$100\frac{\mu g}{\sqrt{Hz}}$	Yes	Isolated
ICM20948[3]	$5.1\frac{mdps}{\sqrt{Hz}}$	$230\frac{\mu g}{\sqrt{Hz}}$	Yes	Isolated

Table 4.1: Overview of IMU models, their noise spectral density and mounting for the CubePilot Cube Orange.

The isolated mounting means the respective IMU is mechanically isolated from the Cube Orange through a material which dampens high frequency vibrations. In practice this

[1]Full ICM20649 datasheet: `https://invensense.tdk.com/wp-content/uploads/2021/07/DS-000192-ICM-20649-v1.1.pdf`

[2]Full ICM20602 datasheet: `https://invensense.tdk.com/wp-content/uploads/2016/10/DS-000176-ICM-20602-v1.0.pdf`

[3]Full ICM20948 datasheet: `https://invensense.tdk.com/wp-content/uploads/2021/10/DS-000189-ICM-20948-v1.5.pdf`

works as a low-pass filter, allowing the slower dynamics of the UAV to pass and high frequency vibrations from the motors to be dampened. The ICM20602 was chosen for this thesis due to it having the lowest noise spectral density for accelerometer and gyroscope. The spectral density of the noise is large compared to high-end IMUs, but acceptable for this application given a sufficiently high update rate of the filter to limit dead reckoning time. In general, the bias of a Microelectromechanical Systems (MEMS) gyroscope is influenced by fluctuations in temperature. By actively controlling the temperature, it is reasonable to assume that the IMU is operating within its calibrated and nominal temperature range, thereby mitigating the bias drift to some degree. As a consequence of bias instability, the bias gradually deviates to a notable extent during operation, even in the presence of a constant temperature. For this reason, it is necessary to estimate the bias in order to compensate for its drift. In the intended use case where the UAV is set to land after a round-trip long range mission, the bias may have drifted significantly. This is motivates the inclusion of bias estimation in the proposed system.

The IMU data was streamed from the autopilot to the companion computer using MAVLink messages over UART. The message rate was increased from the standard $50\,\mathrm{Hz}$ to $100\,\mathrm{Hz}$, allowing the IEKF to more accurately track the vehicle dynamics.

4.2.3 Landing Target

The landing target fiducial marker was manufactured similarly to calibration target discussed in Section 4.2.1, but is expected to undergo daily usage on a larger scale. Considering their specific form factor, these targets typically incur high costs and require significant time for international shipping. This serves as a driving force for exploring alternative options that are more affordable and can be obtained with shorter lead times. The landing targets used in this thesis were custom built from CEWE [41], which are produced in Norway. The landing target was manufactured with a matt finish to limit noise in terms of reflections. An aluminum plate of dimensions $900\,\mathrm{mm} \times 900\,\mathrm{mm} \times 3\,\mathrm{mm}$ was used for durability, weather and scratch resistance. Unlike the calibration target, the landing targets are intentionally designed without a white border. This deliberate decision aims to maximize the available space on the landing platform. In the intended usage scenario, the landing target is affixed to a white background surface, ensuring sufficient contrast for reliable detections. Consequently, the inclusion of additional padding is unnecessary. The landing target is manufactured according to Figure 2.6b and is shown in Figure 4.2. The thickness of the plate is $3\,\mathrm{mm}$, only half the thickness of the calibration target. As a result, the rigidity of the landing target is significantly reduced, leading to increased bending. While this may adversely affect accuracy in situations involving handheld use or mounting with a single point of contact, it is not considered problematic in the current application. This is because the landing target rests horizontally on a flat surface, where the reduced rigidity is not expected to compromise its functionality.

The main drawback of the landing target compared to the calibration target is the manufacturing feature point tolerance. Calib.io guarantees $0.1\,\mathrm{mm}$ accuracy, whereas CEWE claims $1\,\mathrm{mm}$ accuracy. These claims have not been investigated in this thesis as the accuracy was deemed sufficiently high for the intended purpose. This is because the same level of accuracy required for calibration often is not required during operation. According to

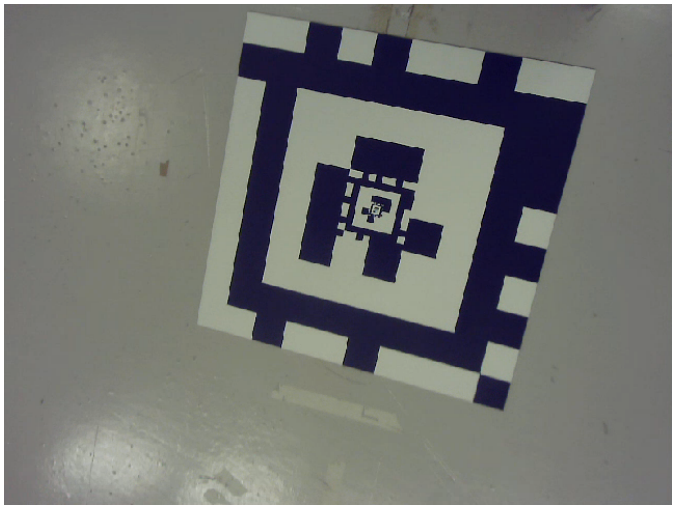


Figure 4.2: Image of the landing target capture from the UAV's onboard camera in-flight as seen from $2\,\mathrm{m}$ AGL.

[42], the pixel Root Mean Square Error (RMSE) of AprilTag localization within the range where the recall was more than 95% remains almost constant as the distance to the tag increases. This result is only indicative for this thesis as the camera resolution used in the original paper was notably higher, but it can be used in a qualitative argument for why the higher tolerances are acceptable. From Equation 2.60, it can be seen that a small displacement in pixel coordinates has a linear relation to the displacement of world coordinates. The approximation is less accurate towards the edges of the image due to distortion, but it is still suggestive for how the estimation error relates to the distance between the camera and the marker. Efforts in terms of experimental data to investigate this relationship were not in scope for this thesis, but the single marker results of [43] coincide with the observation. In the operating range, the absolute position error therefore approximately follows the altitude above the landing target linearly. In practice, one is most concerned with the positioning error close to the landing target, which should be well within the mechanical tolerances of the landing station alignment mechanism in either scenario.

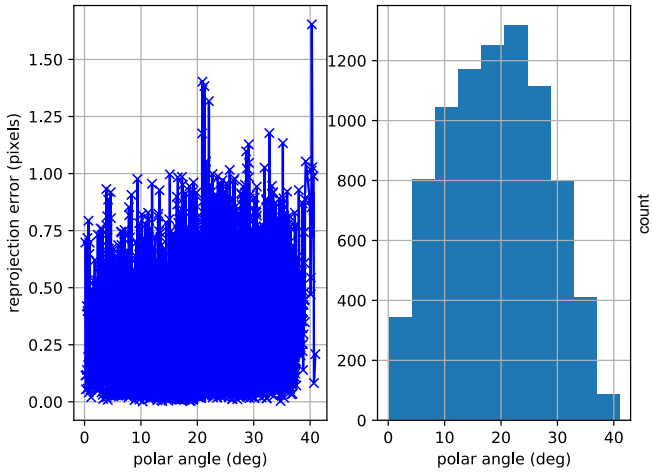
Chapter 5

Results and discussion

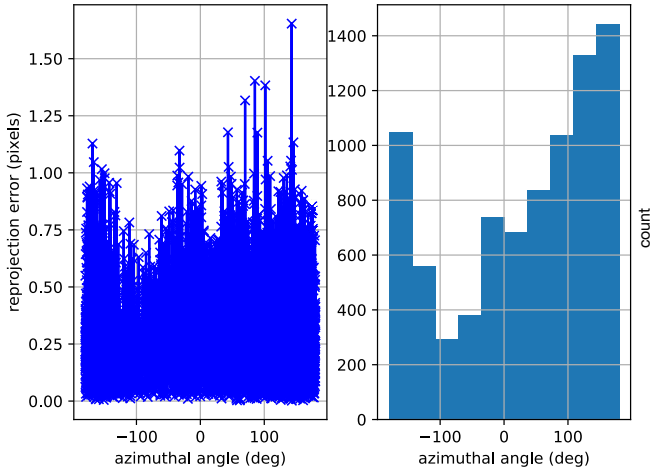
To test the viability of the proposed system, a combination of unit and integration tests were conducted in addition to a final complete system test. First, the quality of the camera calibration was evaluated as it affects all other components related to pose estimation. The reprojection errors and coverage are the most important metrics for this evaluation. Thereafter, the marker detection was tested in terms of recall, detection time and pose calculation time as a function of distance to the marker. Then, the CPU load and RAM of the marker detection and pose calculation was measured to provide insight inon the computational load of the system. Lastly, the filter consistency and full system performance was evaluated. The filter consistency is quantified in terms of NIS and NEES. The full system performance is based on the position and yaw errors over multiple flights. Specifically, the RMSE of the filter estimates and the ground truth mean and standard deviations over all the flights are used to determine the reliability and reproducibility of the system. The same metrics are also discussed in the special case where the UAV has landed, as that is were the accuracy and precision matters most.

5.1 Camera Calibration

The camera calibration was performed using 32 images of the calibration target, applying the method described in Section 3.3.2 and the tools discussed in Section 4.2.1. In each image, the 7×10 AprilTag grid pattern was used with four corners serving as a feature point for each AprilTag. Although every feature point was not visible in every frame, the overall feature count is considered high. After minimizing the functional in Equation 2.73 expanded with the distortion parameters of Equation 3.3 and 3.4, the final reprojection errors for all visible feature points in the images was used as a reference for calibration quality. The reprojection errors are decomposed into their vertical and horizontal component, respectively the polar and azimuthal angle. The distribution of feature points and the corresponding reprojection errors is shown in Figure 5.1.



(a) Reprojection error and feature point distribution for polar angle.

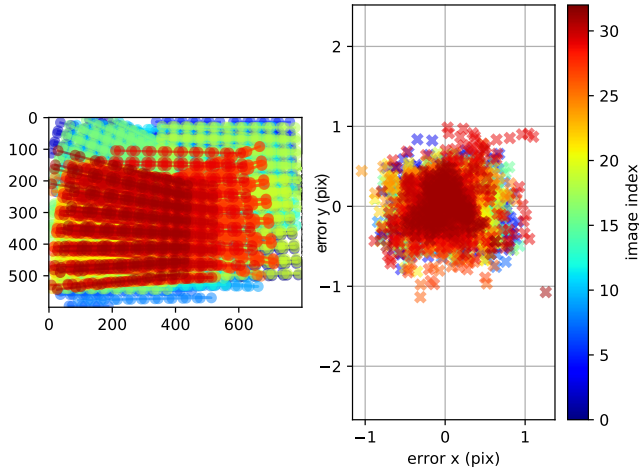


(b) Reprojection error and feature point distribution for azimuthal angle.

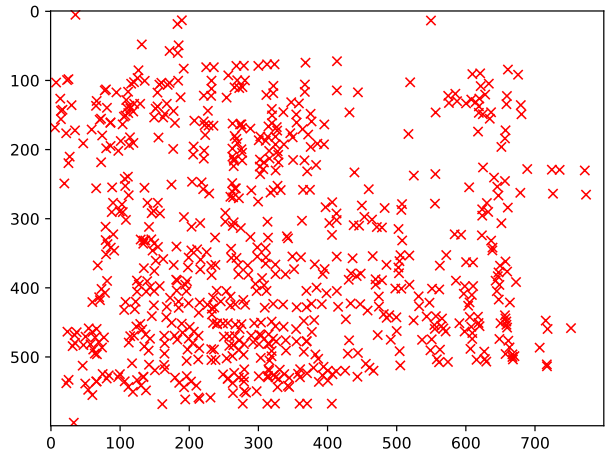
Figure 5.1: Reprojection error for the calibrated camera model and distribution of detected feature points as a function of angular coordinates.

The desired upper bound for reprojection error depends on camera type and resolution. Since the unit of reprojection error is pixels, the reprojection error will in general increase with camera resolution. For a $1920\,\mathrm{px} \times 1080\,\mathrm{px}$ resolution ($2.07\,\mathrm{Mpx}$) camera using pixel binning, a subpixel reprojection error in each dimension is considered good for most computer vision applications. Figure 5.1a and 5.1b show that the reprojection error for the vast majority of feature points are located below this upper bound. Specifically, 99.8% of the decomposed reprojection errors are subpixel in magnitude.

By looking at the union of calibration target feature point locations from all images, the image coverage of the calibration process can be evaluated. If the location of the calibration targets is overrepresented in the image center, the estimated intrinsics do not reflect the true projection of the camera. In particular, the distortion parameters are usually inaccurate due to their effect being most prevalent towards the edges of the image. Consider the case where the calibration is performed exclusively on targets close to the image center. In this case, little to no distortion effects are perceived. The calibration then overfits due to the low amount of noise, which is reflected in a low reprojection error.



(a) Image coverage and reprojection error scatter plot by image index.



(b) Location of removed outliers in the image using pixel coordinates.

Figure 5.2: Visualization of image coverage during the calibration process, the reprojection errors scatter plot for each image frame and the location of outliers in the image.

For this reason, reprojection error alone is not a sufficient metric for evaluating calibration quality. As discussed in Section 2.3.1, parallel target planes might cause degenerate configurations and thus not contribute to the overall intrinsic estimation. The target should therefore be rotated over the calibration sequence to provide more data points. Figure 5.2a shows that the camera's FOV was mostly covered by the 32 images used. Individual images reveal sufficient rotation of the target during the calibration sequence. In the bottom right part of the image, the coverage does not fully extend to the corner. Since the other three corners are covered, it likely does not affect the radial distortion due to its symmetry. However, the gap in coverage might degrade the estimate of tangential distortion. Due to the tangential distortion in general being less prevalent than the radial one and the comparatively small area of the coverage gap, this was not considered to be an issue.

In Figure 5.2b, the distribution of outliers from the calibration images are shown in pixel coordinates. During the calibration, outliers of higher-than-normal reprojection errors are excluded because they may delude the estimated intrinsics. Outliers are expected and may happen for several reasons. During the calibration sequence, the camera was rigidly mounted and the calibration target was moved within the camera's FOV. Since the calibration target is not perfectly rigid and point features are subject to manufacturing imperfections, some noise is expected. Secondly, image noise and errors in the feature detector contribute to noise and increase in reprojection error. If there are clear patterns in the location of outlier, this might indicate an inadequate mathematical model of the camera. For example, if there is a clear radially symmetric bias in scatter towards the edges of the image, this could imply that a radial distortion model of second order is insufficient. In this case, a higher order model or a different model altogether might be more suitable. The distribution shown in Figure 5.2b is mostly uniform over the image, with a slight skew towards the lower left corner. This is likely due bias in the image sequence, in which the lower left corner of the image frame was slightly overrepresented as shown in Figure 5.2a.

5.1.1 Final Intrinsic Parameters and Reprojection Errors

The final intrinsic parameters, reprojection error mean and standard deviations are shown in Table 5.1. Most notably, the reprojection errors are zero mean with an acceptably low standard deviation. The camera model is subject to some distortion, but it is not excessive. Of the two distortion models, the radial one is most prevalent as expected. Overall, the standard deviations for the estimated parameters is relatively low. This is likely due to the large feature count in the calibration images. The standard deviation of the projection parameters is slightly higher than expected, but reflects that the working distance during the calibration was varying, as is the case for the intended application. To reduce the standard deviation in the projection parameters, the calibration target must remain in focus over the entire working distance. To achieve this, it would likely be required to use a camera with an increased depth of field. This would in turn require a smaller aperture, which results in reduced low-light performance. Alternatively, one could use a camera with a longer focal length but that would in turn reduce the FOV. The calibration results are deemed satisfactory for the intended use case in this thesis.

Parameter	Mean value	Standard deviation
f_x	557.71648131	0.91693309
f_y	557.32860671	0.87194409
c_x	409.18438123	0.93325968
c_y	255.37187531	0.74082091
k_1	0.00144607	0.00195112
k_2	0.01087497	0.00222046
p_1	0.00104504	0.00040071
p_2	-0.00287254	0.00046702
Horizontal reprojection error	0.000000	0.237480
Vertical reprojection error	0.000001	0.212465

Table 5.1: Overview of final projection and distortion parameters in addition to reprojection error after camera calibration.

5.2 Marker Detection and Pose Measurement

To further assess the system performance, the marker detection and pose calculations were timed. The recursive landing target shown in Figure 2.6b was used to conduct the tests. During the data collection process, the UAV started in a landed state centered on the landing target, performed a takeoff and climbed to $11\,\mathrm{m}$ AGL. Over the course of the experiments, at least one of the markers was visible in every frame. Despite the possibility for multiple markers being visible, only the marker with the highest confidence determined by the detector was used for pose calculations. The computations are performed on the Raspberry Pi 4B companion computer to better represent the expected computational load during operation. The CPU load factors and RAM usage are given relative to the idle performance of the system.

The computation times are important, because they determine the maximum frequency for IEKF updates. This affects filter drift between updates and therefore consistency. In conjunction with assessing the average system load, it was desirable to analyze how the computation times would vary over the expected trajectory. This is because the AprilTag 3 detector significantly increases detection speed for smaller tags compared to the previous generation. The tag size was expected to vary during the experiment. In particular, the tags are perceived as small during the initial stages of the landing sequence when the UAV is furthest away from the landing target. The worst case behavior was used when determining filter update rates.

5.2.1 Single Error Bit Tolerated

Initially, the AprilTag detection method was performed with the `hamming` argument set to 1, meaning the detector was able to tolerate one erroneous bit in the marker during detection. A vertical velocity of $1\,\mathrm{m\,s^{-1}}$ was used, with a camera FPS of 15. In total, 172 images were used in the first trial. Due to the flight altitude, motion capture could not be used as a ground truth. Therefore, the pose calculations were used to determine the

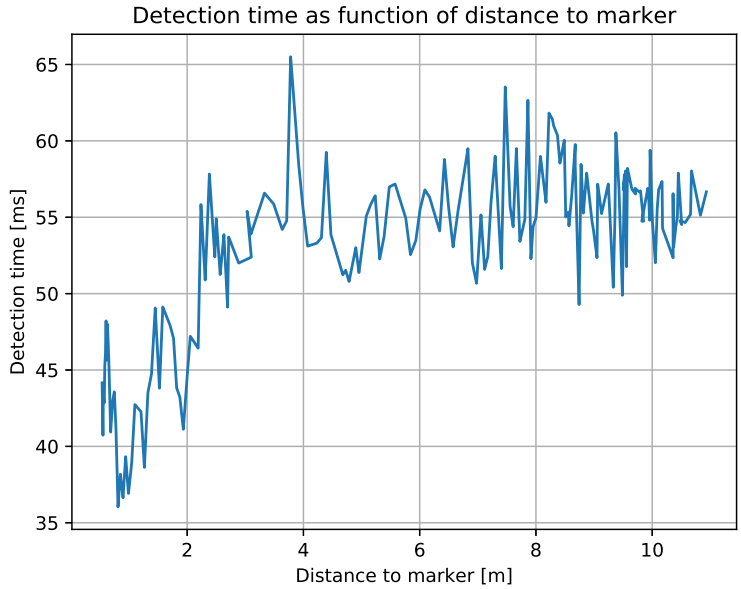


Figure 5.3: Time to detect a landing target where one erroneous bit is tolerated as a function of distance to the marker.

distance between marker and camera. The accuracy of the pose measurements is further discussed in Section 5.3. Nonetheless, they are useful to show trends in the captured data. Detection time and time required to compute relative pose using the IPPE method were determined as a function of distance to the landing target. The results are shown in Figure 5.3 and 5.4.

The most apparent result is how much faster the pose calculation is relative to the marker detection. This is expected, due to the analytical solution of the IPPE algorithm discussed in Section 2.4. The average time for calculating the relative pose was only $0.7\,\mathrm{ms}$ with a standard deviation of $0.9\,\mathrm{ms}$. The expected computation time is higher when the marker is close to the camera, but it is not significant for the overall computational load. Being almost two orders of magnitude slower on average, the tag detection imposes a much larger contribution for the computational load.

In Figure 5.3, the average detection time was $52.8\,\mathrm{ms}$ with a standard deviation of $5.9\,\mathrm{ms}$. It is also clear that the computational time for marker detection increases noticeably as the distance increases. Some fluctuations are seen when the proximity to the target is low, due to the detector switching from the smaller markers to the next one. From $2.15\,\mathrm{m}$ and beyond, only the outermost marker is used for detection. This matches the theoretical results from Table 3.1 for when the second tag can be reliably detected. For the sake of intuition, the landing target image shown in Figure 4.2 is the perceived marker size to the onboard camera at $2\,\mathrm{m}$ AGL. From $10\,\mathrm{m}$ AGL the outermost tag appears similar

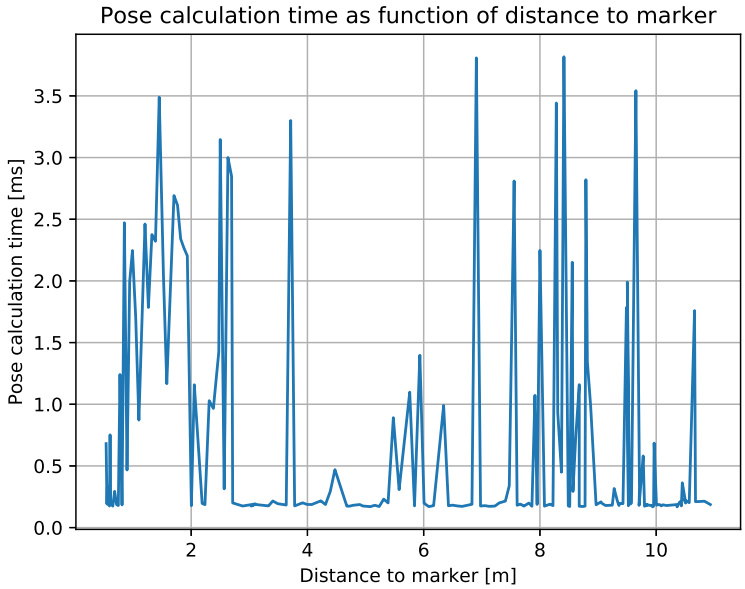


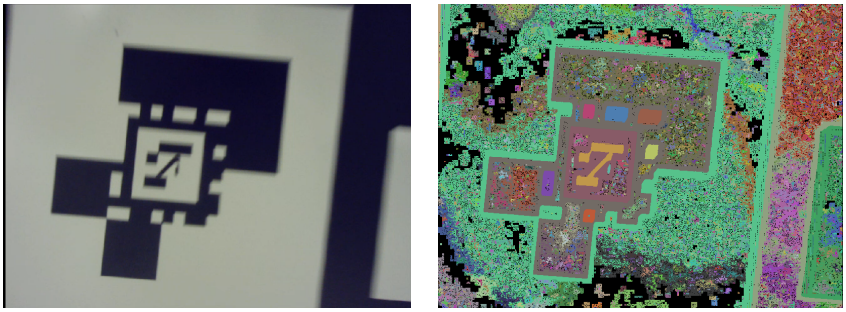
Figure 5.4: Time to compute relative pose as a function of distance to the marker.

in size to the second largest marker at $2\,\mathrm{m}$ AGL. In the range of $3\,\mathrm{m}$ and beyond, the expected computation time stabilized at $59.1\,\mathrm{ms}$ on average. The detection rate matches the the desired filter update frequency, which is set to match the 15 FPS of the camera. Considering the average performance of $52.8\,\mathrm{ms}$ and a worst case behavior of $65.5\,\mathrm{ms}$, this is achievable. For reference, a desktop with a Intel i7-7500U CPU running the same detector at four cores averaged a $35.2\,\mathrm{ms}$ detection time on the same data. Increasing the `decimate` and `blur` arguments of the detector would also speed up the detector, but this results in reduced accuracy of feature point localization and thus a worse pose estimate. In total, the marker detection recall was at 98% during the first experiment.

The marker detection was found to occasionally fail when the camera was located close to the landing target. There are mainly two reasons for this. First, the camera has a fixed focus which is not optimized for close range sharpness. Second, the vibrations and motion of the UAV cause artifacts in terms of motion blur when close to the target. The rotational blur seems to be the main contributor to reduced recall. An example frame is shown in Figure 5.5a. In this case, the lines between the bits have a lower contrast which affects the segmentation step of the detection algorithm. In particular, quadrilaterals are difficult to fit to the tag contours due to the noisy output of the segmentation step as shown in Figure 5.5b.

This effect could likely be reduced by decreasing the shutter time, but that would in turn reduce the low-light performance. The experiments were conducted in a dimly lit environment which reflect the lighting conditions in a production setting. In this case, insufficient

lighting was not considered a problem and it is likely that a decrease of shutter time could be applied to increase the recall. This was not pursued as the current recall was acceptable. One could also adjust the detection confidence threshold for marker candidates, but that would increase the risk of false positives in addition to returning markers with poorly estimated feature point coordinates.



(a) Example of motion blur when the camera is close to the landing target.

(b) Grouping of the connected components in the image.

Figure 5.5: Example of an input image where the AprilTag is not detected. When the camera is subject to a large angular speed during shutter, a the relative marker displacement causes a blurring effect. The rightmost part of the smallest tag is subject to an artifact similar to aliasing. The detector is then unable to determine the location of the bit from the segmented pixel map and fit a quadrilateral shape to the feature points without significant error. In this case, the confidence of the detector was adjusted such that this marker is ignored.

During the experiment, the CPU load factor and system RAM usage was logged. The load factor represents the percentage of one CPU core used on average during the experiment. Since the companion computer has four cores, the load factor can take a maximum value of 400%. During the first experiment, the average load factor was at 314% and the RAM usage was 249 MB over the idle value. These values represent the system load in its entirety, including middleware.

5.2.2 No Error Bits Tolerated

The recall and computation times in the first experiment were overall considered satisfactory, but a second experiment was conducted to assess how the error bit tolerances would affect overall performance using a larger set of pictures. During the second experiment, the `hamming` argument of the detector was set to 0, making it unable to correct for erroneous bits. In total, 687 images were used. The results are shown in Figure 5.6.

In the second experiment, the average detection time was 56.2 ms with a standard deviation of 8.5 ms. The mean and standard deviation do not differ greatly from the first experiment, however the load average and RAM usage is decreased. On average, the load average was reduced from 317% to 274% with the RAM usage decreasing from 249 MB to 152 MB. The lower computational load allows more auxiliary processes to run on the companion computer or more freedom in terms of required processing power. Unfortunately, the worst

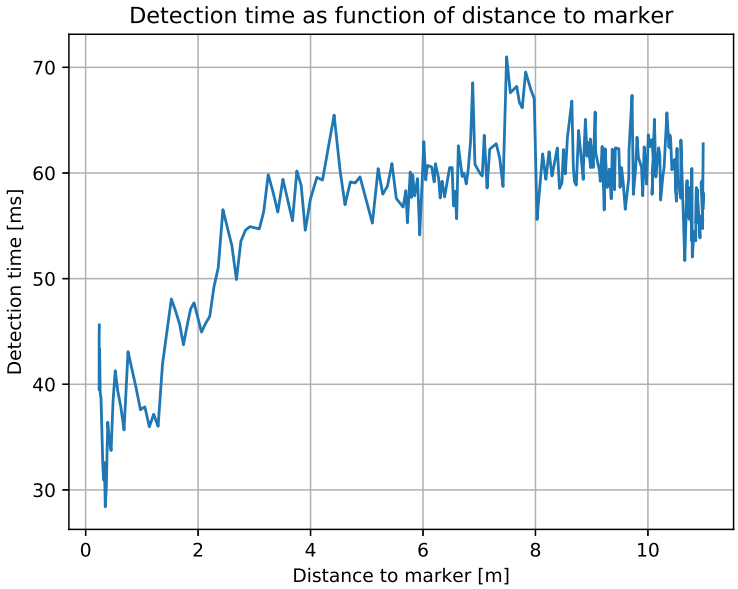


Figure 5.6: Landing target detection time where no erroneous bits are tolerated.

case detection time is increased to $71.0\,\mathrm{ms}$. This means that the filter, temporarily might update at $14\,\mathrm{Hz}$. This is still close to the target update rate, but might affect performance to some degree. The recall was also slightly lower, at 96% over the entire trajectory. The pose calculation time plot is not shown, as it remains consistent between the two experiments. The results are summarized in Table 5.2. Attempts were made to investigate the effect of allowing the detector to accept two erroneous bits, but the companion computer with $8\,\mathrm{GB}$ of RAM suffered from insufficient memory as a result of the exponential memory complexity of the detection algorithm.

Similarly to the majority of detection problems, a fundamental trade-off arises between precision and recall. Increasing the acceptable number of erroneous bits corresponds to an elevation in recall, albeit potentially compromising the detector's precision. In the envisioned utilization scenario, it is improbable for elements within the operational environment to exhibit resemblances to the landing targets, except for the markers on the landing platform. Additionally, experimental results were obtained without any false positives. Consequently, configuring the detector to accommodate a single erroneous bit was deemed admissible. The decision was also motivated by the hamming distance of 12 for the custom48h12 family used in this experiment. This means that all members of the family differ from each other by at least 12 bits. As multiple adjacent landing platforms are envisioned in future applications, the $42,211$ unique tag codewords in the custom48h12 family allow for a scalable solution where irrelevant tags can safely be ignored.

	Metric	Mean	Standard deviation	Maximum
	Pose calculation time [ms]	0.7	0.9	3.8
Single error bit tolerated	Detection time [ms]	52.8	5.9	65.6
	Recall [%]	98		
	Load average [%]	314		
	RAM usage [MB]	249		
No error bits tolerated	Detection time [ms]	56.2	8.5	71.0
	Recall [%]	96		
	Load average [%]	274		
	RAM usage [MB]	152		

Table 5.2: Overview of detection time, recall, load average and RAM usage for both AprilTag detector configurations in addition to the pose calculation time using the IPPE method.

5.3 Filter Consistency

To validate the output of the IEKF in terms of consistency, the NIS and NEES metrics were used as discussed in Section 3.6. The tuning process for NIS was performed on data collected from a piloted decent from $10\,\mathrm{m}$ AGL subject to horizontal movement. The UAV idled for $2\,\mathrm{s}$ to allow the filter to initialize before the movement started. Using the methodology from Section 3.6, the filter was tuned by manually adjusting the $\mathbf{Q}_k$ and $\mathbf{R}_k$ covariance matrices until consistency was achieved. The 3D flight path of the UAV during the experiment is shown in Figure 5.7. The NEES was tuned in a similar manner, but the flight trajectory could not represent the full landing sequence due to height limitations in the motion capture room. For this reason, the trajectory was limited to $4\,\mathrm{m}$ AGL. Still, the results are useful for quantifying the filter performance close to touchdown where the accuracy matters most. When the filter was tuned and the two metrics were considered satisfactory, the performance was validated on a second dataset to prevent overfitting to the data from the first experiment. In all datasets, the touchdown itself was cropped out from the NIS and NEES data because of spikes in the accelerometer data affecting the filter's estimate. In the final decent stage just before touchdown, as described by Figure 3.13, the filter is not used for guidance, meaning this decision does not affect the overall system performance.

5.3.1 NIS

In the inertial process model shown in Equation 3.25, we can see that the main contributors to process noise come from the accelerometer and gyroscope measurements which are propagated into the position and orientation. For this reason, the IMU noise parameters were used as a starting point for the $\mathbf{Q}_k$ matrix. The gyroscope and accelerometer random walk including the bias random walk values were obtained from the datasheet which can be found in Appendix C.2. The elements of the $\mathbf{R}_k$ corresponds to the measurement noise which in this case arises from the pose calculation based on the marker detection using the onboard camera. The measurement noise is in general greater further away from the marker, as discussed in Section 4.2.3. Since the performance close to

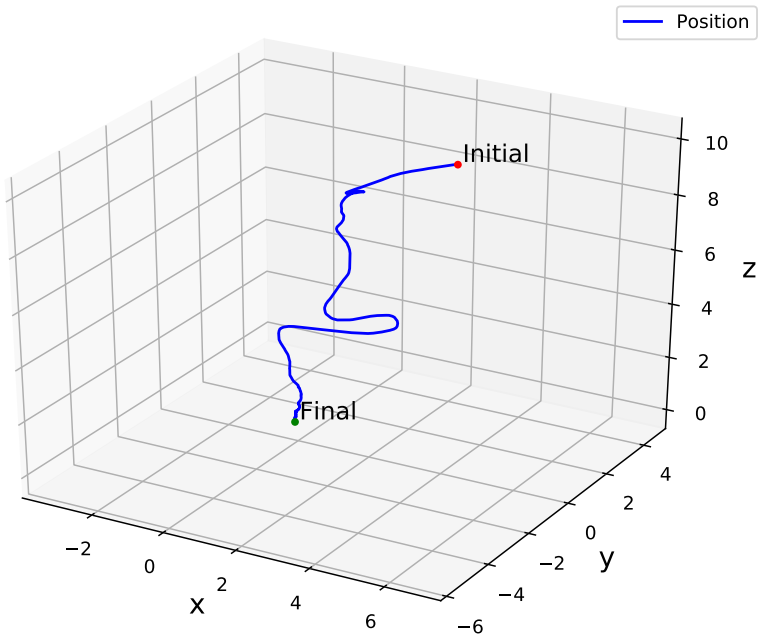


Figure 5.7: 3D visualization of the flight trajectory used for NIS tuning as estimated by the filter after the final tuning was applied. The vehicle position is given in a right-handed coordinate system centered on the landing target such that the z-axis points upwards when the target lies flat on the ground.

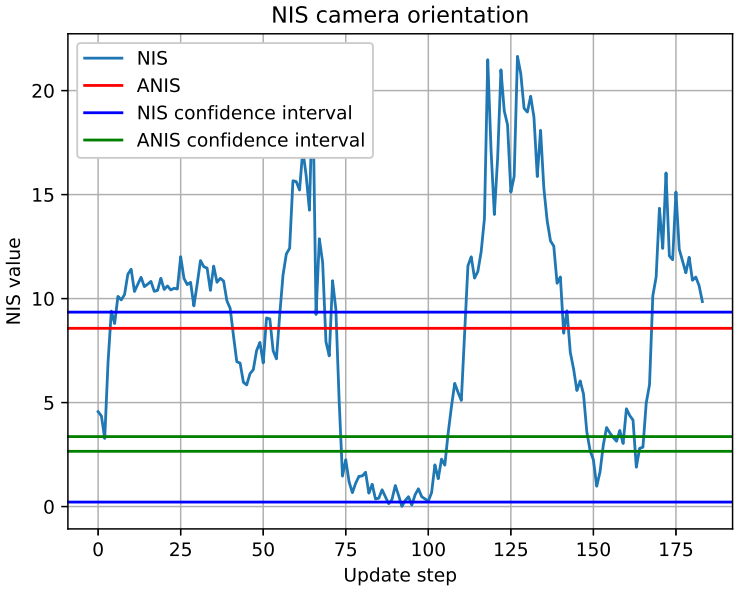
the landing target is of most importance, an initial guess of the measurement noise was obtained by comparing the motion capture data to the raw pose calculation values. The process noise is as shown in Equation 3.24 and the diagonal elements of $\mathbf{Q}_k$ correspond to $\text{diag}(\begin{bmatrix}\boldsymbol{\omega}_{nt}^\top & \mathbf{a}_{nt}^\top & \mathbf{0}^\top & \mathbf{b}_{nt}^{\omega\top} & \mathbf{b}_{nt}^{a\top}\end{bmatrix})$. By assuming constant noise over the respective predict and update intervals h, the initial continuous-time measurement and process noise matrices were obtained as discussed in Section 4.3 of [44]:

$$\mathbf{Q}_k = \mathbf{Q}(t)h_{predict} \tag{5.1a}$$

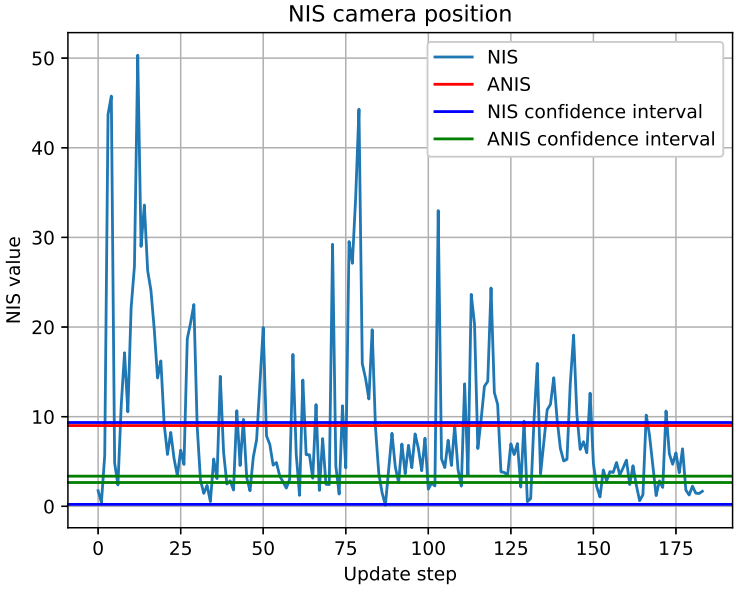
$$\mathbf{R}_k = \frac{\mathbf{R}(t)}{h_{update}} \tag{5.1b}$$

The initial values were adjusted for a $100\,\text{Hz}$ prediction rate, with full pose updates obtained at a $15\,\text{Hz}$ rate. If higher-grade IMUs were used, the update rate could likely be reduced while maintaining consistency because the noise and drift would be less prevalent. To simplify the tuning process, the NIS values were decomposed into a position update and orientation update component. The initial NIS and ANIS values in addition to their respective CI are shown in Figure 5.8.

In Figure 5.8a, only 46.2% of the NIS values fall inside the 95% CI. 54.2% of the values are above the upper bound and the resulting 1.6% are below the lower bound. This



(a) Orientation update component.



(b) Position update component.

Figure 5.8: NIS values with the initial tuning for measurement and process noise covariance matrices.

clearly shows that the filter is overconfident in terms of orientation update NIS, meaning the respective elements of the covariance matrix must be increased. For a higher statistical significance, the ANIS can be compared to a CI scaled by the number of data points. The ANIS of 8.57 is far above the upper bound of 3.36, supporting the hypothesis of inconsistency due to overconfidence.

Similarly to the orientation update, the NIS for position updates shown in Figure 5.8b exhibit symptoms of filter overconfidence. In this case, 67.4% fall inside the 95% CI. 32.1% of the values are above the upper bound while 0.5% are below. This is better than the orientation update NIS, but still not passable as consistent. The ANIS of 9.01 is above the upper bound, but relatively close to the orientation update ANIS. Although a larger percentage fall inside the CI for the position update, the NIS has much higher spikes than in the orientation update. By looking at the ANIS alone, this behavior might go unnoticed which illustrates why both values should be considered when assessing filter consistency.

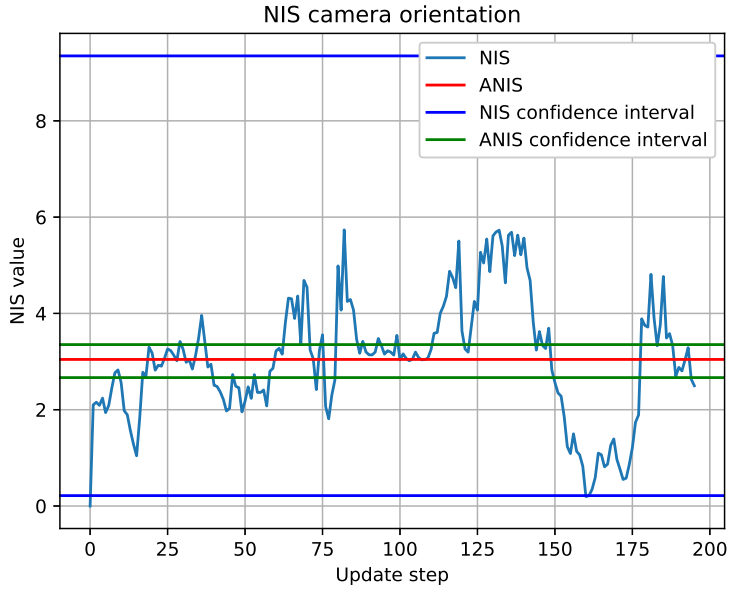
After the tuning process was completed[1], the NIS values were computed for the validation flight. The results are shown in Figure 5.9. In this experiment, 98.5% of the NIS values related to the orientation update fall inside the 95% CI. The resulting 1.5% are below the lower bound. Similarly, the ANIS of 3.04 lies inside the scaled CI of $[\bar{l}_{\mathcal{Y}}(0.05) = 2.67, \bar{u}_{\mathcal{Y}}(0.05) = 3.52]$.

The NIS values related to the position update shown in 5.9b display similar improvements as the orientation update NIS. In this case, 87.2% of the values lie inside the 95% CI. 5.6% of the values are located below the lower bound and 7.1% are located above. Overall, the NIS of the position update is more noisy with larger spikes than the orientation update. This is likely due to the uncertainty in the position part of the relative pose measurement fluctuating more than the orientation part as detector switches from one marker to another or as the perceived size of the marker changes. The ANIS of 3.15 fall inside the scaled CI, albeit somewhat closer to the upper bound than the orientation update NIS. This property is also reflected in the number of samples above the upper NIS CI bound being higher than the samples below the lower bound.

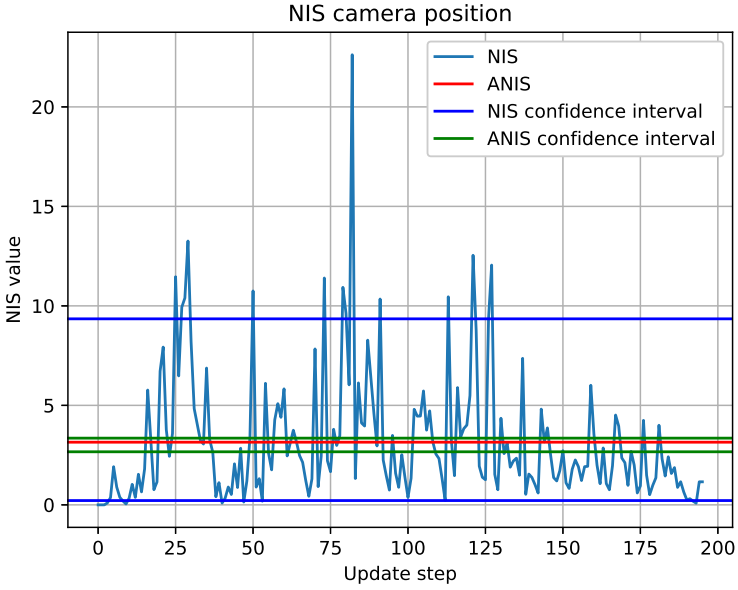
In general, the filter uncertainty tends to grow with increased distance to the landing target. This matches the hypothesis of approximate linear relationship between the pose calculation error and distance to the marker discussed in Section 4.2.3. The same behavior was observed for the orientation estimate. Particularly, the quality of altitude, roll and pitch estimates are adversely affected by an increase in distance between the camera and the marker.

From a geometric point of view, this makes sense as the four feature points of the tracked marker approach a single point in the image with increasing distance to the target. As the distance between the camera and marker increases, the area in the image occupied by the marker decreases. In this case, horizontal movement is easier to detect as it causes a translation of the marker in the image, whereas an altitude change when centered over the marker only result in a minor scaling of the marker in the image. Similarly, it is easier to detect a change in marker orientation relative to the camera if the marker is rotated

[1] Including NEES tuning, which is covered in Section 5.3.2.



(a) Orientation update component.



(b) Position update component.

Figure 5.9: NIS values with the final tuning for measurement and process noise covariance matrices.

around the optical axis. This corresponds to a yaw movement of the UAV when viewed directly from above, making the perceived feature points rotate in the image plane. A roll or pitch movement is in general more difficult to detect, because the perceived change in feature points in the image is similar to that of horizontal translation. When further away, the dilution of precision causes a less accurate pose calculation which affects the filter performance. If one were to adjust the $\mathbf{R}_k$ matrix based on the estimated distance to the landing target, one could limit the filter inconsistency caused by this effect. This was not done in this thesis, as the final values were considered tolerable in order for the filter to pass as consistent in terms of NIS.

5.3.2 NEES

The NIS filter consistency metric from Section 5.3.1 was divided into two components, because individual measurement models were used for orientation and position updates. The NEES values discussed in this section differ, as they are calculated for the entire pose estimated by the filter. Note that the NEES values are limited to a subset of the full state, which is motivated by two main factors: First, no ground truth data is available for the gyroscope and accelerometer biases. Secondly, neither the velocity nor the bias estimates are used when sending setpoints to the autopilot. Only the yaw angle and position estimate is used, but the NEES was computed over the full pose to be congruous with the NIS metric.

This means that six DOF are used for the 95% CI instead of three DOF for the respective NIS metrics. The NEES upper and lower bounds are thus $[l_{\mathcal{X}}(0.05) = 1.24, o_{\mathcal{X}}(0.05) = 14.45]$ according to Equation 3.36. In this implementation, the NEES also differs from the NIS by considering the cross terms between the position and orientation uncertainty of the state covariance matrix, which in general is nonzero because the measurements of both updates originate from the same sensor.

The motion capture system used for recording the ground truth was able to publish pose updates at $350\,\mathrm{Hz}$, but the NEES computation rate was matched to every filter update at $15\,\mathrm{Hz}$. The downside of doing this is that one does not fully encapsulate the estimation error built up during the prediction steps at $100\,\mathrm{Hz}$ between updates. However, the updates are performed at a relatively high rate to begin with so the accumulated error is in that sense limited. The accumulated error of dead reckoning is also encapsulated in the innovations used when computing the NIS. The reason for tying the NEES computation to the filter updates is that it better represents the quality of the landing target estimate sent to the autopilot. Given sufficient filter consistency, these are sent every filter update. This is when the filter estimate is at its best, which in some sense is a dataset bias that is important to consider when looking at the NEES values.

The NEES was tuned in a similar manner as the NIS, using one dataset for tuning and a different one for validation. The final NEES values on the validation data is shown in Figure 5.10. For the same reason as before, the final touchdown when landing was not included in the data. In general, the NEES values are noisier than the NIS values. This is likely due to the fact that both orientation and position errors contribute to a single metric as previously discussed. Of the NEES values, 94.7% lie inside the 95% CI. The resulting

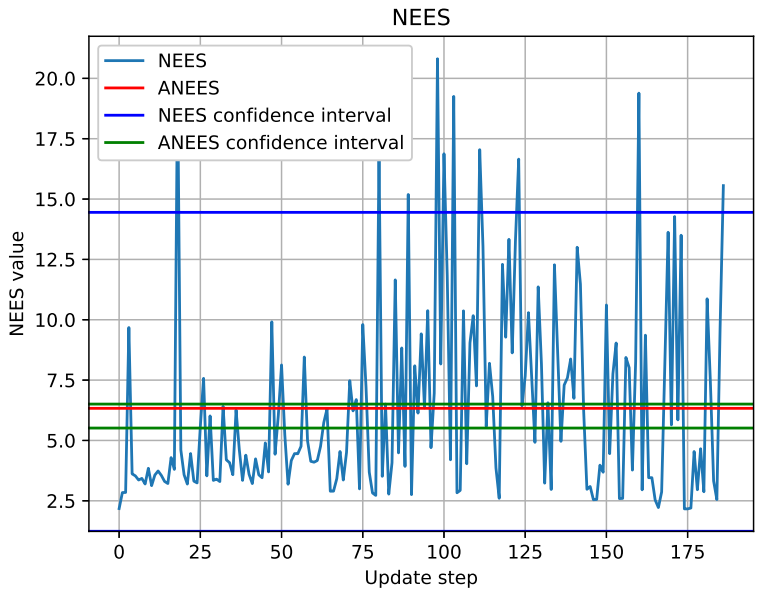


Figure 5.10: NEES values with the final tuning for measurement and process noise covariance matrices.

5.3% are above the upper bound. The ANEES is 6.33, which is considerably closer to the scaled upper bound of 6.51 than the lower bound at 5.51. Considering the values were sampled after the update, it may be desirable to further increase the measurement and process noise covariance to make the filter less confident. Either way, the filter is still passable as consistent. The main results in terms of filter consistency are summarized in Table 5.3.

Metric	Value	Lower bound	Upper bound
NIS inside CI, position component	87.2%	0.22	9.35
NIS inside CI, orientation component	98.5%	0.22	9.35
NEES inside CI, full pose	94.7%	1.24	14.45
ANIS, position component	3.15	2.67	3.35
ANIS, orientation component	3.04	2.67	3.35
ANEES, full pose	6.33	5.51	6.51

Table 5.3: Overview of key metrics for filter consistency, calculated with the final tuning over the validation dataset.

In both the NEES and NIS values for the tuned filter, it is clear that the filter converges quickly. The initial values are low due to the first covariance matrix being set high to prevent overconfidence. The filter converges in less than a second, which is due to the

high rate of updates and the log-linear property of the IEKF. This contributes to preserve energy in terms of hover time during landing. In this thesis, the collected data was representative of operational flight conditions in terms of flight duration, trajectory and when the estimation error was sampled. To obtain higher statistical significance for the consistency hypothesis, longer flights could have been conducted when collecting experimental data. Filter estimates for all prediction steps could have been logged and used for a more comprehensive NEES study. This should be considered if the filter is to be used for flight control directly, but was not further pursued in this thesis as the filter is limited to navigation aiding. In the case of the former, the data should also be collected to represent all flight conditions and not only those relevant to the landing sequence. Further tuning was not conducted in this thesis, because the desired filter performance and consistency was achieved given the intended use case. The final $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices used are shown in Equations 5.2.

$$\mathbf{Q}_k = \begin{bmatrix} 1.22 * 10^{-7}\mathbf{I} & 0 & 0 & 0 & 0 \\ 0 & 9.0 * 10^{-8}\mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0}_{3\times3} & 0 & 0 \\ 0 & 0 & 0 & 1.0 * 10^{-6}\mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & 1.0 * 10^{-6}\mathbf{I} \end{bmatrix} \tag{5.2a}$$

$$\mathbf{R}_k^{orientation} = \begin{bmatrix} 6.6 * 10^{-1} & 0 & 0 \\ 0 & 6.6 * 10^{-1} & 0 \\ 0 & 0 & 6.6 * 10^{-1} \end{bmatrix} \tag{5.2b}$$

$$\mathbf{R}_k^{position} = \begin{bmatrix} 1.5 * 10^{-1} & 0 & 0 \\ 0 & 1.5 * 10^{-1} & 0 \\ 0 & 0 & 6.0 * 10^{-1} \end{bmatrix} \tag{5.2c}$$

5.4 System Performance

The NIS and NEES metrics are useful for determining the filter consistency as defined in Section 3.6. The two metrics are essentially Mahalanobis distances of innovations and estimation errors respectively, which is useful for determining if the residuals are commensurable with their respective covariance matrices. In principle, it is possible for a filter with low estimation error to be inconsistent and vice versa. To analyze the full system performance, a series of test flights were conducted in a final experiment. The test flights were conducted using motion capture to record the UAV's movement, which served as a ground truth for performance assessment.

In this experiment, the filter estimates were used to aid the UAV during the landing sequence from a fixed initial position. This made it possible to analyze the repeatability and reliability of the system. During the experiments, the drone was set to idle at $4\,\mathrm{m}$ AGL for $1\,\mathrm{s}$ while centered horizontally above the landing target in order to let the filter converge. After the filter initialized, the landing target corrections were sent to the main autopilot. As a result, the UAV recentered itself above the marker according to the self-estimated

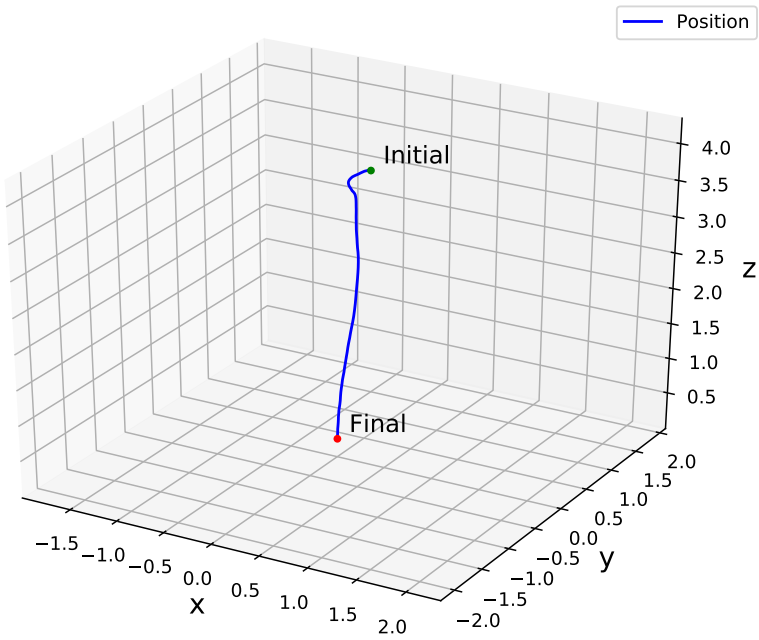


Figure 5.11: 3D visualization of single flight trajectory from an initial starting position to the landed state, as recorded by the motion capture system.

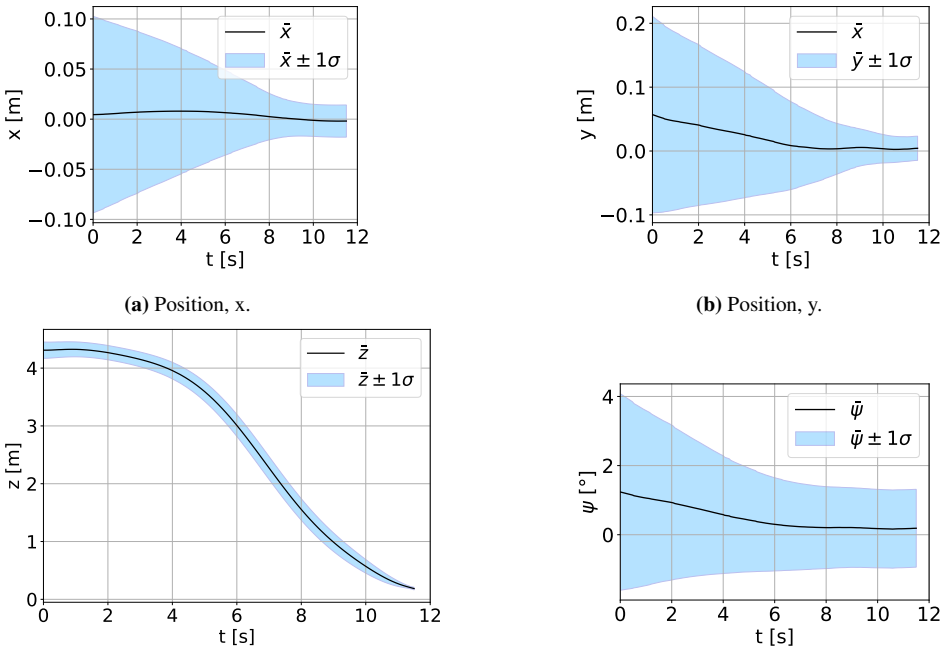
value before continuing the landing sequence. A 3D visualization of an example flight is shown in Figure 5.11. In total, 10 flights were conducted.

From the experimental results, the positioning and alignment errors can be used to analyze the filter performance and repeatability which in turn determines the overall reliability of the system. The position and yaw mean values and their respective standard deviations for all flights are shown as a function of time in Figure 5.12. It is clear that the pose estimation error grows with increased altitude above the landing target. This property is expected, as the filter updates are based on the relative pose calculation of the detected fiducial marker. As discussed, within the operating range the error of the pose calculation is expected to grow approximately linearly with the distance to the marker. Additionally, it is expected that the vertical accuracy is lower than the horizontal accuracy.

Spatial parameter	RMSE
Vertical position [cm]	23.3
Horizontal position [cm]	12.3
Yaw angle [°]	2.5

Table 5.4: The RMSE values for vertical position, horizontal position and yaw angle over all 10 sample flights as estimated by the IEKF.

The RMSE of the IEKF's yaw angle estimate, vertical and horizontal position are listed in



(a) Position, x.

(b) Position, y.

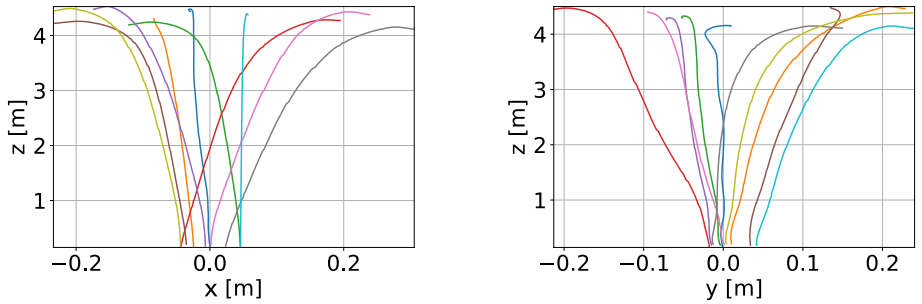
(c) Position, z.

(d) Orientation, yaw.

Figure 5.12: Mean value and standard deviation for position and yaw ground truths as a function of time over all 10 sample flights.

Table 5.4. The RMSE values confirm that the horizontal accuracy is lower than the vertical. The errors are in general consistent with the expected fiducial marker pose estimation accuracy present in [43] when considering the marker size and camera intrinsics used in this thesis.

There are several limitations to consider when analyzing the results of these experiments. First, the spatial limitations of the motion capture setup limited the upper flight volume to $4\,\mathrm{m}$ AGL. As discussed in Section 5.3.1, the filter is consistent in terms of NIS at the operational limits of $10\,\mathrm{m}$ AGL. Despite being consistent, the RMSE of the estimate is expected to grow with an increase in relative distance between the marker and the UAV. The values shown in Figure 5.12 are shown with respect to time. As the autopilot corrects for position or attitude disturbances, it contributes to a temporal offset between the trajectories in the experiment, which introduces noise. The time series shown from this experiment have therefore been cropped to match the shortest flight. For reference, the difference in duration of the longest and shortest flight was $0.4\,\mathrm{s}$. For a more realistic estimate of the system performance, the tests could have been conducted outside in windy conditions. If available, RTK GNSS could have been used as a ground truth. In this case, one would be limited of the $2.5\,\mathrm{cm}$ CEP of the u-blox M8P-2 GNSS module used which is significantly higher than the sub-millimetre accuracy of the motion capture system. Similarly to the filter consistency tuning, more test flights with longer flight times could also be used to



(a) Flight paths seen in the xz-plane.

(b) Flight paths seen in the yz-plane.

Figure 5.13: Side view of the sample test flights. The flight paths are ground truth position of the UAV after it is centered above the marker according to the onboard estimate.

Spatial parameter	Mean	Standard deviation
Horizontal displacement [cm]	0.0	3.5
Yaw misalignment [°]	0.2	1.9

Table 5.5: Mean and standard deviation of displacement in horizontal position and yaw misalignment when the UAV has landed. The values are calculated based on 10 test flights.

increase the statistical significance of the experimental results.

Despite the limitations, the results indicate a clear trend in the landing trajectories: During the final stage of the landing sequence, the estimation error decreases and the UAV position converges to the landing target center while the yaw angle is aligned with the marker. This behavior is apparent from Figure 5.13, demonstrating the systems ability to reliably land in the platform center with high precision. The displacement of the horizontal position and yaw angle as the UAV touched down were close to unbiased as shown in Figure 5.12 with standard deviations of $3.5\,\mathrm{cm}$ and $1.9°$ respectively. The values were sampled just as the vehicle touched the ground in order to not include the bounce which often occurs when physical contact with the ground is made. Recall that the mechanical tolerances of the landing platform are $102\,\mathrm{cm} \times 103\,\mathrm{cm}$ with a yaw misalignment tolerance of up to $35°$. The main results of the full system performance in terms of accuracy and precision when landed are shown in Table 5.5.

Assuming the horizontal position and yaw angle of the UAV when landing approximately follows a normal distribution, the UAV would in 99.7% of the cases land within a circle of radius $10.5\,\mathrm{cm}$ with a yaw error less than $5.7°$ according to the 3σ rule. In principle, this could be used to reduce the mechanical tolerances of the landing platform and thus its footprint. However, reducing the footprint would also make the outermost marker of $90\,\mathrm{cm} \times 90\,\mathrm{cm}$ infeasible with the current layout which constrains the range at which detections are possible. If one were to disable pixel binning and use the full $1920\,\mathrm{px} \times 1080\,\mathrm{px}$ image, the formula for maximum detection range shown in Equation 3.1 can be

used to determine the minimum marker width to reliably obtain detections at an altitude $10\,\mathrm{m}$ AGL. Given $p = 5$ pixels per bit, the minimum marker width in this case is found to be $36.4\,\mathrm{cm}$. This would also ensure contiguous detections in terms of overlap and estimated minimum detection range. Given sufficient lighting, this could be employed on future revisions of the landing platform to greatly reduce its footprint without changing the sensors on the UAV. Although there is no significant bias to horizontal position or yaw angle when landing, there is a bias initially. The yaw angle is approximately $1.9°$ on average whereas the initial y-coordinate is $-4\,\mathrm{cm}$. The bias is likely not caused by incorrect alignment and placement of the landing target, because it would then persist throughout the flight path. It is possible that this is caused by a change in the mechanical alignment of the camera relative to the IMU post calibration or camera imperfections and artifacts not accounted for by the mathematical model based on a pinhole camera model with radial-tangential distortion. In practice, this bias does not affect the convergence of the full system, but it should be addressed if the landing procedure is to be conducted in areas with nearby obstacles.

Chapter 6

Conclusion and future work

The concluding chapter of this thesis encompasses the presentation of conclusions and suggestions for future work. Initially, a comprehensive summary of the thesis work is provided, followed by the deduction of conclusions based on the initial problem definition. Subsequently, potential avenues for extending and enhancing the proposed system are explored and presented.

6.1 Conclusion

In this thesis, a precision landing system for UAVs based on camera and inertial measurements has been designed, implemented and tested in real-life experiments. The system relies on an IEKF for state estimation, using recursive fiducial markers on the landing platform for vision-based pose updates. The inertial process model is driven by IMU measurements, resulting in a standalone filter estimating the full extended pose of the UAV relative to the landing target in addition to IMU biases. The position and yaw estimates of the filter are used as aiding corrections for the autopilot. During the landing sequence, the UAV aligns itself in the horizontal plane based on the estimated center and heading of the landing platform. The vertical position estimate is then used for a smooth touchdown. The log-linear property of the invariant filter grants fast convergence around any trajectory, which reduces the time spent in hover during convergence and increases reliability. Sanity checks are applied to the raw measurements and filter outputs to ensure filter consistency during operation. In the event of estimation inconsistencies, the aiding corrections from the IEKF are halted. Depending on the failsafe settings, the autopilot is then guided back to a safe contingency landing site or the last place aiding corrections were obtained in an attempt to reacquire an estimate. The autopilot interface uses standardized MAVLink messages, making it possible to integrate the solution with existing autopilots such as PX4 or ArduPilot.

The individual system modules were tested sequentially, followed by a full system inte-

gration test with live data. First, the camera calibration was evaluated according to average reprojection error, image coverage and uncertainty of the intrinsic estimates. Thereafter, the AprilTag 3 detector was tested on real data using the estimated intrinsics. The results were evaluated according to pose calculation and marker detection time, detection recall and system load. The computation times were identified by their mean and maximum values over the entire flight trajectory, which were compared to the desired update rate of the IEKF. The computational load was based on average RAM and CPU usage for different configurations of the AprilTag detector. Then, the filter consistency was evaluated in terms of NIS and NEES values. The NIS and NEES values were used to tune the process and measurement noise covariance matrices until consistency was achieved. A data set from a validation flight was then used to identify the final consistency metrics. Lastly, the full system performance was identified by a series of test flights. The filter performance was evaluated according to the RMSE of the position and yaw estimates for all flights. Additionally, the reliability and repeatability of the full system was assessed by determining the mean and standard deviation of the ground truth flight trajectories from a given starting position from all test flights. The same analysis was then conducted for the ground truth data obtained when the drone was in the landed state to determine the landing accuracy and precision.

The precision landing system produces promising results in terms of aiding the UAV to the landed state - centered and aligned with the landing platform - from an uncertain initial position and heading relative to the platform. The landing procedure is conducted in a safe and efficient manner given the mechanical tolerances of the platform, with evidence of system repeatability and reliability. The conclusion is based on theoretical proofs of the estimator, specifically the log-linear property of the IEKF which leads to an autonomous error evolution and a convergence around any trajectory, in addition to empirical evidence gathered through experimental results. The precision and accuracy of the final landed state is sufficiently high, such that future revisions of the landing platform can be developed with lower tolerances for alignment and thus a smaller footprint. In the test flights, the RMSE of the filter was only estimated for a vertical position up to $4\,\mathrm{m}$ AGL. Although consistency is demonstrated up to $10\,\mathrm{m}$ AGL, the estimated error is expected to increase for larger altitudes due to the single marker pose measurement. This must be considered for operational use, since the proposed system assumes that there are no obstacles in the flight path. Generally, the significance of precision in the guiding system is diminished during flight, as it naturally converges to an accurate estimate when the UAV approaches the platform. Overall, the system shows good performance and it is the author's hope that it will be expanded and evolve into an open-source implementation compatible with the rich ecosystem of existing autopilots.

6.2 Future Work

In this section, potential paths for future academic research and application at a larger scale are outlined. The precision landing system proposed in this thesis is subject to several assumptions and delimitations which must be overcome for reliable usage outside a test environment. Furthermore, shortcomings of the design process should be addressed before

adopting the solution at a larger scale.

6.2.1 Camera Calibration and Filter Tuning

In the IEKF, the inertial process model is vehicle independent, because the dynamics are driven by the IMU inputs which are propagated to obtain the full extended pose. However, the filter tuning and calibration depend on the sensors used and can be cumbersome unless means of auto-tuning the filter are applied. It has not been investigated weather the calibration and tuning must be redone in the case of two or more UAVs with identical sensor configuration. In practice, the results are expected to be degraded in some degree due to manufacturing tolerances. Whether individual differences significantly impairs performance is not tested. In this case, means should be taken to reduce time and effort required for manual calibration and tuning. An implementation of [45] based on Bayesian Optimization is suggested to replace manual tuning, which was found to be the most time consuming of the two.

6.2.2 Delimitations, Limitations and Assumptions

Currently, the precision landing method assumes an obstacle free landing site in which the fiducial markers are visible. As discussed, the estimation error is expected to increase with altitude, resulting in a larger obstacle-free buffer required. In practice, this is typically achieved for fixed landing stations. In the event of a forwarded temporary platform, the presence of airspace obstructions such as trees or masts might be a problem. To determine the required buffer, experimental flights should be conducted at the target altitude with RTK GNSS other other means of an approximate ground truth. One could also arrive at a rudimentary approximation of the buffer by repeating the experiments conducted in this thesis with a scaled-down marker in a motion capture environment or extrapolate from the given results. Second, the assumption of landing target visibility is not guaranteed with the current setup. Occlusion due to precipitation, leaves or dust will eventually happen if not mitigated. The same holds true for lighting, which affects the detectors capabilities. To reduce the possibility of occlusion, it is suggested that permanent installations of landing platforms are equipped with roofing which only opens during the brief time period the UAV is landing. The landing target itself could also be heated, which may be required if landing during heavy snowfall. In this thesis, experiments were conducted in dimly lit conditions to represent the intended operational environment. This proved not to be an issue for the onboard low-light camera, but installing lights on the landing platform is considered a simple approach to remove this hardware dependency and increase overall reliability.

6.2.3 System Improvements

The current implementation features a working solution proposal, but there are likely aspects which can be improved. First, it is believed that filter consistency would be improved by adjusting the measurement noise covariance matrix based on the estimated distance to the landing target. The pose measurement error increases with the distance to the marker or a decrease in marker size, which should be reflected by an increase in the covariance

matrix. Tuning the filter according to this relation was not pursued in this thesis, but would likely improve consistency of the filter and contribute to a more accurate covariance estimate.

The IPPE solution to the PnP problem used in this system is subject to ambiguity. This implementation always uses the pose estimate with the lowest reprojection error. In the case of a measurement inconsistent with the current estimate or previous measurements, it is discarded. The presence of pose ambiguity did not yield any significant issues in the conducted experiments. However, it is advisable to explore methods that can restrict the range of marker poses in order to enhance the level of reliability and bolster the assurance of accurate results. The ambiguity is a property of the problem itself and cannot be solved using any PnP algorithm. The original paper [23] suggests four main options. In the intended use case, using additional non-coplanar markers mounted on the landing platform would make the solution to the pose estimate unique.

In the proposed system, no manual adjustment of the camera settings was performed. In total, 14 settings can be adjusted including, but not limited to: Brightness, contrast, saturation and exposure. These were set to the default values or auto if available. Considering that the detector operates on grayscale images, the camera settings could likely be adjusted to trade off accurate color representation for improved low-light performance. One could also disable pixel binning to increase resolution, resulting in a higher detection accuracy with an increased detection range, albeit with a trade-off in terms of diminished low-light performance and a higher computational load. It is also possible that one could fine-tune the exposure such that desired low-light performance would be achieved, while reducing rotational blur. The same holds true for the camera focus, which had to be set manually. Since the scene depth varies from $14\,\mathrm{cm}$ to $10\,\mathrm{m}$, one would ideally use automatic focus to keep the landing target sharp while maintaining a high FOV. This can be challenging in computer vision applications, due to the dynamic nature of the resulting camera intrinsics. In this thesis, the focus was set to compromise the desired sharpness over the flight path.

The system is designed with existing autopilot compatibility in mind, but it has currently only been tested with PX4. In principle, other platforms such as ArduPilot would also be supported. This has not been verified and there is likely some work to be done before full compatibility with both systems is achieved. Section 5.2 showed that the distance to the marker and thus the marker size in the image greatly affected the detection speed. For this reason, one might benefit from performing the marker detection on a cropped image which reflects a region of interest based on the current pose estimate and previous detections. If no detections were made, the full image could be used instead. Being the most time consuming part of the pipeline, this is a natural starting point for increasing performance. Considering the achieved landing accuracy and precision, the mechanical tolerances of the landing platform could be reduced significantly in order to reduce the footprint. This would in turn reduce the free space available for landing target placement. For a scaled down landing platform accommodated by a smaller landing target, disabling binning would likely make the marker detectable from an altitude of $10\,\mathrm{m}$ AGL with a landing target width of only $36.4\,\mathrm{cm}$. This can be achieved with the sensors currently used on the UAV and greatly reduce the footprint and complexity of future revisions of the platform. If this approach is to be pursed, two things must be considered: First, the

low-light performance will be significantly impaired, meaning more illumination on the landing platform is required. Second, the computational load for marker detection will increase beyond the currently available system capacity. To accommodate this, one could disable tolerance for erroneous marker bits and apply the region of interest based detection as formerly discussed.

Lastly, the IEKF used in this thesis can be improved and repurposed for other applications. For example, the filter and the autopilot interface could be expanded to support data from multiple IMUs. Incorporating this approach would result in a reduction of noise present in acceleration and angular velocity measurements, including a more precise estimation of the biases. In principle, one could also use the bias estimates of the autopilots EKF to hot start the IEKF. There are two primary reasons for the absence of additional coupling between the two filters in this thesis: Firstly, to emphasize the rapid convergence capabilities of the IEKF in its existing form. Secondly, to establish the viability of the IEKF as an independent filter. In the context of state estimation for navigation, the state representation usually consists of and extended pose which can be represented in the Lie group framework. In this case, the IEKF outperforms the EKF by ensuring the filter update remains on the Lie manifold. The EKF fails to adhere to the geometric constraints of the attitude representation by linearization, making the estimate leave the manifold. Although the EKF in many ways has become industry standard, the IEKF should be considered for high-accuracy navigation purposes as it improves reliability, reduces errors and has a low convergence time. This could be used in autopilots for state estimation or as a backend for Visual-Inertial Odometry (VIO) or SLAM applications. If the IEKF used in this thesis is to be expanded to the primary filter for navigation, a more comprehensive tuning should be conducted. Specifically, more aggressive maneuvers should be applied to represent the full range of expected in-flight movement including large deviations from non-zero orientation. For this purpose, a more exhaustive study of all state estimates should be conducted while comparing the results to the output of a conventional EKF or MEKF for the given platform.

Bibliography

[1] L. Meier, D. Honegger and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms", in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[2] MAVLink contributors, *MAVLink Common Message Set*, Dronecode Project, 2023. [Online]. Available: https://mavlink.io/en/messages/common.html (visited on 21/05/2023).

[3] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[4] M. Krogius, A. Haggenmiller and E. Olson, "Flexible layouts for fiducial tags", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2019.

[5] J. Sola, J. Deray and D. Atchuthan, "A micro Lie theory for state estimation in robotics", 2018. arXiv: 1812.01537 [cs.RO].

[6] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer", *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2016. DOI: 10.1109/TAC.2016.2594085.

[7] P. Chauchat, "Smoothing algorithms for navigation, localisation and mapping based on high-grade inertial sensors", Thesis, Université Paris sciences et lettres, Feb. 2020. [Online]. Available: https://pastel.archives-ouvertes.fr/tel-02887295 (visited on 21/05/2023).

[8] A. Barrau and S. Bonnabel, "Intrinsic filtering on Lie groups with applications to attitude estimation", *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 436–449, 2015. DOI: 10.1109/TAC.2014.2342911.

[9] R. Howe, "Very basic Lie theory", *The American Mathematical Monthly*, vol. 90, no. 9, pp. 600–623, 1983. DOI: 10.2307/2323277.

[10] R. E. Kalman *et al.*, "Contributions to the theory of optimal control", *Bol. soc. mat. mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[11] T. D. Barfoot, *State estimation for robotics*. Cambridge, U.K.: Cambridge University Press, 2017. DOI: 10.1017/9781316671528.

[12] E. Brekke, *Fundamentals of Sensor Fusion*, 2020. [Online]. Available: https://folk.ntnu.no/edmundfo/msc2020-2021/sf2020.pdf (visited on 01/06/2023).

[13] M. Boutayeb, H. Rafaralahy and M. Darouach, "Convergence analysis of the extended Kalman filter used as an observer for nonlinear deterministic discrete-time systems", *IEEE Transactions on Automatic Control*, vol. 42, no. 4, pp. 581–586, 1997. DOI: 10.1109/9.566674.

[14] A. J. Krener, "The convergence of the extended Kalman filter", 2002. arXiv: math/0212255 [math.OC].

[15] P. Martin and E. Salaün, "Generalized multiplicative extended Kalman filter for aided attitude and heading reference system", in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 8300. DOI: 10.2514/6.2010-8300.

[16] S. Bonnable, P. Martin and E. Salaün, "Invariant extended Kalman filter: Theory and application to a velocity-aided attitude estimation problem", in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, 2009, pp. 1297–1304. DOI: 10.1109/CDC.2009.5400372.

[17] M. Barczyk, S. Bonnabel, J.-E. Deschaud and F. Goulette, "Invariant EKF design for scan matching-aided localization", *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2440–2448, 2015. DOI: 10.1109/TCST.2015.2413933.

[18] R. Szeliski, *Computer Vision: Algorithms and Applications* (Texts in Computer Science), 2nd ed. Springer Cham, 2022, ISBN: 978-3-030-34374-3. DOI: 10.1007/978-3-030-34372-9.

[19] Y. Ma, S. Soatto, J. Košecká and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004, vol. 26, ISBN: 978-1-4419-1846-8. DOI: 10.1007/978-0-387-21779-6.

[20] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. DOI: 10.1109/34.888718.

[21] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2004. DOI: 10.1017/CBO9780511811685.

[22] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, ISBN: 9781421407944. DOI: 10.56021/9781421407944.

[23] T. Collins and A. Bartoli, "Infinitesimal plane-based pose estimation", *International Journal of Computer Vision*, vol. 109, no. 3, pp. 252–286, 2014, ISSN: 0920-5691. DOI: 10.1007/s11263-014-0725-5.

[24] ArduPilot contributors, *ArduPilot open source autopilot system*, ArduPilot, 2023. [Online]. Available: https://ardupilot.org/ (visited on 21/05/2023).

[25] *NEO-M8P datasheet*, UBX-15016656, R11, u-blox, 2022. [Online]. Available: https://content.u-blox.com/sites/default/files/NEO-M8P_DataSheet_UBX-15016656.pdf (visited on 21/05/2023).

[26] QGroundControl contributors, *QGroundControl flight control and mission planning*, QGroundControl, 2023. [Online]. Available: http://qgroundcontrol.com/ (visited on 21/05/2023).

[27] C. B. Duane, "Close-range camera calibration", *Photogramm. Eng*, vol. 37, no. 8, pp. 855–866, 1971.

[28] Blue Robotics Crew, *Low-Light HD USB Camera*, Blue Robotics Inc., 2023. [Online]. Available: https://bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/ (visited on 27/03/2023).

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation", *International journal of computer vision*, vol. 59, pp. 167–181, 2004.

[30] R. Hartley, M. Ghaffari, R. M. Eustice and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation", *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020. DOI: 10.1177/0278364919894385.

[31] V. Kull, "Invariant extended Kalman filter for measurements on Lie groups", M.S. thesis, KTH Royal Institute of Technology, 2021. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:1632658/FULLTEXT01.pdf (visited on 27/03/2023).

[32] Y. Bar-Shalom, X. R. Li and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, 2001. DOI: 10.1002/0471221279.

[33] Y. Pyo, H. Cho, L. Jung and D. Lim, *ROS Robot Programming*. ROBOTIS, 2017, ISBN: 9791196230715. [Online]. Available: http://community.robotsource.org/t/download-the-ros-robot-programming-book-for-free/51 (visited on 21/05/2023).

[34] MAVLink router contributors, *MAVLink router*, GitHub repository, 2023. [Online]. Available: https://github.com/mavlink-router/mavlink-router (visited on 21/05/2023).

[35] GStreamer Team, *GStreamer pipeline-based multimedia framework*, GStreamer, 2023. [Online]. Available: https://gstreamer.freedesktop.org/ (visited on 21/05/2023).

[36] E. R. Potokar, K. Norman and J. G. Mangelson, "Invariant extended Kalman filtering for underwater navigation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5792–5799, 2021. DOI: 10.1109/LRA.2021.3085167.

[37] G. Guennebaud, B. Jacob *et al.*, *Eigen v3*, 2010. [Online]. Available: http://eigen.tuxfamily.org (visited on 21/05/2023).

[38] W. Jakob, *pybind11*, GitHub repository, 2022. [Online]. Available: https://github.com/pybind/pybind11 (visited on 21/05/2023).

[39] calib.io ApS, *Kalibr Targets*, 2023. [Online]. Available: https://calib.io/products/kalibr-targets (visited on 20/04/2023).

[40] P. Furgale, H. Sommer, J. Maye, J. Rehder, T. Schneider and L. Oth, *Kalibr*, GitHub repository, 2023. [Online]. Available: https://github.com/ethz-asl/kalibr (visited on 21/05/2023).

[41] C. N. AS, *Alu plate*, 2023. [Online]. Available: https://www.cewe.no/veggbilder/alu-plate.html (visited on 24/04/2023).

[42] J. Kallwies, B. Forkel and H.-J. Wuensche, "Determining and improving the localization accuracy of AprilTag detection", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 8288–8294. DOI: 10.1109/ICRA40945.2020.9197427.

[43] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead and N. Vitzilaios, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the ARTag, AprilTag, ArUco and STag markers", *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–26, 2021. DOI: 10.1007/s10846-020-01307-9.

[44] A. Gelb *et al.*, *Applied optimal estimation*. MIT press, 1974.

[45] Z. Chen, C. Heckman, S. Julier and N. Ahmed, "Weak in the NEES?: Auto-tuning Kalman filters with Bayesian optimization", in *2018 21st International Conference on Information Fusion (FUSION)*, IEEE, 2018, pp. 1072–1079. DOI: 10.23919/ICIF.2018.8454982.

Appendices

Appendix A

Matrix Lie Group Jacobians

A.1 $SO(3)$

Given $\boldsymbol{\theta} = \mathrm{Log}(\mathbf{M}),\ \mathbf{M} \in SO(3)$, the right Jacobian and its inverse [5] are given by

$$\mathbf{J}_r(\boldsymbol{\theta}) = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2 \tag{A.1a}$$

$$\mathbf{J}_r^{-1}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1}{2}[\boldsymbol{\theta}]_\times + (\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta})[\boldsymbol{\theta}]_\times^2 \tag{A.1b}$$

The left Jacobian and its inverse are

$$\mathbf{J}_l(\boldsymbol{\theta}) = \mathbf{J}_r^\top(\boldsymbol{\theta}) \tag{A.2a}$$

$$\mathbf{J}_l^{-\top}(\boldsymbol{\theta}) = \mathbf{J}_r^{-\top}(\boldsymbol{\theta}) \tag{A.2b}$$

A.2 $SE(3)$

Given $\boldsymbol{\tau} = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\theta}\end{bmatrix} = \mathrm{Log}(\mathbf{M}),\ \mathbf{M} \in SE(3)$, the left Jacobian and its inverse are given by

$$\mathbf{J}_l(\boldsymbol{\tau}) = \begin{bmatrix}\mathbf{J}_l(\boldsymbol{\theta}) & \mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\theta})\\ \mathbf{0}_{3\times3} & \mathbf{J}_l(\boldsymbol{\theta})\end{bmatrix} \tag{A.3a}$$

$$\mathbf{J}_l^{-1}(\boldsymbol{\tau}) = \begin{bmatrix}\mathbf{J}_l^{-1}(\boldsymbol{\theta}) & -\mathbf{J}_l^{-1}(\boldsymbol{\theta})\mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\theta})\mathbf{J}_l^{-1}(\boldsymbol{\theta})\\ \mathbf{0}_{3\times3} & \mathbf{J}_l^{-1}(\boldsymbol{\theta})\end{bmatrix} \tag{A.3b}$$

where

$$\begin{aligned}\mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\theta}) = \frac{1}{2}[\boldsymbol{\rho}]_\times &+ \frac{\theta-\sin\theta}{\theta^3}([\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times + [\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times + [\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times) \\ &+ \frac{\frac{\theta^2}{2}+\cos\theta-1}{\theta^4}([\boldsymbol{\theta}]^2_\times[\boldsymbol{\rho}]_\times + [\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]^2_\times - 3[\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times) \\ &+ \frac{\theta+\frac{\theta}{2}\cos\theta-\frac{3}{2}\sin\theta}{\theta^5}([\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]^2_\times + [\boldsymbol{\theta}]^2_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times)\end{aligned} \tag{A.4}$$

The right Jacobian and its inverse are

$$\mathbf{J}_r(\boldsymbol{\tau}) = \mathbf{J}_l(-\boldsymbol{\tau}) \tag{A.5a}$$
$$\mathbf{J}_r^{-1}(\boldsymbol{\tau}) = \mathbf{J}_l^{-1}(-\boldsymbol{\tau}) \tag{A.5b}$$

A.3 $SE_2(3)$

Given $\boldsymbol{\tau} = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\nu}\\ \boldsymbol{\theta}\end{bmatrix} = \mathrm{Log}(\mathbf{M}),\ \mathbf{M} \in SE(3)$, the left Jacobian and its inverse are given by

$$\mathbf{J}_l(\boldsymbol{\tau}) = \begin{bmatrix}\mathbf{J}_l(\boldsymbol{\theta}) & \mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\nu}) & \mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\theta}) \\ \mathbf{0}_{3\times3} & \mathbf{J}_l(\boldsymbol{\theta}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{J}_l(\boldsymbol{\theta})\end{bmatrix} \tag{A.6a}$$

$$\mathbf{J}_l^{-1}(\boldsymbol{\tau}) = \begin{bmatrix}\mathbf{J}_l^{-1}(\boldsymbol{\theta}) & -\mathbf{J}_l^{-1}(\boldsymbol{\theta})\mathbf{Q}(\boldsymbol{\nu},\boldsymbol{\theta})\mathbf{J}_l^{-1}(\boldsymbol{\theta}) & -\mathbf{J}_l^{-1}(\boldsymbol{\theta})\mathbf{Q}(\boldsymbol{\rho},\boldsymbol{\theta})\mathbf{J}_l^{-1}(\boldsymbol{\theta}) \\ \mathbf{0}_{3\times3} & \mathbf{J}_l^{-1}(\boldsymbol{\theta}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{J}_l^{-1}(\boldsymbol{\theta})\end{bmatrix} \tag{A.6b}$$

The right Jacobian and its inverse are

$$\mathbf{J}_r(\boldsymbol{\tau}) = \mathbf{J}_l(-\boldsymbol{\tau}) \tag{A.7a}$$
$$\mathbf{J}_r^{-1}(\boldsymbol{\tau}) = \mathbf{J}_l^{-1}(-\boldsymbol{\tau}) \tag{A.7b}$$

Appendix B

MAVLink message definitions

In this chapter, the MAVLink message definitions most relevant to this thesis are presented. For a more extensive compilation of commonly used messages and the requirements mandated by the MAVLink standard, please refer to [2].

HEARTBEAT (#0)

[Message] The heartbeat message shows that a system or component is present and responding. The type and autopilot fields (along with the message component id), allow the receiving system to treat further messages from this system appropriately (e.g. by laying out the user interface based on the autopilot). This microservice is documented at https://mavlink.io/en/services/heartbeat.html

Field Name	Type	Values	Description
type	uint8_t	MAV_TYPE	Vehicle or component type. For a flight controller component the vehicle type (quadrotor, helicopter, etc.). For other components the component type (e.g. camera, gimbal, etc.). This should be used in preference to component id for identifying the component type.
autopilot	uint8_t	MAV_AUTOPILOT	Autopilot type / class. Use MAV_AUTOPILOT_INVALID for components that are not flight controllers.
base_mode	uint8_t	MAV_MODE_FLAG	System mode bitmap.
custom_mode	uint32_t		A bitfield for use for autopilot-specific flags
system_status	uint8_t	MAV_STATE	System status flag.
mavlink_version	uint8_t_mavlink_version		MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version

Figure B.1: MAVLink heartbeat message definition.

HIGHRES_IMU (#105)

[Message] The IMU readings in SI units in NED body frame

Field Name	Type	Units	Values	Description
time_usec	uint64_t	us		Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude of the number.
xacc	float	m/s/s		X acceleration
yacc	float	m/s/s		Y acceleration
zacc	float	m/s/s		Z acceleration
xgyro	float	rad/s		Angular speed around X axis
ygyro	float	rad/s		Angular speed around Y axis
zgyro	float	rad/s		Angular speed around Z axis
xmag	float	gauss		X Magnetic field
ymag	float	gauss		Y Magnetic field
zmag	float	gauss		Z Magnetic field
abs_pressure	float	hPa		Absolute pressure
diff_pressure	float	hPa		Differential pressure
pressure_alt	float			Altitude calculated from pressure
temperature	float	degC		Temperature
fields_updated	uint16_t		HIGHRES_IMU_UPDATED_FLAGS	Bitmap for fields that have updated since last message
id **	uint8_t			Id. Ids are numbered from 0 and map to IMUs numbered from 1 (e.g. IMU1 will have a message with id=0)

Figure B.2: MAVLink high resolution IMU message definition.

LANDING_TARGET (#149)

[Message] The location of a landing target. See: https://mavlink.io/en/services/landing_target.html

Field Name	Type	Units	Values	Description
time_usec	uint64_t	us		Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude of the number.
target_num	uint8_t			The ID of the target if multiple targets are present
frame	uint8_t		MAV_FRAME	Coordinate frame used for following fields.
angle_x	float	rad		X-axis angular offset of the target from the center of the image
angle_y	float	rad		Y-axis angular offset of the target from the center of the image
distance	float	m		Distance to the target from the vehicle
size_x	float	rad		Size of target along x-axis
size_y	float	rad		Size of target along y-axis
x **	float	m		X Position of the landing target in MAV_FRAME
y **	float	m		Y Position of the landing target in MAV_FRAME
z **	float	m		Z Position of the landing target in MAV_FRAME
q **	float[4]			Quaternion of landing target orientation (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
type **	uint8_t		LANDING_TARGET_TYPE	Type of landing target
position_valid **	uint8_t			Boolean indicating whether the position fields (x, y, z, q, type) contain valid target position information (valid: 1, invalid: 0). Default is 0 (invalid).

Figure B.3: MAVLink landing target message definition.

Appendix C

IMU gyroscope and accelerometer specifications

The full gyroscope and accelerometer specifications given by their respective datasheets of the TDK InvenSense ICM-20649, ICM-20602 and ICM-20948 featured in the CubePilot Cube Orange autopilot are listed below. The key properties are listed in Table 4.1, along with a link to the full datasheet from which these sections are extracted.

C.1 ICM-20649

3 ELECTRICAL CHARACTERISTICS

3.1 GYROSCOPE SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8 V, VDDIO = 1.8 V, T_A = 25°C, unless otherwise noted.

Note: All specifications apply to Standard (Duty-Cycled) Mode and Low-Noise Mode, unless noted otherwise.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	GYRO_FS_SEL = 0		±500		dps	1
	GYRO_FS_SEL = 1		±1000		dps	1
	GYRO_FS_SEL = 2		±2000		dps	1
	GYRO_FS_SEL = 3		±4000		dps	1
Gyroscope ADC Word Length			16		bits	1
Sensitivity Scale Factor	GYRO_FS_SEL = 0		65.5		LSB/(dps)	1
	GYRO_FS_SEL = 1		32.8		LSB/(dps)	1
	GYRO_FS_SEL = 2		16.4		LSB/(dps)	1
	GYRO_FS_SEL = 3		8.2		LSB/(dps)	1
Sensitivity Scale Factor Tolerance	25°C		±0.5		%	3
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±2		%	2
Nonlinearity	Best fit straight line; 25°C		±0.1		%	2, 4
Cross-Axis Sensitivity			±2		%	2, 4
ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C (Component-level)		±5		dps	3
ZRO Variation Over Temperature	-40°C to +85°C		±0.05		dps/°C	2
GYROSCOPE NOISE PERFORMANCE (GYRO_FS_SEL=0)						
Noise Spectral Density	Based on Noise Bandwidth = 10 Hz		0.0175		dps/√Hz	3
GYROSCOPE MECHANICAL FREQUENCIES		25	27	29	kHz	3
LOW PASS FILTER RESPONSE	Programmable Range	5.7		197	Hz	1, 4
GYROSCOPE START-UP TIME	From Full-Chip Sleep mode		35		ms	2, 4
OUTPUT DATA RATE	Standard (duty-cycled) Mode	4.4		562.5	Hz	1
	Low-Noise Mode GYRO_FCHOICE = 1; GYRO_DLPFCFG = x	4.4		1.125k	Hz	
	Low-Noise Mode GYRO_FCHOICE = 0; GYRO_DLPFCFG = x			9k	Hz	

Table 1. Gyroscope Specifications

Notes:
1. Guaranteed by design.
2. Derived from validation or characterization of parts, not guaranteed in production.
3. Tested in production.
4. Low-noise mode specification.

3.2 ACCELEROMETER SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8 V, VDDIO = 1.8 V, T_A = 25°C, unless otherwise noted.

Note: All specifications apply to Standard (Duty-Cycled) Mode and Low-Noise Mode, unless noted otherwise.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	ACCEL_FS = 0		±4		*g*	1
	ACCEL_FS = 1		±8		*g*	1
	ACCEL_FS = 2		±16		*g*	1
	ACCEL_FS = 3		±30		*g*	1
ADC Word Length	Output in two's complement format		16		bits	1
Sensitivity Scale Factor	ACCEL_FS = 0		8,192		LSB/*g*	1
	ACCEL_FS = 1		4,096		LSB/*g*	1
	ACCEL_FS = 2		2,048		LSB/*g*	1
	ACCEL_FS = 3		1,024		LSB/*g*	1
Initial Tolerance	Component-level		±0.5		%	3
Sensitivity Change vs. Temperature	-40°C to +85°C ACCEL_FS=0		±0.026		%/ºC	2
Nonlinearity	Best Fit Straight Line		±0.5		%	2, 4
Cross-Axis Sensitivity			±2		%	2, 4
Initial Tolerance	Component-level, all axes		±65		mg	3
Zero-G Level Change vs. Temperature	0°C to +85°C		±0.80		mg/°C	2
ACCELEROMETER NOISE PERFORMANCE						
Noise Spectral Density	Based on Noise Bandwidth = 10 Hz		285		µg/VHz	3
LOW PASS FILTER RESPONSE	Programmable Range	5.7		246	Hz	1, 4
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	1
ACCELEROMETER STARTUP TIME	From Sleep mode		20		ms	2, 4
	From Cold Start, 1ms V_{DD} ramp		30		ms	2, 4
OUTPUT DATA RATE	Low-Power Mode	0.27		562.5	Hz	1
	Low-Noise Mode ACCEL_FCHOICE = 1; ACCEL_DLPFCFG = x	4.5		1.125k	Hz	
	Low-Noise Mode ACCEL_FCHOICE = 0; ACCEL_DLPFCFG = x			4.5k	Hz	

Table 2. Accelerometer Specifications

Notes:
1. Guaranteed by design.
2. Derived from validation or characterization of parts, not guaranteed in production.
3. Tested in production.
4. Low-noise mode specification.

C.2 ICM-20602

3 ELECTRICAL CHARACTERISTICS

3.1 GYROSCOPE SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8V, VDDIO = 1.8V, T_A=25°C, unless otherwise noted.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		dps	3
	FS_SEL=1		±500		dps	3
	FS_SEL=2		±1000		dps	3
	FS_SEL=3		±2000		dps	3
Gyroscope ADC Word Length			16		bits	3
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(dps)	3
	FS_SEL=1		65.5		LSB/(dps)	3
	FS_SEL=2		32.8		LSB/(dps)	3
	FS_SEL=3		16.4		LSB/(dps)	3
Sensitivity Scale Factor Initial Tolerance	25°C		±1		%	1
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±2		%	1
Nonlinearity	Best fit straight line; 25°C		±0.1		%	1
Cross-Axis Sensitivity			±1		%	1
ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±1		dps	1
ZRO Variation vs. Temperature	-40°C to +85°C		±0.01		dps/ºC	1
OTHER PARAMETERS						
Rate Noise Spectral Density	@ 10 Hz		0.004		dps /√Hz	1, 4
Total RMS Noise	Bandwidth = 100 Hz		0.04		dps -rms	1, 4
Gyroscope Mechanical Frequencies		25	27	29	KHz	2
Low Pass Filter Response	Programmable Range	5		250	Hz	3
Gyroscope Start-Up Time	Time from gyro enable to gyro drive ready		35	100	ms	1
Output Data Rate	Low-Noise mode	3.91		8000	Hz	3
	Low Power Mode	3.91		333.33	Hz	3

Table 1. Gyroscope Specifications

Notes:

1. Derived from validation or characterization of parts, not guaranteed in production.
2. Tested in production.
3. Guaranteed by design.
4. Noise specifications shown are for low-noise mode.

3.2 ACCELEROMETER SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8V, VDDIO = 1.8V, T_A=25°C, unless otherwise noted.

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY							
Full-Scale Range	AFS_SEL=0			±2		*g*	2
	AFS_SEL=1			±4		*g*	2
	AFS_SEL=2			±8		*g*	2
	AFS_SEL=3			±16		*g*	2
ADC Word Length	Output in two's complement format			16		bits	2
Sensitivity Scale Factor	AFS_SEL=0			16,384		LSB/*g*	2
	AFS_SEL=1			8,192		LSB/*g*	2
	AFS_SEL=2			4,096		LSB/*g*	2
	AFS_SEL=3			2,048		LSB/*g*	2
Sensitivity Scale Factor Initial Tolerance	Component-level			±1		%	1
Sensitivity Change vs. Temperature	-40°C to +85°C			±1.5		%	1
Nonlinearity	Best Fit Straight Line			±0.3		%	1
Cross-Axis Sensitivity				±1		%	1
ZERO-G OUTPUT							
Initial Tolerance	Component-level, all axes			±25		*mg*	1
	Board-level, all axes			±40		*mg*	1
Zero-G Level Change vs. Temperature	-40°C to +85°C	X and Y axes		±0.5		mg/ºC	1
		Z axis		±1		mg/ºC	1
OTHER PARAMETERS							
Power Spectral Density	@ 10 Hz			100		µg/√Hz	1, 3
RMS Noise	Bandwidth = 100 Hz			1.0		mg-rms	1, 3
Low-Pass Filter Response	Programmable Range		5		218	Hz	2
Accelerometer Startup Time	From sleep mode to valid data			10	20	ms	2
Output Data Rate	Low-Noise mode		3.91		4000	Hz	2
	Low Power Mode		3.91		500	Hz	

Table 2. Accelerometer Specifications

Notes:

1. Derived from validation or characterization of parts, not guaranteed in production.
2. Guaranteed by design.
3. Noise specifications shown are for low-noise mode.

C.3 ICM-20948

3 ELECTRICAL CHARACTERISTICS

3.1 GYROSCOPE SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8V, VDDIO = 1.8V, T_A=25°C, unless otherwise noted.

NOTE: All specifications apply to Low-Power Mode and Low-Noise Mode, unless noted otherwise

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	GYRO_FS_SEL=0		±250		dps	1
	GYRO_FS_SEL=1		±500		dps	1
	GYRO_FS_SEL=2		±1000		dps	1
	GYRO_FS_SEL=3		±2000		dps	1
Gyroscope ADC Word Length			16		bits	1
Sensitivity Scale Factor	GYRO_FS_SEL=0		131		LSB/(dps)	1
	GYRO_FS_SEL=1		65.5		LSB/(dps)	1
	GYRO_FS_SEL=2		32.8		LSB/(dps)	1
	GYRO_FS_SEL=3		16.4		LSB/(dps)	1
Sensitivity Scale Factor Tolerance	25°C		±1.5		%	2
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±3		%	2
Nonlinearity	Best fit straight line; 25°C		±0.1		%	2, 3
Cross-Axis Sensitivity			±2		%	2, 3
ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C (Component-level)		±5		dps	2
ZRO Variation Over Temperature	-40°C to +85°C		±0.05		dps/°C	2
GYROSCOPE NOISE PERFORMANCE (GYRO_FS_SEL=0)						
Noise Spectral Density	Based on Noise Bandwidth = 10 Hz		0.015		dps/√Hz	2
GYROSCOPE MECHANICAL FREQUENCIES		25	27	29	kHz	2
LOW PASS FILTER RESPONSE	Programmable Range	5.7		197	Hz	1, 3
GYROSCOPE START-UP TIME	From Full-Chip Sleep mode		35		ms	2, 3
OUTPUT DATA RATE	Low-Power Mode	4.4		562.5	Hz	1
	Low-Noise Mode GYRO_FCHOICE=1; GYRO_DLPFCFG=x	4.4		1.125k	Hz	
	Low-Noise Mode GYRO_FCHOICE=0; GYRO_DLPFCFG=x			9k	Hz	

Table 1. Gyroscope Specifications

NOTES:

1. Guaranteed by design.
2. Derived from validation or characterization of parts, not guaranteed in production.
3. Low-noise mode specification.

3.2 ACCELEROMETER SPECIFICATIONS

Typical Operating Circuit of section 4.2, VDD = 1.8V, VDDIO = 1.8V, T_A=25°C, unless otherwise noted.

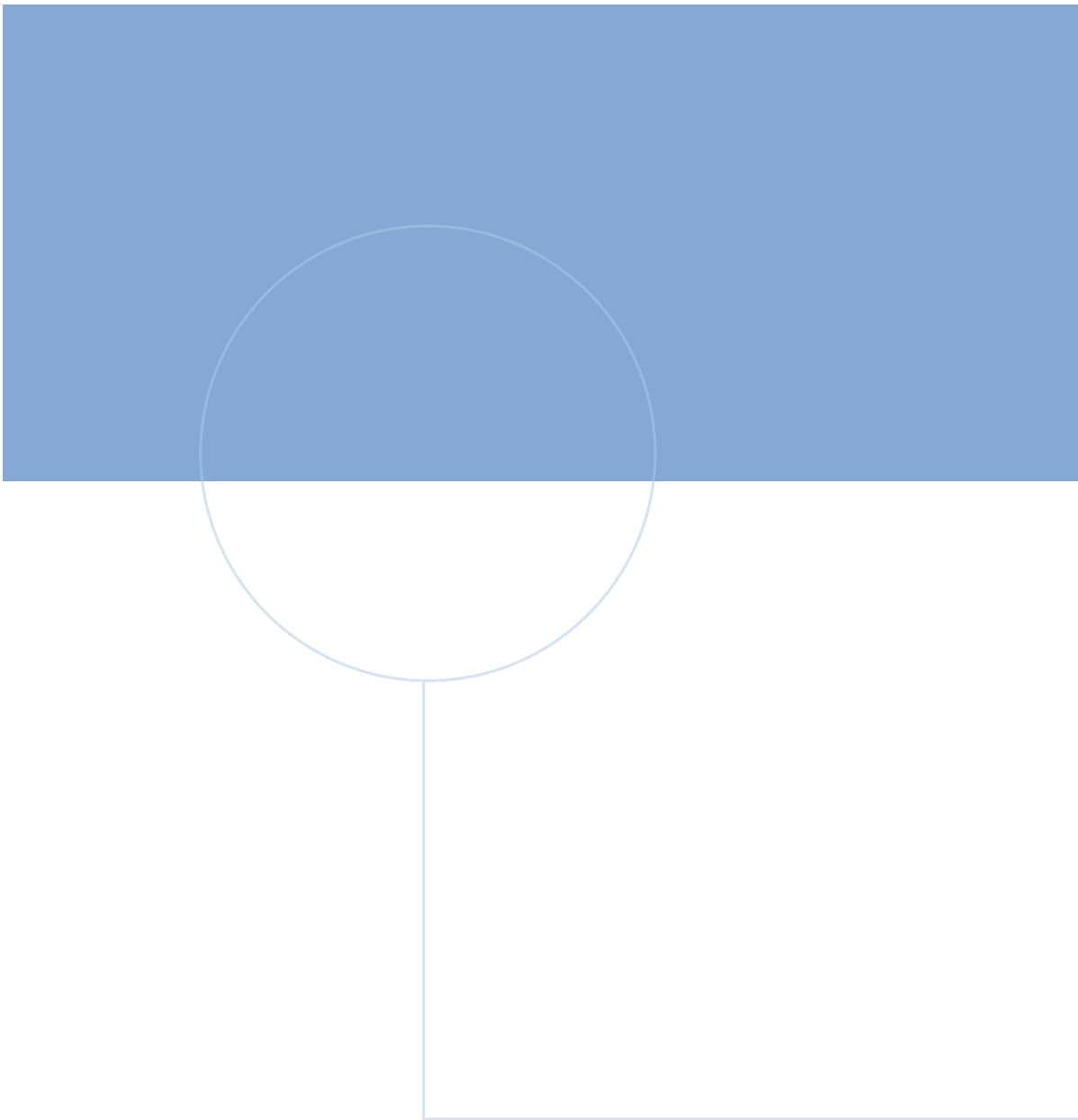
NOTES: All specifications apply to Low-Power Mode and Low-Noise Mode, unless noted otherwise

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	ACCEL_FS=0		±2		*G*	1
	ACCEL_FS=1		±4		*G*	1
	ACCEL_FS=2		±8		*G*	1
	ACCEL_FS=3		±16		*G*	1
ADC Word Length	Output in two's complement format		16		Bits	1
Sensitivity Scale Factor	ACCEL_FS=0		16,384		LSB/*g*	1
	ACCEL_FS=1		8,192		LSB/*g*	1
	ACCEL_FS=2		4,096		LSB/*g*	1
	ACCEL_FS=3		2,048		LSB/*g*	1
Initial Tolerance	Component-level		±0.5		%	2
Sensitivity Change vs. Temperature	-40°C to +85°C ACCEL_FS=0		±0.026		%/ºC	2
Nonlinearity	Best Fit Straight Line		±0.5		%	2, 3
Cross-Axis Sensitivity			±2		%	2, 3
ZERO-G OUTPUT						
Initial Tolerance	Component-level, all axes		±25		*mg*	2
Initial Tolerance	Board-level, all axes		±50		*mg*	2
Zero-G Level Change vs. Temperature	0°C to +85°C		±0.80		*mg*/°C	2
ACCELEROMETER NOISE PERFORMANCE						
Noise Spectral Density	Based on Noise Bandwidth = 10 Hz		230		µ*g*/√Hz	2
LOW PASS FILTER RESPONSE	Programmable Range	5.7		246	Hz	1, 3
ACCELEROMETER STARTUP TIME	From Sleep mode		20		ms	2, 3
	From Cold Start, 1 ms V_{DD} ramp		30		ms	2, 3
OUTPUT DATA RATE	Low-Power Mode	0.27		562.5	Hz	1
	Low-Noise Mode ACCEL_FCHOICE=1; ACCEL_DLPFCFG=x	4.5		1.125k	Hz	
	Low-Noise Mode ACCEL_FCHOICE=0; ACCEL_DLPFCFG=x			4.5k	Hz	

Table 2. Accelerometer Specifications

NOTES:

1. Guaranteed by design.
2. Derived from validation or characterization of parts, not guaranteed in production.
3. Low-noise mode specification.



NTNU
Norwegian University of
Science and Technology

Aviant