Preface

This thesis concludes my masters degree in Engineering Cybernetics at the Norwegian University of Science and Technology.

The quadcopter, ground vehicle and surface vehicle used in this thesis, all equipped with necessary sensors, controllers and communication systems are provided by Norwegian Defence Research Establishment (FFI). FFI have also facilitated with office space and paid for travel expenses. The background and contributions of the project are described in more detail in Section 1.4 Background and Contributions

First of all, I would like to thank my co-supervisor Aleksander Simonsen from FFI, for help and guidance, and never the less for his interest and engagement in my work. Also, thanks to all the other researchers at FFI that have been very accommodating and have gladly attended discussions. Furthermore, thanks to my supervisor Kristin Y. Pettersen for signing up as supervisor and answering my technical questions. Lastly, thanks to Erik Wilthil and Andreas Lindahl Flåten for great discussions and for sharing some of their prime knowledge on state estimators.

Abstract

Multirotor Unmanned Aerial Vehicles (UAVs) high maneuverability and their capability to hover, makes them an extensively used platform in many fields of applications. However, their limitations in flight time challenge the ambition of using multirotor UAVs in fully autonomous operations. By introducing ground or maritime vehicles for deployment and recovery of the UAVs, or even serve as a service platform performing automatic battery replacement, it is possible to perform autonomous operations with multirotor UAVs beyond todays limitations in regards range and duration. To achieve a seamless synergy between the UAVs and the vehicle including a landing pad, requires the UAV to be able to perform autonomous landing on the landing pad wile it is in motion.

This thesis addresses autonomous landing of a multirotor UAV on a vehicle in motion by using traditional navigation sensors in combination with a camera based measurement system. The camera based measurements and the traditional navigation measurements are processed in a Kalman filter developed in this assignment which performs sensor fusion, estimates navigation states as well as calculating the sensor biases. Moreover, two different guidance methods are compared, and a state machine generating flight paths and adjusting controller gains are developed.

The camera based measurement system, the state estimator and the controller are all implemented on the UAV and physical tests have been conducted in real time. Results from the test show that the UAV is, in a robust manner, able to locate, track and precisely land on a static landing pad. Unfortunately, there was no time to conduct

final tests on landing pad in motion. However, results from simulations and the state estimator indicates that the system is able to carry out autonomous landing on a landing pads in motion.

Sammendrag

Ubemannede multirotor luftfartøys (UML) gode evne til både å manøvrere, og til å kunne stå stille i luften, gjør dem til svært populære plattformer med mange ulike bruksområder. Deres ulempe er begrensende flytid, hvilket gjør UML uegnet til mange fullautonome operasjoner. Ved å introdusere andre farkoster for distribusjon og innhenting, automatisk batteribytte og vedlikehold av UML, kan bruken av multirotor luftfartøy i autonome operasjoner nå langt utover dagens bruksområder. For å kunne oppnå en slik samhandling mellom UML og farkosten det skal samarbeide med, må imidlertid multirotor luftfarøtyet kunne lande autonomt på farkosten mens farkosten er i bevegelse.

Denne avhandlingen omhandler temaet autonom landing med UML på en farkost i bevegelse. For å kunne oppnå dette har det blitt brukt tradisjonelle navigasjonssensorer i kombinasjon med kamerabasert målesystem. Avlesninger fra det kamerabaserte målesystemet og navigasjonssensorene er prosessert i et Kalmanfilter utarbeidet i denne oppgaven. Kalmanfilteret utfører sensorfusjonering, og returnerer estimater av navigasjonsvariabler og sensorskjevheter. Videre i oppgaven har det blitt utarbeidet en tilstandsmaskin som genererer flybaner og justerer kontrollerparametere, i tillegg til at to navigasjonsmetoder er definert og sammenlignet.

Det kamerabaserte navigasjonssystemet, tilstandsestimatoren, en av navigasjonsmetodene og tilstandsmaskinen er implementert på et UML som softwaremoduler. Resultater fra et titalls fysiske tester der softwaremodulene kjører i sanntid på luftfartøyet, viser med høy robusthet og etterprøvbarhet at UML kan lokalisere og lande

presist på en statisk landingsplattform. Tiden strakk dessverre ikke til for å få gjennomført de gjenstående testene på autonom landing på farkost i bevegelse. Imidlertid gir resultater fra simuleringer, og testresultater fra tilstandsestimatoren gjennomført på farkost i bevegelse, en sterk indikasjon på at systemet skal kunne gjennomføre autonom landing på farkost i bevegelse.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The multirotor UAV is a popular and much used platform getting more and more introduced into our society by hobbyists, researchers, photographers, the coastguard, farmers, the military and many others. Norwegian Defence Research Establishment (FFI) are using UAVs in many of their fields of research such as, surveillance, Search and Rescue (SAR), Electronic Warfare (EW), autonomy, swarms, communication and environmental monitoring.

Due to the multirotor UAVs high maneuverability, though limited flight time, other autonomous vehicles such as Unmanned Surface Vehicles (USVs), Unmanned Ground Vehicles (UGVs) or even fixed wing UAVs can be used as a platform to transport and serve multirotor UAVs. For autonomous cooperation between multirotor UAVs and other vehicles to be realized, an autonomous system for precise landing on landing pads in motion needs to be established.

1.2 Literature Review

The idea of deploying and recovering multirotor UAVs from a vehicles is not new. Some examples are the autonomous drone service integrated in the Mercedes-Benz Vision Van (AG (2017)) and the ship-to-shore drone delivery system from Field Innovation Team (Gibbs and agencies (2016)).

In the last decade, multiple studies have been conducted in the area of autonomous landing of multirotor UAVs on vehicles in motion. In the paper of Borowczyk et al. (2016) they have been successful in autonomously landing a commercial UAV on the roof of a car at velocities up to $14m/s$. In their application, a smart-phone and a visual fiducial marker were strapped to the roof of the car to provide position and velocity measurements of the landing pad. Araar et al. (2017) did some interesting research on autonomous landing on moving platforms using vision based navigation. The visual fiducial markers used for the navigation were designed to have multiple markers at different sizes, aimed for extensive range of detection. The two filters Extended Kalman and Extended K_∞ where compared for performing the sensor fusion of the visual measurements and the Inertial Measurement Unit (IMU) data, where the Extended Kalman Filter (EKF) by far resulted in the best accuracy. There is also worth mentioning that the system where only tested for velocities up to $1.8m/s$.

To achieve robust and accurate autonomous on a landing pad at speed, accurate navigation methods needs to be established. This implies the need of a good state estimator that estimates the position and velocities of the UAV and landing pad. Throughout time there have been developed many different methods for guidance and navigation. The sun and stars, compass, inertial sensors, landmarks, radar, radio triangulation and Global Navigation Satellite Systems (GNSS) are among the many used methods. Lately, lightweight and low cost inertial sensors, pressure sensors and GNSS receivers have been introduced. According to Beard and McLain (2012) these kind of lightweight and low cost sensors and receivers have made a huge impact on the development of of small unmanned aircrafts.

Many different navigation methods have been carefully tested and implemented on multirotor UAVs. One method frequently used for outdoor navigation is the combi-

nation of GNSS and IMU (Beard and McLain (2012)). Other methods for navigating in GNSS-denied environments are established, as the usage of Ultra-Wideband (UWB) (Tiemann et al. (2015)), infrared motion capture system (Zou et al. (2016)) or vision based navigation (Huang et al. (2015)). More general, solving navigation equations often involve fusion of several sensors returning asynchronous measurements.

The Extended Kalman filter is shown as a method with high performance to solve the Inertial Navigation System (INS)/GNSS integration (Groves (2013)). There are two main methods for implementing the EKF to solve navigation equations, the direct and indirect method. The direct method estimates all the navigation-states in the filter, while the indirect method only estimates errors (Vik (2009)). By using the direct method, the dynamics of the vehicle can be included in the state equations. An accurate dynamic model of the vehicle implemented in an EKF using the direct method, makes the state estimate more accurate and robust against sensor failures by implementing Dead-reconing. On the other hand, creating an accurate dynamic model can be challenging due to many physical parameters to be defined (Roumeliotis et al. (1999)). Martin and Salaün (2010) defines an accurate model for the rotor drag on a quadcopter by using EKF to estimate physical parameters. In the paper from Tailanian et al. (2014) they describe how to implement a full state direct Kalman filter on a multirotor UAV.

The indirect method, often called the error state method, differs from the direct method by that it estimates only the sensors error states. This implies that the system dynamics is not included in the filter, whats makes the filter flexible and universal. The same error state filter can therefore be used in various applications without development of accurate dynamic models or re engineering the filter (Roumeliotis et al. (1999)). Another argument for using the indirect method, is that even low cost Microelectromechanical Systems (MEMS) IMU's gives an relative high accurate measurement that often overcomes the accuracy of the developed model.

Several architectures, for different level of integration have been developed for GNSS/INS integration. In the book of Vik (2009), the uncoupled, loosely coupled, tight coupled and deeply coupled integration methods are discussed. The tightly and deeply coupled integration methods are the most accurate and robust integration

methods. Even the case where there are to few satellites available to calculate the receivers position, the raw measurements from the GNSS receiver will still provide useful information to the filter. Unlike the loosely and uncoupled integration, where the calculated position and velocity-data form the receiver are used. On the other hand, the loosely and uncoupled systems are lees complex and can therefore more easily be implemented. Another benefit is that the GNSS receiver can be switched without changing the code, while tightly and deeply coupled integration methods are customized to a specific sensor Vik (2009).

1.3 Problem description

For multirotor UAVs to have a useful role as a sensor platform, the operation of tracking and recovering to its landing pad needs to be robust and fully automated. Furthermore, to be able to operate from vehicles at sea, or other vehicles without interrupting the ongoing mission, landing on a platform in motion is essential. In order to solve this problem, the assignment has been structured into the following tasks:

- Investigate the available onboard sensors which are relevant for autonomous landing with small quadcopters.
- Consider fiducial marker detection systems and develop a concept for a system in which the marker is being clearly visible for wide range of relevant altitudes.
- Investigate and develop a state estimator for relative positioning which allows asynchronous measurement updates.
- Investigate different control and guidance strategies.
- Implement the marker-detection, the state estimator and a suitable controller on a real platform for testing.
- Perform experiments in which quadcopter is being ordered to land on either a vehicle, a surface vessel or a landing platform from a position nearby.

1.4 Background and Contributions

In this project, a multi marker method described in 3.1.1 has been developed. The method is using a combination of ordinary fiducial markers to extend the detection range. The custom sensor unit described in 5.2.3.1 has also been made during this work. The enclosure of the custom sensor unit was designed and 3D printed, the components were connected together and the flight controller was configured. FFI supplied software and configured the Single Board Computer (SBC) mounted on the custom sensor unit.

Moreover, the following software modules are developed and implemented in this work using C++. All of the listed software modules are tested and run in real time on the SBC mounted on the UAV

- A fiducial marker detection module described in 5.1.2.1. The module detects fiducial markers using Open Source Computer Vision Library (OpenCV) and calculates the relative pose between the UAV and the landing pad.
- A state estimator module described in 5.1.2.2. The module reads and transforms sensor inputs, performing sensor fusion and estimates navigation states and biases.
- A controller module described in 5.1.2.3. The module runs a Guidance Controller controlling the UAV position. It do also include a state machine, generating position set points and controller gains to the Guidance Controller.

The following hardware were given as background material from FFI

- Quadcopter UAV including camera and a SBC set up with ROS and communication link to the workstation and manual control. Additionally, the SBC were set up with communication to the UAV flight controller, allowing velocity set points and receiving measurements.
- UGV included navigation sensors and communication link to the UAV
- USV included navigation sensors and communication link to the UAV

Moreover, FFI contributed with the following software modules

- Software module for reading out raw image from the camera mounted on the UAV
- Software module for communicating with the landing pad
- Safety module between the set points generated by the controller and the flight controller

Additionally, the following sections are based on the authors previous work conducted in the feasibility study carried out in the fall semester 2017 (Line; 2017)

- 1.2 Literature Review
- 2 Modeling and Notations
- 3.2.1 GNSS
- 3.2.2 Inertial Measurement Unit
- 3.2.3 Barometer

1.5 Outline

This thesis is organized as follows. Chapter 2 concerns notations, coordinate frames and transformations. Moreover, a mathematical model describing the system dynamics of a rigid body quadcopter UAV is derived i the same chapter. Sensor equations and Kalman filter equations are found in chapter 3. Further on, in chapter 4 a controller logic and two Guidance methods are presented. Chapter 5 describes the implementation of software and hardware in addition to the test setup. The results from simulation and physical tests are then presented and discussed in chapter 6, before the conclusion and further work ends the assignment in chapter 7.

Chapter 2

Modeling and Notations

In this chapter, the notations and reference frames used in the assignment will be stated. Methods to transfer between coordinate frames will also be examined. Furthermore, a dynamic model of a quadrotor UAV is developed. A representation of the system is useful for simulations, deriving state estimators and for controller design. In this assignment, the following assumptions have been made: The quadcoper is assumed to be symmetric, in other words, all the motors and propellers are equally sized and the lever from the quadcoper center to the motors have the same lengths. Moreover, the multirotor is assumed to have a rigid body, the mass of the propellers is approximated to zero and that the drag force due to air resistance is negligible.

The quad-rotor UAV has been a popular platform due to its simplicity. By using four variable speed motors with fixed pitch propellers, the UAV gets a simple mechanical structure that makes the UAV fully controllable. A god approximation of the force from the propellers is stated in equation 2.1.

$$f = k\omega^2 \tag{2.1}$$

Where $k > 0$ is a constant depending on the shape of the propeller, gear ratio and density of air. ω is the angular velocity of the motor (Lozano (2013)). This chapter is

based on the feasibility study written by the author (Line; 2017).

2.1 Notation

The notations used in this assignment are mainly based on the notations used in Fossen (2011). A coordinate free vector is written as $\vec{u}$. To write a vector relative to coordinate system {n}, the notation $\boldsymbol{u}^n$ is used. When a point is differentiated, it must be done with respect to a reference frame. The notation used for describing this is the subscript $u_{obj/ref}$. For example, the velocity of a particle in reference frame {a} relative to reference frame {b} given in frame {c} is written as $v^c_{a/b}$. Vectors are written as lowercase letters, while matrices are written as uppercase. The inverse of a matrix or vector is written as $\boldsymbol{A}^{-1}$ and $\boldsymbol{A}^\top$ as the transpose.

2.2 Reference Frames

To be able to derive system equations for a vehicle in motion, several reference frames needs to be established. An overview of the reference frames used in this assignment are summarized in table 2.1. Where the Body frame is fixed to the frame of the UAV, where it is often placed in the the UAV center as illustrated in figure 2.3. The ECI frame is centered in the Earths mass center and is fixed in space, unlike the ECEF frame which is also centered in the Earths mass center but is rotating with the Earth (Fossen (2011)). Both the ECI and ECEF have their z axis pointing along with the Earth's rotation axis (Vik (2009)). The NED frame is in this assignment defined as a fixed frame located at a tangent plane at the Earth's reference ellipsoid close to the UAV. The NED frame's x axis is pointing towards true North, the y axis is pointing towards East and the resulting z frame points downwards normal to the ellipsoid. ENU frame is often used as an alternative to the NED frame. The x, y and z axis of the ENU frame is pointing in the East, North and upwards normal to the ellipsoid respectively. The CG frame is placed in the center of gravity of the vehicle, and is orientated at the same direction as the Body frame (Fossen (2011)).

Name	Description	Symbol
Body	Body-fixed reference	b
CG	Center of gravity	g
ECEF	Earth-centered Earth fixed	f
ECI	Earth-centered inertial	i
NED	North-East-Down	n
ENU	East-North-Up	e
GEO	Geodetic Coordinate System	ge

Table 2.1: Reference frames

2.3 Euler angles

The attitude of the Body frame relative to NED is often given by the Euler angles $\Theta_{nb} = [\phi,\ \theta,\ \psi]^\top$ (Fossen (2011)). The Euler angles geometrical definition is given in figure 2.1, where ϕ, θ and ψ are often referred to as roll, pitch and yaw respectively.

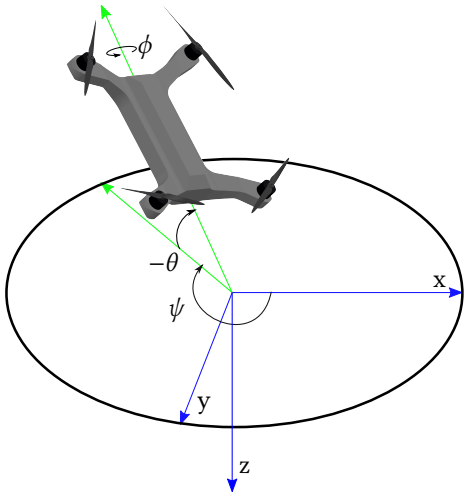


Figure 2.1: Geometrical definition of Euler angles

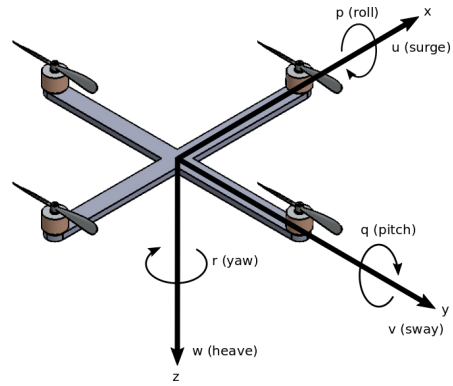


Figure 2.2: Linear velocities u,v,w and the angular velocities p, q, r
The linear velocities u,v,w and the angular velocities p, q, r. All given in Body frame

Transformation of a vector $\in \mathbb{R}^3$ given in the body frame b to the NED frame n is often performed using a rotation matrix $\boldsymbol{R}_b^n(\boldsymbol{\Theta_{nb}})$. As figure 2.1 illustrates, this assignment uses the zyx convention. In other words, the rotational transformation is performed by first rotate an angle ψ about the z axis, followed by the rotation θ about the y axis and finally rotate ϕ about the x axis. The transformation matrix $\boldsymbol{R}_b^n$ in zyx convention as a function of $\boldsymbol{\Theta_{nb}}$ is then given as (Fossen (2011)).

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta_{nb}}) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.2}$$

where c_x and s_x is abbreviations for $\cos(x)$ and $\sin(x)$ respectively. The rotational transformation from NED to body, can be found by taking the inverse of 2.28 (Fossen; 2011).

$$\boldsymbol{R}_n^b(\boldsymbol{\Theta_{nb}}) = \boldsymbol{R}_b^n(\boldsymbol{\Theta_{nb}})^{-1} \tag{2.3}$$

Due to the inconsistent use of the reference frames NED and ENU, a method to transfer between thees two frames needs to be established. The attitude of the ENU frame relative to the NED frame can be given as the Euler angles $\boldsymbol{\Theta}_{ne} = [\pi, 0, \pi/2]^\top$. By using the rotation matrix given in 2.2, the rotational transformation from NED to ENU can be given as

$$\boldsymbol{R}_e^n(\boldsymbol{\Theta_{ne}}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.4}$$

Due to the symmetry of the matrix, we get that $\boldsymbol{R}_e^n(\boldsymbol{\Theta_{ne}}) = \boldsymbol{R}_n^e(\boldsymbol{\Theta_{ne}})$.

The angular velocity transformation from the body-fixed angular velocities to the Euler rate vector can be given as (Fossen (2011))

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^b \tag{2.5}$$

Where $\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})$ is

$$\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{2.6}$$

in this matrix, s_x, c_x and t_x is abbreviations for $\sin(x)$, $\cos(x)$ and $\tan(x)$ respectively.

2.4 Unit Quaternions

Unit quaternions is an alternative to the Euler-angle representation described in 2.3. This four parametric representation of a rotation has the benefit of being able to represent singularity-free three dimensional rotations (Fossen (2011)). A quaternion $\boldsymbol{q}$ is defined as:

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{2.7}$$

where η is the real part and $\boldsymbol{\epsilon}$ is the complex part of the quaternion. In addition, the unit quaternions must satisfy the following constraint (Fossen (2011))

$$\boldsymbol{q}^\top \boldsymbol{q} = 1 \tag{2.8}$$

The angular velocity of a unit quaternion can be derived as

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\eta} \\ \dot{\boldsymbol{\epsilon}} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\boldsymbol{\epsilon}^\top \\ \eta \boldsymbol{I}^{3\times 3} + \boldsymbol{S}(\boldsymbol{\epsilon}) \end{bmatrix} \boldsymbol{\omega}^b_{b/n} \tag{2.9}$$

Furthermore, the rotation matrix for unit quaternions from coordinate frame a to b is

states as (Fossen (2011))

$$\boldsymbol{R}(\boldsymbol{q})_a^b = \begin{bmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2(\epsilon_1\epsilon_2-\epsilon_3\eta) & 2(\epsilon_1\epsilon_3+\epsilon_2\eta) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\eta) & 1-2(\epsilon_1^2+\epsilon_3^2) & 2(\epsilon_2\epsilon_3-\epsilon_1\eta) \\ 1(\epsilon_1\epsilon_3-\epsilon_2\eta) & 2(\epsilon_2\epsilon_3+\epsilon_1\eta) & 1-2(\epsilon_1^2+\epsilon_2^2) \end{bmatrix} \tag{2.10}$$

As in the Euler angles notation in 2.3, there is also an angular velocity transformation for unit quaternions (Fossen (2011))

$$\dot{\boldsymbol{q}} = \boldsymbol{T}_q(\boldsymbol{q})\boldsymbol{\omega}_{b/n}^b \tag{2.11}$$

Where $\boldsymbol{T}_q(\boldsymbol{q})$ is

$$\boldsymbol{T}_q(\boldsymbol{q}) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \tag{2.12}$$

Converting Euler angles to quaternions and the other way around can be done by requiring the that the rotation matrices for both Euler-angles and unit quaternions are equal (Vik (2009)).

$$\boldsymbol{R}(\boldsymbol{q})_a^b = \boldsymbol{R}(\boldsymbol{\Theta})_a^b \tag{2.13}$$

2.5 Geodetic coordinates

One method to represent a global position near the surface of the earth is by using the geodetic coordinate system. Geodetic coordinate systems depends on a reference ellipsoid of the earth. The coordinate system describes the position of a point with the variables longitude, latitude and height respectively denoted as λ, ϕ and h. Where the longitude is the rotational angle between the Prime Median and the point, latitude is the angle between the equator plane and the normal of the reference ellipsoid passing trough the point and height is the distance between the ellipsoid and the point (Cai

et al.; 2011).

All GNSS receivers used in this assignment refers to the same WGS-84 ellipsoid model. Cai et al. (2011) summarize the WGS-84 parameters as

$$R_{Ea} = 6378137.0m \tag{2.14}$$

$$f = 1/298.257223563 \tag{2.15}$$

where R_{Ea} is the semi-major axis and f is the flattering factor. Further on, the semi-minor axis R_{Eb}, first eccentricity e, median radius of curvature M_E and the prime vertical radius of curvature N_E are defined as

$$R_{Eb} = R_{Ea}(1 - f) \tag{2.16}$$

$$e = \frac{\sqrt{R_{Ea}^2 - R_{Eb}^2}}{R_{Ea}} \tag{2.17}$$

$$M_e = \frac{R_{Ea}(1 - e^2)}{(1 - e^2 \sin^2 \phi)^{3/2}} \tag{2.18}$$

$$N_E = \frac{R_{Ea}}{\sqrt{1 - e^2 \sin^2 \phi}} \tag{2.19}$$

A point $\boldsymbol{p}^{ge} = [\lambda, \phi, h]^\top$ in the geodetic coordinate system can be given in the ECEF coordinate system by using the conversion method

$$\boldsymbol{p}^f = \begin{bmatrix} (N_E + h)\cos\phi\cos\lambda \\ (N_E + h)\cos\phi\sin\lambda \\ [N_E(1 - e^2) + h]\sin\phi \end{bmatrix} \tag{2.20}$$

(Cai et al.; 2011)

Furthermore, the point $\boldsymbol{p}$ can be represented relative to a local NED coordinate system. In this assignment, the local NED frame $\boldsymbol{p}_{loc}$ is defined to be whatever the UAV reads from the GNSS sensor when the UAV arms the motors. Equation 2.21 states

the position conversion given in Cai et al. (2011)

$$\boldsymbol{p}^n = \boldsymbol{R}_f^n(\boldsymbol{p}_{loc}^{ge})(\boldsymbol{p}^f - \boldsymbol{p}_{loc}^f) \tag{2.21}$$

where $\boldsymbol{R}_f^n(\boldsymbol{p}_{loc}^{ge})$ is the rotation matrix from ECEF to the loc NED given as

$$\boldsymbol{R}_f^n(\boldsymbol{p}_{loc}^{ge}) = \begin{bmatrix} -s_{\phi_{loc}} c_{\lambda_{loc}} & -s_{\phi_{loc}} s_{\lambda_{loc}} & c_{\phi_{loc}} \\ -s_{\lambda_{loc}} & c_{\lambda_{loc}} & 0 \\ -c_{\phi_{loc}} c_{\lambda_{loc}} & -c_{\phi_{loc}} s_{\lambda_{loc}} & -s_{\phi_{loc}} \end{bmatrix} \tag{2.22}$$

where c_x and s_x is abbreviations for $\cos(x)$ and $\sin(x)$ respectively.

2.6 Skew Symmetric Matrix

A matrix is said to be skew symmetric if and only if $S = S^\mathsf{T}$ (Spong et al. (2006)). The 3×3 skew symmetric matrix can be stated as:

$$S(\boldsymbol{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

where $\boldsymbol{x} \in \mathbb{R}^3$. This matrix is useful for performing the vector cross product defined by: (Fossen (2011))

$$\boldsymbol{a} \times \boldsymbol{b} := S(\boldsymbol{a})\boldsymbol{b}$$

where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$.

2.7 Dynamic Modeling of a Quad-Rotor

In this section, the dynamic model for a rigid body quadcoper will be derived using the Newton-Euler approach. Euler's equation can for linear- ($\vec{p}_g$) and angular momentum

($\vec{h}_g$) is defined as in equation 2.23 and 2.24 respectively (Fossen (2011)).

$$\frac{^n d}{dt}\vec{p}_g = \vec{f}_g \qquad \vec{p}_g = m\vec{v}_{g/n} \tag{2.23}$$

$$\frac{^n d}{dt}\vec{h}_g = \vec{m}_g \qquad \vec{h}_g = I_g\vec{\omega}_{b/n} \tag{2.24}$$

where $\vec{f}_g$ is the force acting on the vehicle, $\vec{m}_g$ is the moment acting on the same rigid body, m is the total mass, I_g is the inertia about the center of gravity, $\vec{v}_{g/n}$ and $\vec{\omega}_{b/n}$ is the linear and angular velocities respectively.

The equation for linear momentum in 2.23 can be rewritten as

$$\vec{f}_g = m\frac{^n d}{dt}\vec{v}_{g/n} \tag{2.25}$$

, and by expressing the equations in body frame the following is obtained:

$$\vec{f}_g = m(\frac{^b d}{dt}\vec{v}_{g/n} + \vec{\omega_{b/n}} \times \vec{v_{b/n}}) \tag{2.26}$$

$$\boldsymbol{f}_g^b = m(\dot{\boldsymbol{v}}_{g/n}^b + \boldsymbol{S}(\boldsymbol{\omega}_{b/n}^b)\boldsymbol{v}_{g/n}^b) \tag{2.27}$$

Equation 2.24 can be reformulated such that the rotational motion about CG can be expressed in Body frame as

$$\boldsymbol{m}_g^b = \boldsymbol{I}_g(\dot{\boldsymbol{\omega}}_{b/n}^b - \boldsymbol{S}(\boldsymbol{\omega}_{\boldsymbol{b/n}}^{\boldsymbol{b}})\boldsymbol{\omega}_{\boldsymbol{b/n}}^{\boldsymbol{b}}) \tag{2.28}$$

where $\boldsymbol{S}$ is the skew symmetric matrix defined in 2.6.

Resulting Newton Euler equations from 2.27 and 2.28 can be written as (Fossen (2011))

$$\begin{bmatrix} m\boldsymbol{I}^{3\times3} & \boldsymbol{0}^{3\times3} \\ \boldsymbol{0}^{3\times3} & \boldsymbol{I}_g \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{v}}_{g/n}^b \\ \dot{\boldsymbol{\omega}}_{g/n}^b \end{bmatrix} + \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\omega}_b^{b/n}) & \boldsymbol{0}^{3\times3} \\ \boldsymbol{0}^{3\times3} & -\boldsymbol{S}(\boldsymbol{\omega}_b^{b/n}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{g/n}^b \\ \boldsymbol{\omega}_{g/n}^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_g^b \\ \boldsymbol{m}_g^b \end{bmatrix} \tag{2.29}$$

where $\boldsymbol{I}^{3\times3}$ is the 3×3 identity matrix and $\boldsymbol{0}^{3\times3}$ is a 3×3 matrix containing only zeros. The matrix involving the system inertia constants can be states as $\boldsymbol{M}_{RB}^{CG}$ and the matrix containing the Coriolis-Centripetal parameters as $\boldsymbol{C}_{RB}^{CG}$. Further on, the equation 2.29 can be written in the compact matrix form (Fossen (2011))

$$\boldsymbol{M}_{RB}^{CG}\begin{bmatrix}\dot{\boldsymbol{v}}_{g/n}^{b}\\ \dot{\boldsymbol{\omega}}_{g/n}^{b}\end{bmatrix}+\boldsymbol{C}_{RB}^{CG}\begin{bmatrix}\boldsymbol{v}_{g/n}^{b}\\ \boldsymbol{\omega}_{g/n}^{b}\end{bmatrix}=\begin{bmatrix}\boldsymbol{f}_{g}^{b}\\ \boldsymbol{m}_{g}^{b}\end{bmatrix} \tag{2.30}$$

where $\boldsymbol{M}_{RB}^{CG}$ is unique, while there is possible to find a large number of representations for $\boldsymbol{C}_{RB}^{CG}$ (Fossen (2011)).

System equations given in 2.30 are expressed for system motions relative to center of gravity. A more suitable set of equations would be to have the system motions described about a chosen body frame. A vector $\boldsymbol{r}_{g}^{b}$ describes the translation from center of gravity to the body frame. By using coordinate transformation, we can according to Fossen (2011) rewrite 2.30 as

$$\begin{bmatrix}m\boldsymbol{I}^{3\times3} & -mS(\boldsymbol{r}_{g}^{b})\\ mS(\boldsymbol{r}_{g}^{b}) & \boldsymbol{I}_{b}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{v}}_{b/n}^{b}\\ \dot{\boldsymbol{\omega}}_{b/n}^{b}\end{bmatrix}+\begin{bmatrix}mS(\boldsymbol{\omega}_{b/n}^{b}) & -mS(\boldsymbol{\omega}_{b/n}^{b})S(\boldsymbol{r}_{g}^{b})\\ -mS(\boldsymbol{r}_{g}^{b})S(\boldsymbol{\omega}_{b/n}^{b}) & -S(\boldsymbol{I}_{b}\boldsymbol{\omega}_{b/n}^{b})\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_{b/n}^{b}\\ \boldsymbol{\omega}_{b/n}^{b}\end{bmatrix}=\begin{bmatrix}\boldsymbol{f}_{b}^{b}\\ \boldsymbol{m}_{b}^{b}\end{bmatrix} \tag{2.31}$$

which again, can be written more compact(Fossen (2011)).

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}}+\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}=\boldsymbol{\tau}_{RB} \tag{2.32}$$

Where $\boldsymbol{M}_{RB}$ is often referred to as the Rigid-Body System Inertia Matrix while $\boldsymbol{C}_{RB}$ is called Coriolis-Centripetal Matrix. $\boldsymbol{\nu}$ is the state vector and $\boldsymbol{\tau}_{RB}$ is a generalized vector

of external forces and moments, both written out in 2.33.

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \qquad \boldsymbol{\tau}_{RB} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} \tag{2.33}$$

There are two main linear forces acting on the quadcopter. The gravitational force and the force given by the propellers. For the model developed in this assignment, we approximate the gravitational force to be parallel to the NED frame's z axis. The force due to gravitation is then given as $\boldsymbol{f}^n_{g,g} = mg[0\,0\,1]^T$. By transferring this force to body frame, the gravitational force equation is written $\boldsymbol{f}^b_{g,g} = \boldsymbol{R}^b_n mg[0\,0\,1]^T$, where $\boldsymbol{R}^b_n$ is the rotation matrix described in section 2.3. The force from the motor-driven propellers is given in equation 2.1. For a quadcopter design, the motors are fixed to the frame and the propellers force are parallel to the body frame z axis $\sum_{i=1}^{4} k\omega_i^2[0\,0\,1]^T$. The sum of forces acting on the rigid UAV frame can then be expressed as

$$\boldsymbol{f}^b_g = \boldsymbol{R}^b_n mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - k \sum_{i=1}^{4} \omega_i^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.34}$$

where ω_i is the rotational speed of motor i. The directions of the forces are given by the frames of reference given in figure 2.3.

Angular momentum acting on the UAV frame is generated from the difference in force generated from motors across the x and y axis of the body frame (Lozano (2013)). The difference in force from motor one and three gives an angular momentum around the y axis, while difference in force from motor two and four generates angular momentum around the x axis. The relation between angular- and linear moment is $\boldsymbol{m} = \boldsymbol{r} \times \boldsymbol{f}$, where $\boldsymbol{r}$ is the vector from center of rotation to the linear force, and $\boldsymbol{f}$ is the linear moment (Egeland and Gravdahl (2002)). There is also generated angular

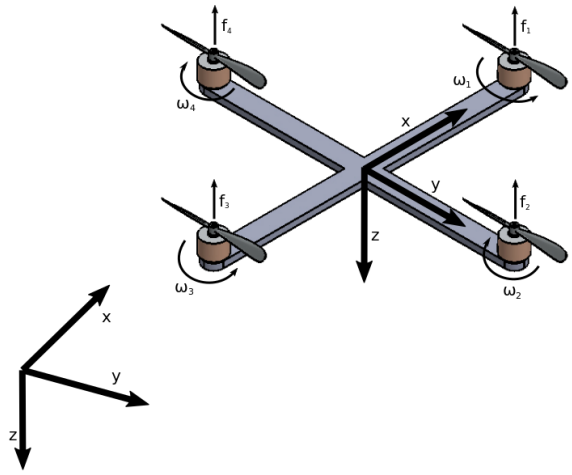


Figure 2.3: Body frame and NED frame
Illustrates that the position of body-frame is in the center of the UAV. The difference in force from motors one and three generates torque about the body frame y axis, while motor 2 and 4 affects the torque about the x axis.

momentum around the z axis. This momentum are generated due to the drag force from the propellers (Nicolai and Carichner (2001)), and can be formulated as $d\boldsymbol{\omega}_i^2$, where d is the drag factor and ω_i is the angular velocity for propeller i. By using the right-hand rule to determine the directions at the forces and momentums, we can summarize the input torques as

$$\boldsymbol{m}_b^b = \begin{bmatrix} lk(\boldsymbol{\omega}_4^2 - \boldsymbol{\omega}_2^2) \\ lk(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) \\ d(-\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 - \boldsymbol{\omega}_3 + \boldsymbol{\omega}_4) \end{bmatrix} \tag{2.35}$$

Where l is the distance from the body frame to the propeller.

By using the angular velocity transformation and the rotation matrix differential equations defined in 2.3, the kinematic equations for the position and attitude can be summarized on the form $\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}$ as (Fossen (2011))

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{n/b}^n \\ \dot{\boldsymbol{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_n b) & \boldsymbol{0}^{3\times3} \\ \boldsymbol{0}^{3\times3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_n b) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{b/n}^b \\ \boldsymbol{\omega}_{b/n}^b \end{bmatrix} \tag{2.36}$$

The Euler angle approach is used for this equation set. The kinematic equation can also be defined for the singularity free unit quaternions defined in 2.4 (Fossen2011)

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{n/b}^n \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{q}) & \boldsymbol{0}^{3\times3} \\ \boldsymbol{0}^{4\times3} & \boldsymbol{T}_q(\boldsymbol{q}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{b/n}^b \\ \boldsymbol{\omega}_{b/n}^b \end{bmatrix} \tag{2.37}$$

The 12 state differential equation set can be summarized as

$$\begin{bmatrix} \dot{\boldsymbol{\nu}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{RB}^{-1}(\boldsymbol{\tau}_{RB} - \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}) \\ \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \end{bmatrix} \tag{2.38}$$

where $\boldsymbol{J}$ is either the $\boldsymbol{J}_\Theta$ from equation 2.36 using the Euler method or $\boldsymbol{J}_q$ from equation 2.37 using quaternions.

Chapter 3

Navigation

3.1 Fiducial Markers and Camera

Autonomous landing on a relative small landing pad relies on an accurate relative measurement between the UAV and the landing pad. Several methods were considered, such as radio ranging, infrared sensing, LiDAR and Reflective markers. The fiducial marker detection method where selected due to its simplicity, robustness and low cost. In robot navigation, multiple fiducial marker detection methods for camera pose estimation have been developed, such as the ArUco library, Intersense, ARTag, CyberCode, ReacTIVision, BinARyID and more (Garrido-Jurado et al.; 2014).

In the paper of Garrido-Jurado, noz Salinas, Madrid-Cuevas and Marín-Jiménez (2014), they presents the ArUco library for camera pose estimation relative to fiducial markers. The library includes a general method to generate markers with the lowest fault rate possible. ArUco tag detection functions are also included in the ArUco library, where a local adaptive thresholding approach is used rather than the Canny edge detector to limit the computational load in the image segmentation phase. Further on, contour extraction and filtering is performed to filter out all contours that are not 4-vertex polygons. All the resulting polygons are then reshaped as rectangular images and compared with the generated marker library by dividing the rectangle in to a

grid and translate each element to 1 or 0 depending on the average amount of white or black in the element. The generation and detection of ArUco fiducial markers are included in the OpenCV project Itseez (2018).

3.1.1 Multi marker system

One of the main requirements of the relative measurement, is to have an accurate measurement both at altitudes up to 25m and at close range down to 30cm. Due to limitations in imaging sensors and fixed optics, one single marker will not be able to give an accurate measurement within the necessary altitude rage. Two alternatives to the single marker were therefor developed in this work, a Recursive ArUco Marker (RAM) method and a Pixel Replacement in ArUco Marker (PRiAM) method.

The RAM is built up with several unique ArUco markers in decreasing sizes placed in a circular shaped pattern. An example of the RAM is given in figure 3.1. In the PRiAM method, an ArUco fiducial marker with a black center pixel is selected as the larger (outer) tag. The center pixel is then replaced by an AruCo tag with the same dimensions as the pixel. Figure 3.2 gives an example of this method using two unique ArUco markers. In this PRiAM method, the tag used as the center pixel in the outer marker must contain a dominance of black pixels to be treated as a black pixel. The main benefits of using the PRiAM rather than the RAM method, is that the the center of the tag is equal at all heights and that the footprint of the tag is not affected compared to a single marker. On the other hand, the pixel replacement method may be more sensitive to noise, due to the mixture of white and black pixels in the inner tag. It is also restricted to fiducial markers that are rectangular and have a center pixel, while the recursive tag method can be used by any fiducial marker systems.

3.2 Sensor input

Both the UAV and the landing pad consists of navigation sensors such as GNSS, magnetometers, barometer, linear accelerometers and rate gyros. Measurements from these sensors are then used in their already implemented pose and velocity state

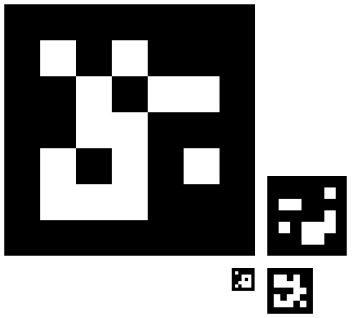


Figure 3.1: Example of RAM

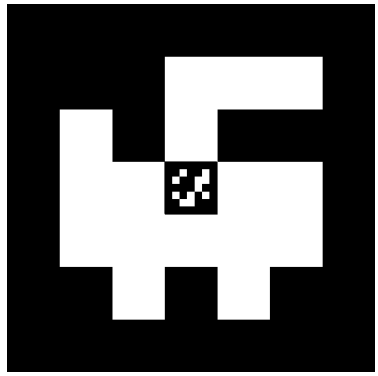


Figure 3.2: Example of PRiAM

estimators. The GNSS/INS integration will therefore not be present in this assignment. Some solutions to the GNSS/INS problem can be studied in future details in Vik (2009). The resulting pose and velocity estimates and their corresponding error covariance are treated as new measurements in the state estimator developed in section 3.3. The fiducial pose estimate given from the ArUco tag detection in section 3.1 is also treated as a measurement and fed in to the state estimator. Sensor equations are developed in this section to get an understanding of the biases and noise in these measurements.

3.2.1 GNSS

In this assignment, where the UAV is flying relatively close to the defined NED frame, we can state the GNSS position and velocity equations as:

$$\boldsymbol{y}_{pos,GNSS} = \begin{bmatrix} \lambda \\ \phi \\ h \end{bmatrix} + \boldsymbol{w}_{pos} \tag{3.1}$$

$$\boldsymbol{y}_{vel,GNSS} = \boldsymbol{v}^b_{b/n} + \boldsymbol{w}_{vel} \tag{3.2}$$

Where λ and ϕ is longitude and latitude respectively, often stated as $\Theta_{en} \in \mathcal{S}^2$. $h \in \mathbb{R}$ is the altitude above the WGS-84 ellipsoid. $\boldsymbol{w}_{pos} \in \mathbb{R}^3$ is the zero-mean Gaussian white

noise. $\boldsymbol{v}^n_{b/n}$ and $\boldsymbol{w}_{vel}$ is the linear velocity of the UAV relative to the NED frame and its associated zero-mean Gaussian white noise. Section 2.5 describes a method for transforming the geodetic coordinates to a position vector relative to NED. A GNSS receiver gives position estimates at typically 1-5Hz.

3.2.2 Inertial Measurement Unit

By combining the benefit of global positioning from the GNSS system and the high measurements rate from the accelerometers, magnetometers and rate gyros, a fast and accurate position estimate can be achieved. This sensor combination is commonly used, and for that reason the industry has developed a unit including three accelerometers, rate gyros and magnetometers, named IMU (Beard and McLain (2012)). Further on in this section, the IMU is assumed to be placed in the center of the body frame with the sensors pointing parallel to the body axis.

Accelerometers

There are three major accelerations affecting the IMU during a flight. Its the accelerations from the body due to changes in the linear velocity, centripetal acceleration due to rotational velocity and the gravitational acceleration. The sensor equation can be derived as (Vik (2009)):

$$\boldsymbol{y}_{accel} = \dot{\boldsymbol{v}}^b_{b/n} + \boldsymbol{\omega}^b_{b/n} \times \boldsymbol{v}^b_{b/n} - \boldsymbol{R}^b_n \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \boldsymbol{\beta}_{accel} + \boldsymbol{w}_{accel} \tag{3.3}$$

where $\boldsymbol{\beta}_{accel}$ is the bias from sensor drifting and other none estimated terms. The $\boldsymbol{w}_{accel}$ is the zero-mean Gaussian white noise.

Rate gyro

Rate gyros measures the angular velocity about the sensors sensitive axis with respect to the Earth-centered inertial frame. Du to high measurement biases in MEMS gyros

(10-3600deg/hr), the measurement of Earth rotation can therefore be neglected. The resulting rate gyro sensor equation can be derived as

$$\boldsymbol{y}_{gyro} = \boldsymbol{\omega}^b_{b/n} + \boldsymbol{\beta}_{gyro} + \boldsymbol{w}_{gyro} \tag{3.4}$$

$\boldsymbol{\omega}^b_{b/n}$ $\boldsymbol{\beta}_{gyro}$ is the bias from sensor drifting and other none estimated terms while $\boldsymbol{w}_{gyro}$ is the zero-mean Gaussian white noise.

Magnetometer

A magnetometer measures the strength of a magnetic field in its sensing axis. The direction of the field can be measured by using three sensors placed orthogonal to each other. If the vehicles horizontal plane is leveled with the North East plane, in other words $\phi \approx \theta \approx 0$, the magnet heading angle can be measured as (Fossen (2011), Beard and McLain (2012)):

$$\psi_m = -\operatorname{atan2}(m_y, m_x) \tag{3.5}$$

where m_x and m_y is the magnetic reading along the x and y axis respectively. The atan2 function is the two-argument arctangent function that uses the sign of m_y and m_x to select the appropriate quadrant (Spong et al. (2006)).

To be able to derive the heading equation for applications that do not have approximated zero roll and pitch angles, the angles need to be measured or estimated. By achieving these angles, the equation can be stated as (Fossen (2011)):

$$\begin{aligned} h_x &= m_x \cos(\theta) + m_y \sin(\theta)\sin(\phi) + m_z \cos(\phi)\sin(\theta) \\ h_y &= m_y \cos(\phi) - m_z \sin(\phi) \\ \psi_m &= \left\{ \begin{array}{lll} -\operatorname{atan2}(h_y, h_x) & \text{if} & h_x \neq 0 \\ \pi/2 & \text{if} & h_x = 0, h_y < 0 \\ 3\pi/2 & \text{if} & h_x = 0, h_y > 0 \end{array} \right\} \end{aligned} \tag{3.6}$$

where h_x and h_y are the horizontal components from $\boldsymbol{m}$ transformed to b frame. These equations are often calculated in the *strapdown equations*, where ψ_m is the output from these equations and can be treated as a measurement y_{mag}. Due to the difference in

magnetic north and true north, in other words $\psi = \delta + \psi_m$, the measurement will include a bias term (Beard and McLain (2012)). The measurement equation is therefore stated as:

$$y_{mag} = \psi^b_{b/n} + \beta_{mag} + w_{mag} \tag{3.7}$$

where β_{mag} is the bias term due to Earth's magnetic field declination and other magnetic fields and w_{mag} is the zero-mean Gaussian white noise. The disadvantage of the magnetometer is that it is affected by other time varying magnetic fields, such as power cables and motors.

3.2.3 Barometer

An important navigation-parameter in aerial vehicles is the height above ground. By using an absolute pressure sensor measuring the atmospheric pressure relative to a fixed reference pressure, an relative altitude estimate can be calculated using a basic equation of hydrostatics for a static fluid (Beard and McLain (2012))

$$\delta p = \rho g \delta z \tag{3.8}$$

where δp is the change in pressure due to change in height δz, for a static fluid and constant density ρ. The air in the atmosphere is compressible and have a density varying on altitude and weather-conditions. On the other hand, the variation in density at altitudes lower than 1000m are so small that they can be neglected. The variations in weather-conditions can also be neglected for quadrotor applications due to the short fly-range. The change in pressure δp, can be defined as $p_g - p_m$, where p_g and p_m is the pressure at sea level and sensor altitude respectively. Hence, the sensor equation can be given as in 3.9.

$$y_{pres} = \rho g h + \beta_{pres} + w_{pres} \tag{3.9}$$

Where β_{pres} is the bias term, w_{pres} is the zero-mean Gaussian white noise and ρ is the density of air.

3.2.4 UAV and Landing Pad Sensors

As mentioned in the introduction of this chapter, the sensor outputs from the GNNS, IMU and barometer are processed in an already implemented navigation equations and state estimator in the UAV and the landing pad.

Measurement Type	Measurement UAV	Measurement LP
Global position	$\boldsymbol{p}_u^{ge}$	$\boldsymbol{p}_l^{ge}$
Global position covariance	$\boldsymbol{r}_{p_u^e}$	$\boldsymbol{r}_{p_l^e}$
Orientation	$\boldsymbol{q}_{eu}$	$\boldsymbol{q}_{el}$
Linear velocity	$\boldsymbol{v}_{u/e}^e$	$\boldsymbol{v}_{l/e}^e$
Angular velocity	$\boldsymbol{\omega}_{u/e}^e$	$\boldsymbol{\omega}_{l/e}^e$

Table 3.1: Measurement types received from the UAV and landing pad

Table 3.1 gives an overview of the measurements read form the UAV and the landing pad, where $\boldsymbol{p}_u^g$ and $\boldsymbol{p}_l^g$ are the position given in geodetic coordinates and $\boldsymbol{r}_{p_u^e}$ and $\boldsymbol{r}_{p_l^e}$ are the associated covariance given in ENU. $\boldsymbol{q}_{eu}$ and $\boldsymbol{q}_{el}$ are the UAV and the landing pad orientation relative to ENU given in quaternions. $\boldsymbol{v}_{u/e}^e$ and $\boldsymbol{v}_{l/e}^e$ are the linear velocities of the UAV and landing pad given in and relative to ENU, and $\boldsymbol{\omega}_{u/e}^e$ and $\boldsymbol{\omega}_{l/e}^e$ are the angular velocities of the UAV and landing pad given in and relative to ENU. To be able to use the measurements in the state estimator developed in chapter 3, all the measurements needs to be transformed according to the measurement vector given in the same chapter. The position measurements are the results from the internal navigation filter in the UAV and the landing pad. The position estimates $\boldsymbol{p}_u^{ge}$ and $\boldsymbol{p}_l^{ge}$ does therefore include the benefit of global position from the GNSS sensor and high frequency relative position from the IMU and barometer. Section 2.5 presents a method to represent the geodetic coordinates relative to the *n* frame. Equation 3.10 and 3.11 summarizes the method used to transfer the position to NED.

$$\boldsymbol{p}_{u/n}^n = \boldsymbol{R}_f^n(\boldsymbol{p}_{loc}^{ge})(\boldsymbol{p}_u^f - \boldsymbol{p}_{loc}^f) + \boldsymbol{\beta}_{u/n}^n + \boldsymbol{w}_{p_u} \tag{3.10}$$

$$\boldsymbol{p}_{l/n}^n = \boldsymbol{R}_f^n(\boldsymbol{p}_{loc}^{ge})(\boldsymbol{p}_l^f - \boldsymbol{p}_{loc}^f) + \boldsymbol{\beta}_{l/n}^n + \boldsymbol{w}_{p_l} \tag{3.11}$$

where $\boldsymbol{p}_{loc}$ is the local NED origin and $\boldsymbol{R}_f^n$ is the rotation matrix from ECEF to NED given in equation 2.21. $\boldsymbol{p}_u^f$ and $\boldsymbol{p}_l^f$ are the UAV and landing pad position given in ECEF. The transformation from geodetic coordinates to ECEF are given in equation 2.20. $\boldsymbol{\beta}$ and $\boldsymbol{w}$ is the measurement bias and zero-mean Gaussian white noise respectively.

The position covariance matrices $\boldsymbol{r}_{p_u^e} \in \mathbb{R}^{3x3}$ and $\boldsymbol{r}_{p_l^e} \in \mathbb{R}^{3x3}$ does only contain elements on its diagonal and represents the position error covariance, and are given in the ENU frame. The covariance matrix can be represented in the NED frame by switching the element $\boldsymbol{r}_{p^e}(1,1)$ with the element $\boldsymbol{r}_{p^e}(2,2)$.

$$\boldsymbol{r}_{p^n} = \begin{bmatrix} \boldsymbol{r}_{p^e(2,2)} & 0 & 0 \\ 0 & \boldsymbol{r}_{p^e(1,1)} & 0 \\ 0 & 0 & \boldsymbol{r}_{p^e(3,3)} \end{bmatrix} \tag{3.12}$$

The rotation between the ENU frame and body frame $\boldsymbol{q}_{eu} = [\boldsymbol{\eta}_{eu}, \boldsymbol{\epsilon}_{1,eu}, \boldsymbol{\epsilon}_{2,eu}, \boldsymbol{\epsilon}_{3,eu}]^\top$ and $\boldsymbol{q}_{el} = [\boldsymbol{\eta}_{el}, \boldsymbol{\epsilon}_{1,el}, \boldsymbol{\epsilon}_{2,el}, \boldsymbol{\epsilon}_{3,el}]^\top$ can be transformed to represent the rotation between the NED frame and body frame by using the geometric relations given in 3.13.

$$\boldsymbol{q}_{nu} = \begin{bmatrix} \boldsymbol{\eta}_{eu} \\ \boldsymbol{\epsilon}_{2,eu} \\ \boldsymbol{\epsilon}_{1,eu} \\ -\boldsymbol{\epsilon}_{3,eu} \end{bmatrix} + \boldsymbol{w}_q \qquad \boldsymbol{q}_{nl} = \begin{bmatrix} \boldsymbol{\eta}_{el} \\ \boldsymbol{\epsilon}_{2,el} \\ \boldsymbol{\epsilon}_{1,el} \\ -\boldsymbol{\epsilon}_{3,el} \end{bmatrix} + \boldsymbol{w}_q \tag{3.13}$$

where $\boldsymbol{w}_q$ is the zero-mean Gaussian white noise

The linear and angular velocities can be given related to NED by using the rotation matrix $\boldsymbol{R}_e^n(\boldsymbol{\Theta}_{ne})$ given in 2.4

$$\boldsymbol{v}_{u/n}^n = \boldsymbol{R}_e^n(\boldsymbol{\Theta}_{ne})\boldsymbol{v}_{u/e}^e + \boldsymbol{w}_v \tag{3.14}$$

$$\boldsymbol{\omega}_{u/n}^n = \boldsymbol{R}_e^n(\boldsymbol{\Theta}_{ne})\boldsymbol{\omega}_{u/e}^e + \boldsymbol{w}_\omega \tag{3.15}$$

$$\boldsymbol{v}_{l/n}^n = \boldsymbol{R}_e^n(\boldsymbol{\Theta}_{ne})\boldsymbol{v}_{l/e}^e + \boldsymbol{w}_v \tag{3.16}$$

$$\boldsymbol{\omega}_{l/n}^n = \boldsymbol{R}_e^n(\boldsymbol{\Theta}_{ne})\boldsymbol{\omega}_{l/e}^e + \boldsymbol{w}_\omega \tag{3.17}$$

where $\boldsymbol{w}_v$ and $\boldsymbol{w}_\omega$ are the respective zero-mean Gaussian white noise for the linear and angular velocity.

3.3 State estimators

As illustrated in figure 3.3, the measurements send to the state estimator originates from the landing pad, the UAV and the camera mounted on the UAV. Both the Landing pad and the UAV measurements are relative to and given in the same NED reference system. While the relative measurement from the camera, returns the pose of the landing pad relative to the UAV frame given in NED.

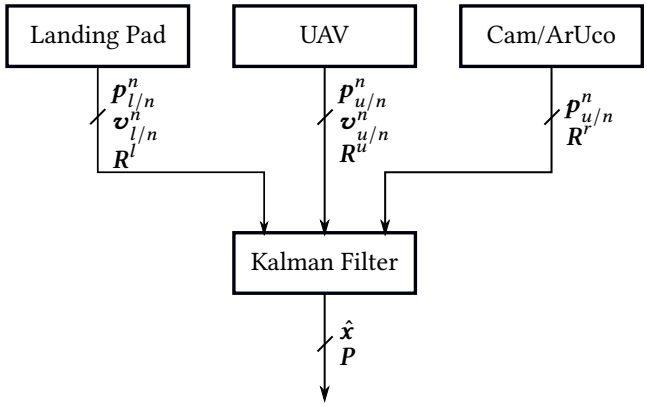


Figure 3.3: Measurements used in the Kalman filter
Measurements used in the Kalman filter to estimate the states $\hat{\boldsymbol{x}}$ and the covariance $\boldsymbol{P}$ of the estimate

The different sensor inputs have different update-rates and the measurements read from the camera arrive with a variable time step. A method to implement a Kalman filter that handles asynchronous measurements must for that reason be developed. First of all, the filter does not know the time until the next filter update. Consequently the a priori covariance and state estimate needs to be calculated at the beginning of

each step k.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{\Phi}_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{\Delta}_k \boldsymbol{u}_{k-1} \tag{3.18}$$

$$\boldsymbol{P}_k^- = \boldsymbol{\Phi}_k \boldsymbol{P}_{k-1} \boldsymbol{\Phi}_k^\top \tag{3.19}$$

where $\hat{\boldsymbol{x}}_k^-$ and $\boldsymbol{P}_k^-$ is the a priori state estimate and covariance respectively. $\hat{\boldsymbol{x}}_{k-1}$ and $\boldsymbol{u}_{k-1}$ is the state estimate and system input at time t_{k-1}. The state transition matrix $\boldsymbol{\Phi}_k$ and the input matrix $\boldsymbol{\Delta}_k$ are time varying matrices due to change in $\Delta t = t_k - t_{k-1}$.

The Kalman filter equations for a filter with asynchronous update rate can be given on the form:

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{\Phi}_k \boldsymbol{x}_{k-1} + \boldsymbol{\Delta}_k \boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \\ \boldsymbol{z}_k &= \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{3.20}$$

where $\boldsymbol{x}_k$ is the process state vector at time t_k, $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Delta}_k$ is the state and input transition matrix respectively, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ is the zero-mean Gaussian white noise, $\boldsymbol{z}_k$ is the measurement at time t_k and $\boldsymbol{H}_k$ is the connection between the measurement and state vector.

Furthermore, Brown and Hwang (1997) describes a method for updating the filter with one measurement at a time. The $\boldsymbol{H}$ matrix can be rewrite in partitioned form as:

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{H}_k^a \\ \hdashline \boldsymbol{H}_k^b \\ \hdashline \vdots \end{bmatrix} \tag{3.21}$$

Where each block represent its corresponding measurement a, b ... . When updating the filter with a measurement, all blocks are set to zero except the block representing the measurement that is updating the filter. This filter update sequence is illustrated in figure 3.4

As figure 3.4 illustrates, the filter is updated if there is a new measurement available. If there are multiple measurements available at the same time-step k, the filter iterates

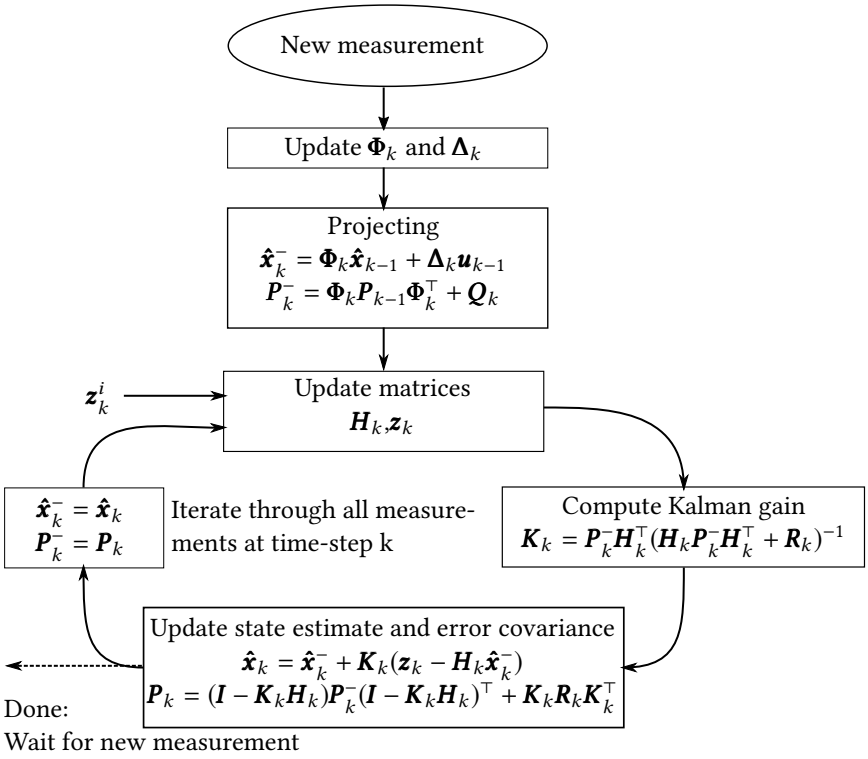


Figure 3.4: Updating the Kalman filter with sensor input

trough all the measurements and updates the filter with one measurement at a time.

There is also possible to update the states and error covariance between measurement by performing dead reckoning. State estimate request can for instance be required from a controller or user interface. One simple way to implement dead reckoning in practice, is to use the same filter update sequence as given in figure 3.4 and set the H_k matrix to zero, this updates both the state estimate and its corresponding error covariance.

3.3.1 Position and linear velocity state estimation

For applications concerning landing pads that have zero or small pitch and roll angles, a position and linear velocity state estimation may be suitable.

In the application of landing on a moving landing pad, the most essential states are the relative distance between the UAV and the landing pad and the velocity of the landing pad, often referred to as speed over ground. For convenience, the states and the measurements are all given relative to NED. The states are therefore chosen to be the 6×1 vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^n_{l/u} \\ \boldsymbol{v}^n_{l/n} \end{bmatrix} \tag{3.22}$$

By using a constant velocity approximation of the UAV and landing pad, the dynamic equations can be stated as

$$\boldsymbol{p}^n_{l/u,k} = \boldsymbol{p}^n_{l/u,k-1} + (\boldsymbol{v}^n_{l/n,k-1} - \boldsymbol{v}^n_{u/n,k-1})(t_k - t_{k-1}) + \boldsymbol{w}_k \tag{3.23}$$

$$\boldsymbol{v}^n_{l/n,k} = \boldsymbol{v}^n_{l/n,k-1} + \boldsymbol{w}_k \tag{3.24}$$

where t_k and t_{k-1} are the time-stamps in seconds of the time-steps k and $k-1$ respectively. Figure 3.5 gives a graphical illustration of how the relative position vector $\boldsymbol{p}^n_{l/u}$ can be written as a function of the UAV- and landing pad position vectors, $\boldsymbol{p}^n_{u/n}$ and $\boldsymbol{p}^n_{l/n}$.

The equations in 3.23 can be written on the Kalman filter equation form of given in 3.20 by selecting the matrices Φ_k, Δ_k and $\boldsymbol{H}_k$ as

$$\Phi_k = \begin{bmatrix} \boldsymbol{I} & \Delta\boldsymbol{t} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \qquad \Delta_k = \begin{bmatrix} -\Delta\boldsymbol{t} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.25}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix, $\boldsymbol{0}$ is a 3×3 matrix with all elements equal to zero and $\Delta\boldsymbol{t}$ is a 3×3 matrix with the relative time-step $(t_k - t_{k-1})$ on its diagonal and zeros on the none-diagonal elements. It is worth nothing that the Φ and Δ matrices are given in the time-step k and not $k-1$. The system is initially represented as time invariant,

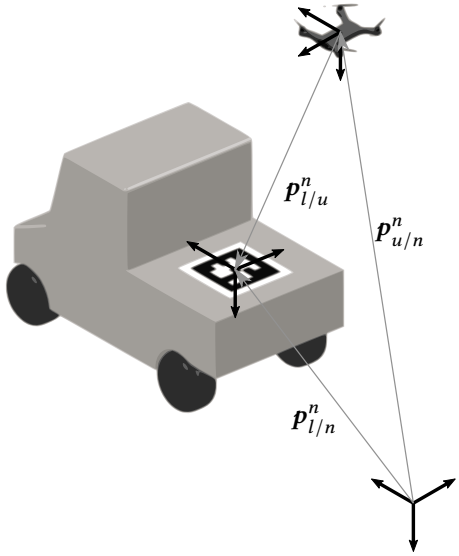


Figure 3.5: Position vectors

but the inconsistence in the time-steps makes the system matrices a function of time. The input vector $\boldsymbol{u}$ and measurement vector $\boldsymbol{z}$ are then given as

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{v}^n_{u/n} \end{bmatrix} \qquad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{u/n} \\ \boldsymbol{p}^n_{l/u} \\ \boldsymbol{v}^n_{l/n} \end{bmatrix} \tag{3.26}$$

where $\boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{u/n}$ represents the position measurements from the landing pad subtracted by the position measurements from the UAV. The second row in the measurement vector $\boldsymbol{z}$ corresponds to the relative ArUco measurement $\boldsymbol{p}^n_{l/u}$ and the last row is the landing pad velocity measurement transmitted from the landing pad.

3.3.2 Position, linear velocity and bias state estimation

As given in section 3.2.4, the position estimate from the UAV and landing pad includes the biases $\boldsymbol{\beta}^n_{l/n}$ and $\boldsymbol{\beta}^n_{u/n}$ respectively. These biases varies and should therefore be estimates during the flight. One method to deal with these biases is to estimates them in the Kalman filter as extra states. To reduce the number of states in the filter, the biases can be reduced to a relative bias given as

$$\boldsymbol{\beta}^n_{l/u} = \boldsymbol{\beta}^n_{l/n} - \boldsymbol{\beta}^n_{u/n} \tag{3.27}$$

By adding the the bias states given in equation 3.27 to the filter, the state vector and filter equations are extended to

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^n_{l/u} \\ \boldsymbol{v}^n_{l/n} \\ \boldsymbol{\beta}^n_{l/u} \end{bmatrix} \tag{3.28}$$

and

$$\boldsymbol{p}^n_{l/u,k} = \boldsymbol{p}^n_{l/u,k-1} + (\boldsymbol{v}^n_{l/n,k-1} - \boldsymbol{v}^n_{u/n,k-1})(t_k - t_{k-1}) + \boldsymbol{w}_k \tag{3.29}$$

$$\boldsymbol{v}^n_{l/n,k} = \boldsymbol{v}^n_{l/n,k-1} + \boldsymbol{w}_k \tag{3.30}$$

$$\boldsymbol{\beta}^n_{l/u,k} = \boldsymbol{\beta}^n_{l/u,k-1} + \boldsymbol{w}_k \tag{3.31}$$

where the bias is modeled as a Gaussian random walk model. The corresponding state transition matrix, input transition matrix and measurement connection matrix are then given as.

$$\Phi_k = \begin{bmatrix} \boldsymbol{I} & \Delta\boldsymbol{t} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \qquad \Delta_k = \begin{bmatrix} -\Delta\boldsymbol{t} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \tag{3.32}$$

As the measurement connection matrix $\boldsymbol{H}_k$ illustrates, the relative position measurement from the UAV and landing pad does also include the bias $\boldsymbol{\beta}^n_{l/u}$. This gives the

input and measurement vectors

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{v}^n_{u/n} \end{bmatrix} \qquad z = \begin{bmatrix} \boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{u/n} + \boldsymbol{\beta}^n_{l/u} \\ \boldsymbol{p}^n_{l/u} \\ \boldsymbol{v}^n_{l/n} \end{bmatrix} \tag{3.33}$$

3.3.3 Full state estimation

In addition to the relative position, landing pad velocity and bias, the relative orientation and landing pad angular velocity are added as states in the full state estimator.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^n_{l/u} \\ \Theta_{ul} \\ \boldsymbol{v}^n_{l/n} \\ \boldsymbol{\omega}^n_{l/n} \\ \boldsymbol{\beta}^n_{l/u} \end{bmatrix} \tag{3.34}$$

Here, the measurement and input vectors z and u, are supplemented with the measurements of orientation and angular velocities.

$$z = \begin{bmatrix} \boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{u/n} + \boldsymbol{\beta}^n_{l/u} \\ \boldsymbol{p}^n_{l/u} \\ \Theta_{ul} \\ \boldsymbol{v}^n_{l/n} \\ \boldsymbol{\omega}^n_{l/n} \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} \boldsymbol{v}^n_{u/n} \\ \boldsymbol{\omega}^n_{u/n} \end{bmatrix} \tag{3.35}$$

where Θ_{ul} is measured form the fiducial marker, $\boldsymbol{\omega}^n_{l/n}$ is measured from the landing pad navigation system and $\boldsymbol{\omega}^n_{u/n}$ is measured from the UAV navigation system. By still using the approach of constant velocity on the landing pad, we can formulate the

discrete dynamic equations as

$$
\begin{aligned}
\boldsymbol{p}^n_{l/u,k} &= \boldsymbol{p}^n_{l/u,k-1} + (\boldsymbol{v}^n_{l/n,k-1} + \boldsymbol{v}^n_{u/n,k-1})(t_k - t_{k-1}) + \boldsymbol{w}_k \\
\boldsymbol{\Theta}_{ul,k} &= \boldsymbol{\Theta}_{ul,k-1} + (\boldsymbol{\omega}^n_{l/n,k-1} + \boldsymbol{\omega}^n_{u/n,k-1})(t_k - t_{k-1}) + \boldsymbol{w}_k \\
\boldsymbol{v}^n_{l/n,k} &= \boldsymbol{v}^n_{l/n,k-1} + \boldsymbol{w}_k \\
\boldsymbol{\omega}^n_{l/n,k} &= \boldsymbol{\omega}^n_{l/n,k-1} + \boldsymbol{w}_k \\
\boldsymbol{\beta}^n_{l/u,k} &= \boldsymbol{\beta}^n_{l/u,k-1} + \boldsymbol{w}_k
\end{aligned}
$$

This gives the state and input transition matrices

$$
\Phi_{k-1} = \begin{bmatrix}
\boldsymbol{I} & \boldsymbol{0} & \Delta\boldsymbol{t} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \Delta\boldsymbol{t} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}
\end{bmatrix}
$$

$$
\Delta = \begin{bmatrix}
\Delta\boldsymbol{t} & \boldsymbol{0} \\
\boldsymbol{0} & \Delta\boldsymbol{t} \\
\boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0}
\end{bmatrix}
$$

And the measurement matrix

$$
\boldsymbol{H}_k = \begin{bmatrix}
\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\
\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0}
\end{bmatrix}
$$

3.3.4 State Estimation for Static Landing Pad

For applications where we have the prior knowledge that the landing will be static during the flight, this information can be used in the state estimator to improve the estimates. One method of adjusting the Kalman filter in the reduced state estimator, is to modify the Φ_k matrix such that the dynamic equation describing the velocity of the landing pad is stated as

$$\boldsymbol{v}^n_{l/n,k} = 0 + \boldsymbol{w}_k \tag{3.36}$$

Furthermore, the elements in the covariance matrix $\boldsymbol{Q}_k$, representing the landing pad velocity estimate is to be set close to zero. For the full state estimation filters, both the linear- and angular velocity is to be set to zero, and the corresponding elements in the covariance matrix is to be set close to zero.

$$\boldsymbol{v}^n_{l/n,k} = 0 + \boldsymbol{w}_k \tag{3.37}$$

$$\boldsymbol{\omega}^n_{l/n,k} = 0 + \boldsymbol{w}_k \tag{3.38}$$

Chapter 4

Controller

4.1 Controller Logic

The autonomous landing system developed in this work is just a piece of FFI's autonomous UAV platform. All control systems and sensor modules are controlled by the Hybrid Autonomous Layer (HAL) module, that takes the high level decisions and delegates task to underlaying modules. For instance, during a search and rescue mission HAL takes the decision of when the UAV have to change battery. HAL then requests autonomous landing on to the battery exchange robot mounted on a moving vehicle. The autonomous landing system then starts performing its landing sequence. At any time during the landing operation, HAL may reschedule and disable the landing. For safety reasons, the velocity set-point generated by the landing controller, and all other control modules are sent to a safety node. The safety node checks the velocity set-point and restricts it if necessary before it sends the velocity command to the UAV flight controller. The safety module do also switch to manual control as soon as someone uses the remote control.

The set-point and HAL communication-flow described in this section are illustrated in Figure 4.1

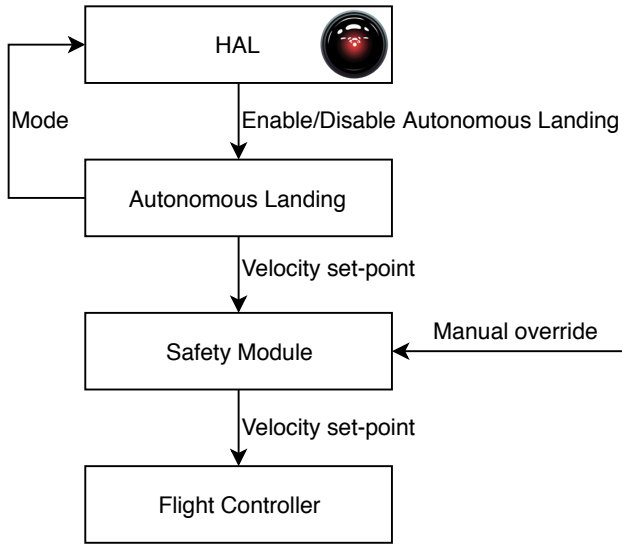


Figure 4.1: Velocity set-point and HAL communication flow chart

4.1.1 State machine

This section takes a close look at the state machine used to implement the autonomous landing system. The state machine in this section returns position set-point and a controller gain to be fed in to a target tracking controller. Two target tracking controllers are derived in closer details in section 4.2. The state machine do also return a boolean output to the flight controller to trigger a force land command. This command tels the flight controller to ignore the velocity set-point and just go straight down and turn of its propellers when it touches the ground.

The state machine is built up with seven states. Stop, Intercept, Hover, Lower, Gain adjust, Final stage and Land. Figure 4.2 gives a graphical illustration of the different states. The state stop, is only activated when autonomous landing is disabled and will the return zero as velocity output. The intercept state has the purpose of bringing the UAV to the hover point located at a fixed distance above the landing pad. In hover

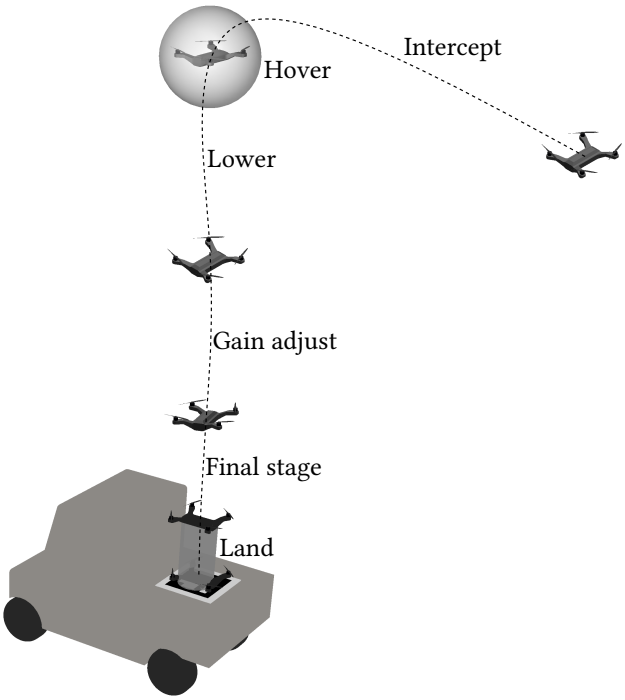


Figure 4.2: State machine states visualized for a UAV landing on a vehicle

mode, the UAV will stay in this hover point fixed above the landing pad while the position state estimator improves its estimates. This hover point is selected at a certain height that ensures the landing pad to be fixed within the camera field of view. The state machine will stay in this state until the covariance of the position estimate reaches an acceptable level. The lower state lowers the UAV towards the landing pad at a given descent velocity. When the UAV reaches a certain height, the state machine enters the gain adjust state. This state adjusts the controller gain to a more aggressive gain to ensure a more precise position control. At the final stage state, the descent velocity is reduced to get an even more precise position control. And finally, the land states triggers the force land command to the flight controller, which brings the UAV all the

way down to the landing pad and disarms the motors. Algorithm 1 lists a pseudo code to summarize the state machine outputs in the different states.

Algorithm 1 State machine output

$\Delta t = getCurrentTime() - t_0$
if $state = stop$ **then**
    $Stop$
else if $state = intercept$ **or** $hover$ **then**
    $gain \leftarrow gain_1$
    $pos_{sp} \leftarrow hooverHeight$
else if $state = lower$ **then**
    $gain \leftarrow gain_1$
    $pos_{sp} \leftarrow pos_{sp} - decentVel_u \Delta t$
else if $state = gainAdjust$ **then**
    $gain \leftarrow gain_2$
    $pos_{sp} \leftarrow pos_{sp} - decentVel_u \Delta t$
else if $state = finalStage$ **then**
    $gain \leftarrow gain_2$
    $pos_{sp} \leftarrow pos_{sp} - decentVel_l \Delta t$
else if $state = land$ **then**
    $land \leftarrow True$
$t_0 = getCurrentTime()$

In the pseudo code given above, $land$ is the boolean output to the flight controller that triggers the landing command, $gain$ is the controller gain in the UAV position controller and pos_{sp} is the position set-point to the same controller. The parameters $gain_1$, $gain_2$, $hoverHeight$, $decentVel_u$ and $decentVel_l$ are parameters tunable from a parameter file.

The logic for switching between the states in the state machine are given in algorithm 2. As the pseudo code indicates, the switching to the hover and land state are triggered by the presence of the UAV inside a given sphere and cylinder respectively. These "imaginary" geometrical figures are illustrated in figure 4.2 and have tunable dimensions.

The parameters $hoverSphere$, $min\ covar$, $height_{ga}$, $height_{fs}$ and $landingCylinder$ given in algorithm 2 are also tunable in a given parameter file. The variable $abort$ is

Algorithm 2 State machine shifting logic

```
1: if autonomousLanding = False then
2:     state ← stop
3: else if abort = true then
4:     state ← intercept
5: else if state = intercept then
6:     if UAV_pos ∈ hoverSphere then
7:         state ← hover
8: else if state = hover then
9:     if nav covar ≤ min covar then
10:        state ← lower
11: else if state = lower then
12:    if UAV_height ≤ height_ga then
13:        state ← gainAdjust
14: else if state = gainAdjust then
15:    if UAV_height ≤ height_fs then
16:        state ← finalStage
17: else if state = finalStage then
18:    if UAV_pos ∈ landingCylinder then
19:        state ← land
```

a control variable given from a condition monitoring and fault detection system. A condition monitoring and fault detection system will not be included in this work.

4.2 Guidance Methods

The objective of this section is to define several methods for having the UAV matching both the position and speed of the LP. This objective is often referred to as rendezvous in the space industry and corresponds to the motion control objective in a target tracking scenario (Breivik (2010)). In guidance theory, the interceptor and target are also referred to as evader and pursuer or predators and pray (Shneydor (1998b)). Further on in this section, UAV and LP will be used as synonyms for interceptor and target, respectively.

The target tracking scenario in this assignment can be states as

$$\lim_{t\to\infty}(\boldsymbol{p}^n_{u/n} - \boldsymbol{p}^n_{d/n}) = 0 \tag{4.1}$$

$$\lim_{t\to\infty}(\boldsymbol{v}^n_{u/n} - \boldsymbol{v}^n_{l/n}) = 0 \tag{4.2}$$

where

$$\boldsymbol{p}^n_{d/n} = \boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{h/l} \tag{4.3}$$

is the desired tracking point and $\boldsymbol{p}^n_h = [0, 0, h]^\top \in \mathbb{R}^3$ is a vector defining the hover height h above the landing pad.

4.2.1 Parallel Navigation Guidance

Parallel navigation has for long been used for mariners to avoid collisions at sea, and is therefor also known under the names "constant bearing" and "collision course navigation" (Shneydor; 1998a). The parallel navigation notation originate from its use in air-to-air missile guidance applications (Fossen; 2011), where the main objective is to achieve collision with the target. Breivik (2010) applies the parallel navigation concept together with an asymptotic interception method to form a stable motion control of marine craft. Their paper proves stability for surface position and heading

for a fully actuated marine surface craft performing motion control using the constant bearing guidance method.

The fundamental of the parallel guidance is that the UAV velocity is a combination of a parallel velocity of the landing pad and a velocity vector pointing directly at the landing pad. Figure 4.3 and equation 4.4 illustrates this method in further details.

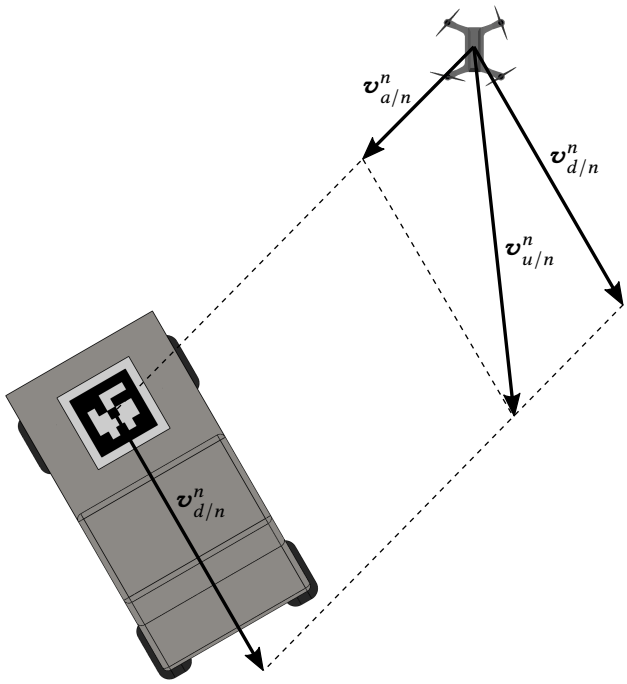


Figure 4.3: Velocity vectors in the Parallel navigation guidance
2D representation of the velocity vectors in the Parallel navigation guidance

$$\boldsymbol{v}^n_{u/n} = \boldsymbol{v}^n_{d/n} + \boldsymbol{v}^n_{a/n} \tag{4.4}$$

where $\boldsymbol{v}^n_{a/n}$ is the approach velocity pointing in the same direction as the line of sight vector between the UAV and the landing pad. $\boldsymbol{v}^n_{d/n}$ is the desired position velocity, and $\boldsymbol{v}^n_{u/n}$ is the velocity vector send as set-point to the UAV flight controller (Fossen; 2011).

A simple kinematic model of a quadcopter can be modeled as

$$\dot{\boldsymbol{p}}^n_{u/n} = \boldsymbol{v}^n_{u/n} \tag{4.5}$$

By applying the kinematic control design given in Breivik and Fossen (2007) as a three axis position UAV controller, the stability proof in the same paper can be rewritten to fit the kinematic equation given in 4.5. Equation 4.6 states a continuously differentiable positive definite Lyapunov function candidate for the differential equation in 4.5

$$V = \frac{1}{2}(\boldsymbol{p}^n_{u/d})^\top \boldsymbol{p}^n_{u/d} \tag{4.6}$$

where

$$\boldsymbol{p}^n_{u/d} = \boldsymbol{p}^n_{u/n} - \boldsymbol{p}^n_{d/n} \tag{4.7}$$

is the relative position between the UAV and its desired position $\boldsymbol{p}^n_{d/n}$. The time derivative of $\boldsymbol{p}^n_{u/d}$ is

$$\dot{\boldsymbol{p}}^n_{u/d} = \dot{\boldsymbol{p}}^n_{u/n} - \dot{\boldsymbol{p}}^n_{d/n} = \boldsymbol{v}^n_{u/n} - \boldsymbol{v}^n_{d/n} \tag{4.8}$$

Differentiating the Lyapunov function candidate given in 4.6 with respect to time along the trajectories of $\boldsymbol{p}^n_{u/d}$ gives

$$\dot{V} = (\boldsymbol{p}^n_{u/d})^\top \dot{\boldsymbol{p}}^n_{u/d} = (\boldsymbol{p}^n_{u/d})^\top (\boldsymbol{v}^n_{u/n} - \boldsymbol{v}^n_{d/n}) \tag{4.9}$$

Implementing the parallel navigation guidance method gives the controller

$$\boldsymbol{v}^n_{u/n} = \boldsymbol{v}^n_{d/n} - \boldsymbol{\kappa}\frac{\boldsymbol{p}^n_{u/d}}{|\boldsymbol{p}^n_{u/d}|} \tag{4.10}$$

where $\boldsymbol{\kappa}$ is a controller gain. This parallel navigation guidance controller is designed to achieve collision with the target. Breivik and Fossen (2007) implements an asymptotic

gain to achieve asymptotically interception velocity with the desired position. This is realized by selecting the controller gain as

$$\boldsymbol{\kappa} = U_{c,max} \frac{|\boldsymbol{p}^n_{u/d}|}{\sqrt{(\boldsymbol{p}^n_{u/d})^\top \boldsymbol{p}^n_{u/d} + \Delta^2}} \tag{4.11}$$

where $U_{c,max}$ is the maximum approach velocity and Δ is a tunable control parameter. By adding the asymptotic gain given in 4.11 in to the parallel navigation guidance method in 4.10, the controller can be rewritten as

$$\boldsymbol{v}^n_{u/n} = \boldsymbol{v}^n_{d/n} - U_{c,max} \frac{\boldsymbol{p}^n_{u/d}}{\sqrt{(\boldsymbol{p}^n_{u/d})^\top \boldsymbol{p}^n_{u/d} + \Delta^2}} \tag{4.12}$$

substituting 4.12 in to 4.9 yields

$$\dot{V} = -U_{c,max} \frac{(\boldsymbol{p}^n_{u/d})^\top \boldsymbol{p}^n_{u/d}}{\sqrt{(\boldsymbol{p}^n_{u/d})^\top \boldsymbol{p}^n_{u/d} + \Delta^2}} < 0, \quad \forall \boldsymbol{p}^n_{u/d} \neq 0 \tag{4.13}$$

which is negative definite when Δ and $U_{c,max}$ are positive and bounded. By using Lyapunov's theorem (Khalil; 2015), the system is shown to be Uniformly Globally Asymptotically Stable (UGAS) (Fossen; 2011). Moreover, Belleter (2016) proves in his thesis that the system is Uniform Semiglobal Exponential Stability (USGES).

4.2.2 Optimal Guidance

Optimal Guidance or optimal control problems aim to find the optimal control output that minimizes a control objective at the same time as some given constraints are met. A reasonable control objective to minimize in the subject of landing a UAV on a platform in motion, is to use the distance between the UAV and the tracking point, in addition to the relative velocity between them.

A general quadratic objective function with linear constraints for a discrete time

systems can be written as (Foss and Heirung; 2013)

$$f(z) = \sum_{k=0}^{N} f(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) = \sum_{k=0}^{N-1} \frac{1}{2} \boldsymbol{x}_{k+1}^{\top} \boldsymbol{Q}_{k+1} \boldsymbol{x}_{k+1} + \frac{1}{2} \boldsymbol{u}_k^{\top} \boldsymbol{R}_k \boldsymbol{u}_k \tag{4.14}$$

subject to

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k \tag{4.15}$$

$$\boldsymbol{x}^{low} \leq \boldsymbol{x}_k \leq \boldsymbol{x}^{high} \tag{4.16}$$

$$\boldsymbol{u}^{low} \leq \boldsymbol{u}_k \leq \boldsymbol{u}^{high} \tag{4.17}$$

$$-\Delta \boldsymbol{u}^{high} \leq \Delta \boldsymbol{u}_k \leq \Delta \boldsymbol{u}^{high} \tag{4.18}$$

where

$$\Delta \boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1} \tag{4.19}$$

$$\boldsymbol{z}^{\top} = (\boldsymbol{x}_1^{\top}, \ldots, \boldsymbol{x}_N^{\top}, \boldsymbol{u}_0^{\top}, \ldots, \boldsymbol{u}_{N-1}^{\top}) \tag{4.20}$$

$$n = N(n_x + n_u) \tag{4.21}$$

$$\boldsymbol{x}_0, \boldsymbol{u}_{-1} \text{ is given} \tag{4.22}$$

$$\boldsymbol{Q}_k \geq 0 \tag{4.23}$$

$$\boldsymbol{R}_k \geq 0 \tag{4.24}$$

furthermore, N is the prediction horizon, $\boldsymbol{x}$ and $\boldsymbol{u}$ are decision variables and represents the system states and control input respectively and k is the time step from $k = 0$ to $k = N$. The vectors $\boldsymbol{x}^{low}$, $\boldsymbol{x}^{high}$, $\boldsymbol{u}^{low}$ and $\boldsymbol{u}^{high}$ contains the upper and lower limits of the states and input respectively. Additionally, the vectors $\Delta \boldsymbol{u}^{high}$ and $\Delta \boldsymbol{u}^{low}$ enable constraints on the change of control input. The $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ matrices in the objective function 4.14 are matrices weighting the individual states and control inputs. These weighting matrices and the prediction horizon N are used for tuning the optimization function to give its desired behavior.

One way to implement optimal guidance by using the general quadratic objective

function given in 4.14 and its corresponding constraints (4.19-4.24), is by using the relative position and velocity between the UAV and the desired tracking point as states and their respective velocities relative to the n frame as control input.

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}^n_{d/u} \\ \boldsymbol{u}^n_{d/u} \end{bmatrix} \qquad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{u}^n_{u/n} \\ \boldsymbol{u}^n_{d/n} \end{bmatrix} \tag{4.25}$$

Furthermore, a constant velocity discrete time model can be implemented in the equality constraints 4.15,

$$\boldsymbol{p}^n_{d/u,k+1} = \boldsymbol{p}^n_{d/u,k} + \Delta \boldsymbol{t}\, \boldsymbol{u}^n_{d/u,k} \tag{4.26}$$

$$\boldsymbol{u}^n_{d/u,k+1} = \boldsymbol{u}^n_{d/n,k} - \boldsymbol{u}^n_{u/n,k} \tag{4.27}$$

Gives the $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ matrices

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{I} & \Delta \boldsymbol{t} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{B}_k = \begin{bmatrix} 0 & 0 \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \tag{4.28}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix, $\boldsymbol{0}$ is a 3×3 matrix with all elements equal to zero and $\Delta \boldsymbol{t}$ is a 3×3 matrix with the relative time-step $(t_{k+1} - t_k)$ on its diagonal and zeros on the non-diagonal elements. By setting the constraints representing the control input for the desired tracking point to the measured velocity of the tracking point, ensures constant velocity estimate of the desired tracking point. The constraints representing the control input for the UAV is set to the desired maximum velocity for the UAV.

Solving the quadratic objective function 4.14 with the linear constraints in 4.19-4.24, the states and control input given in 4.25 and their system matrices in 4.28 gives an open loop solution to the Optimal Guidance problem (Foss and Heirung; 2013). Feedback can be added to the Guidance method by introducing the linear Model Predictive Control (MPC) (Foss and Heirung; 2013). Algorithm 3 taken from Foss and Heirung (2013), describes how the linear MPC with state feedback closes the loop. The use of MPC with state feedback requires that all the states are measured. In this application, this

Algorithm 3 MPC with state feedback

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Get current states $\boldsymbol{x}_k$
3: Solve problem 4.14 on the prediction horizon from k to $k + N$ with $\boldsymbol{x}_k$ as initial condition
4: Apply the first control move $\boldsymbol{u}_t$ from the solution

require that the relative position $\boldsymbol{p}^n_{d/u}$ and velocity $\boldsymbol{u}^n_{d/u}$ are measured.

Chapter 5

Implementation and Test Setup

5.1 Implementation

5.1.1 Robotics Operating System

Robotic Operating System (ROS) is an open source framework for building robotic applications developed by contributers in the Open Source Robotics Foundation (Open Source Robotics Foundation; 2018a). The ROS framework consists of a set of software libraries and tools for developing robotic applications in a modular way. A system running on ROS consists of multiple nodes communicating through topics. Figure 5.1 illustrates how the nodes are connected using topics in the implementation done in this assignment. The green nodes *odometry* and *safety_module* are made by FFI, the nodes *node_aruco*, *node_navigation* and *node_controller* marked blue are developed by the author and the read nodes are made by other contributers to the ROS community. All the nodes implemented by the author are implemented as ROS nodes by using the programming language c++. The ROS nodes are tested on ROS version 1.12.13 (Kinetic). The corresponding table 5.1 lists all the topics included in the */mavros/**

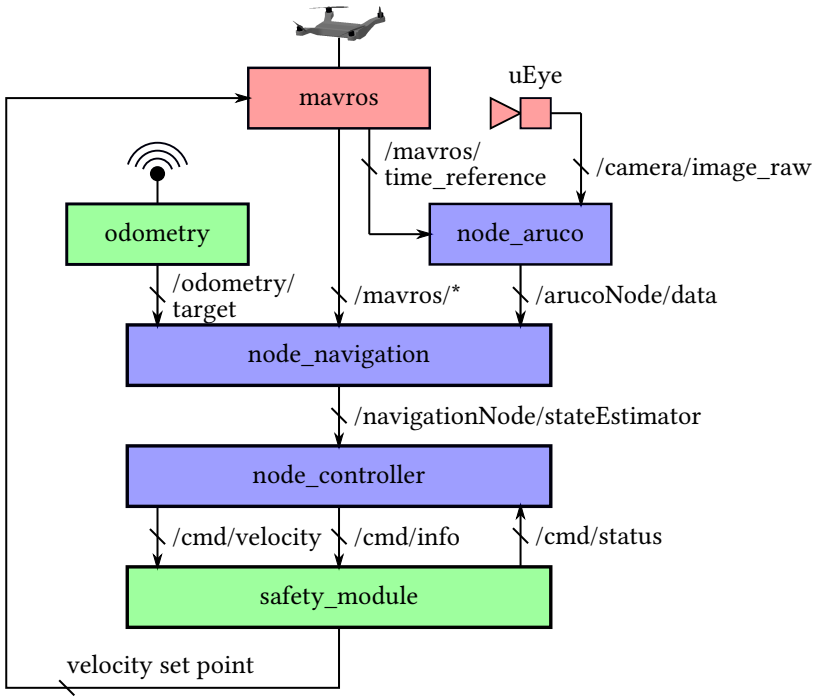


Figure 5.1: ROS node overview

branch in figure 5.1. Furthermore, table 5.2 and 5.3 lists all the measurements read

/mavros/state
/time_reference
/global_position/global
/imu/data
/local_position/velocity

Table 5.1: Topics in /mavros/* from figure 5.1

from the UAV and the landing pad.

Message	Measurement
/mavros/global_position/global	$\boldsymbol{p}_u^{ge}$
/mavros/global_position/global/position_covariance	$\boldsymbol{r}_{p_u^e}$
/mavros/imu/data/orientation	$\boldsymbol{q}_{eu}$
/mavros/local_position/velocity/twist/linear	$\boldsymbol{v}_{u/e}^e$
/mavros/local_position/velocity/twist/angular	$\boldsymbol{\omega}_{u/e}^e$

Table 5.2: Messages on the mavros topic received from the UAV

Message	Measurement
/odometry/target/pose/pose/position	$\boldsymbol{p}_l^{ge}$
/odometry/target/pose/pose/orientation	$\boldsymbol{q}_{el}$
/odometry/target/pose/covariance	$\boldsymbol{r}_{l,pose}$
/odometry/target/twist/twist/linear	$\boldsymbol{v}_{l/e}^e$
/odometry/target/twist/twist/angular	$\boldsymbol{\omega}_{l/e}^e$
/odometry/target/twist/covariance	$\boldsymbol{r}_{l,twist}$

Table 5.3: Messages on the odometry topic received from the landing pad

5.1.2 Implemented Nodes

5.1.2.1 node_aruco

The main objective of the *node_aruco* is to, from an image stream, get a measurement of the position and orientation of a fiducial marker relative to the camera. A flow chart given in figure 5.2 illustrates the structure of the node. The image matrix received from the uEye camera contains of $m \times n$ 8 bit unsigned integers, where $m \times n$ represents the camera resolution. Each matrix element gives the intensity of the pixel in the corresponding image. The image matrix is send in to the *cv::aruco::detectMarkers* function included in the OpenCV library (Itseez; 2018). The function returns an array with the ID of the markers detected and its corresponding corners. Furthermore, the pose is estimated for each of the detected markers using the

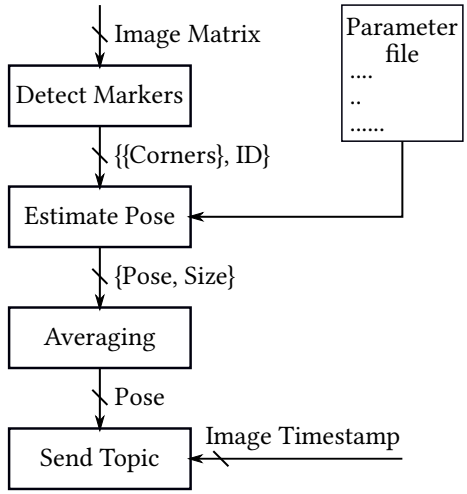


Figure 5.2: Flow chart of the node_aruco

function *cv::aruco::estimatePoseSingleMarkers* and parameters given in a parameter file. Parameters given in the parameter file are constants describing the positions, ID's and dimensions of the ArUco tags creating a multi marker tag. Moreover, the parameter file includes camera calibration matrices. Pose and tag dimensions are then used in an averaging algorithm to improve the pose measurements. The position averaging method used is given in equation 5.1 and are used for ArUco markers averaging in the paper from Tørdal and Hovland (2017).

$$\bar{\boldsymbol{p}} = \frac{1}{A}\sum_{i=1}^{N_m} a_i \boldsymbol{p}_i, \qquad A = \sum_{i=1}^{N_m} a_i \tag{5.1}$$

where a_i and $\boldsymbol{p}_i$ are the respectively tag area and position vector, N_m is the number of tags to be averaged and $\bar{\boldsymbol{p}}$ is the resulting average vector. Finally, the averaged pose estimate is tagged with the time stamp from the image and sent on the */arucoNode/data* topic. Moreover in the initialization phase, the local n frame is set to the point measured by the UAV GNSS sensor.

5.1.2.2 node_navigation

When the navigation node is started, it waits for the UAV to get armed. This is to ensure that the UAV have GNSS fix and to ensure that the user have placed the UAV at its takeoff position. Thereafter, the Kalman filter is initialized, the global NED is defined and the time difference between the SBC time and GNSS time is calculated. The filter

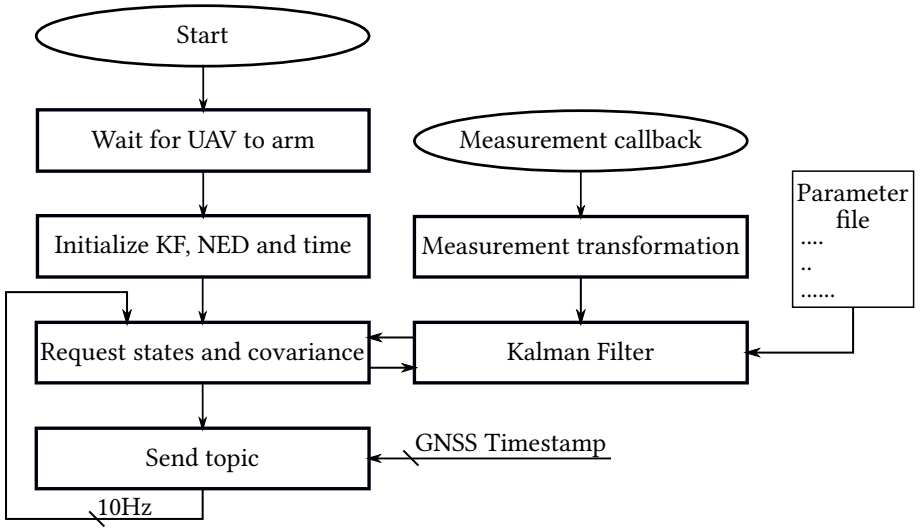


Figure 5.3: Flow chart of the node_navigation

can either be automatically- or manually initialized, selected by a parameter. If the filter is automatically initialized, the UAV have to be placed in the center of the landing pad before initialization. In automatic initialization, the relative position estimate $\boldsymbol{p}^n_{l/u}$ is set to zero, the landing pad velocity $\boldsymbol{v}^n_{l/n}$ is set to the measured velocity $\boldsymbol{v}^n_{l/n}$ and the bias term $\boldsymbol{\beta}^n_{l/u}$ is set to measured relative position $\boldsymbol{p}^n_{l/n} - \boldsymbol{p}^n_{u/n}$. The covariance estimate is set to predefined values based on experience set in the parameter file. Manually initialization uses predefined values from the parameter file both on the state and covariance estimate.

To be able to match the measurements from the landing pad with the measurements form the UAV and camera, all measurement are stamped with GNSS time. Messages

from the camera and the UAV are originally stamped with time from the computer clock which differs from the GNSS time. In the initialization phase, a time difference $\Delta_{ros/GNSS}$ is calculated by taking the difference between *ros::Time::now()* and GNSS time. The time stamps from the camera and the UAV measurements are then adjusted to GNSS time by adding the $\Delta_{ros/GNSS}$ to the initial time stamp. After the filter is initialized, it starts updating the filter from measurement updates. When a measurement arrives, the measurement are transformed in to its relevant coordinate frame by using the conversion methods given in section 3.2. Due to delays on the camera- and Landing Pad measurements, a method to match the corresponding measurements of the UAV orientation and position is needed. The method used in *node_navigation* stores the latest second of UAV orientation an position measurements and their corresponding time stamps in dynamic buffers. When a camera or landing pad measurement arrives, the corresponding UAV orientation and position are calculated using the dynamic array and linear interpolation between the stored measurements.

As discussed in section 3.3, there are two ways to update the Kalman filter is updated. Either by updating the filter with new measurements or by dead reckoning on request. As illustrated in figure 5.3, *node_navigation* requests the estimated states and their corresponding covariance, times the states with the GNSS time stamp and sends them out on the */navigationNode/stateEstimator* topic ten times a second.

5.1.2.3 node_controller

The flow chart given in figure 5.4 describes the work flow of the *node_controller*. The state machine referred to in the figure is defined in section 4.1.1, where status from the safety module and the estimated states from the Kalman filter are given as inputs. Furthermore, a position set point $\boldsymbol{p}^n_{d/n}$ and its velocity $\boldsymbol{v}^n_{d/n}$ together with a controller gain Δ are set as input to the Parallel Navigation Guidance Controller given in section 4.2.1. The velocity set point $\boldsymbol{v}^n_{u/n,sp}$ from the Parallel Navigation Guidance Controller is sent as a velocity command on a ROS topic.

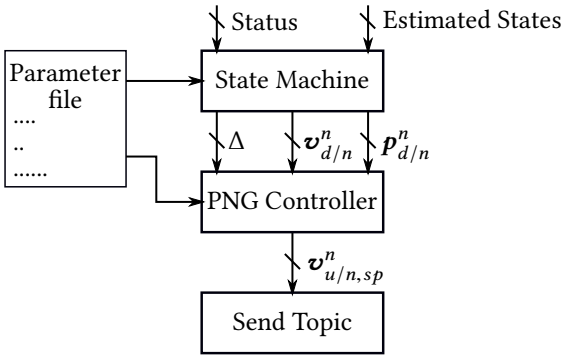


Figure 5.4: Flow chart of the node_controller

5.2 Test Setup

5.2.1 Communication

The communication setup between the UAV, landing pad, manual controller and the work station is illustrated in figure 5.5. Figure 5.6 shows a photograph of the router, UAV, and the Custom Sensor Unit (CSU) used in the tests.

5.2.2 Quadcopter

The quadcopter used in this work is a 3DR Solo Drone produced by 3D Robotics photographed in figure 5.7. The UAV is a low cost commercial available quadcopter mainly used by hobbyists and photographers. In addition to the off the shelf quadcopter, a SBC, a wireless adapter and a global shutter camera is connected to drone. Figure 5.8 illustrates how the equipment is mounted on to the UAV and table 5.4 lists the details of the equipment. The SBC is running ROS on Ubuntu 16.04 and communicates with the quadcopter using the mavROS interface (Open Source Robotics Foundation; 2018b).

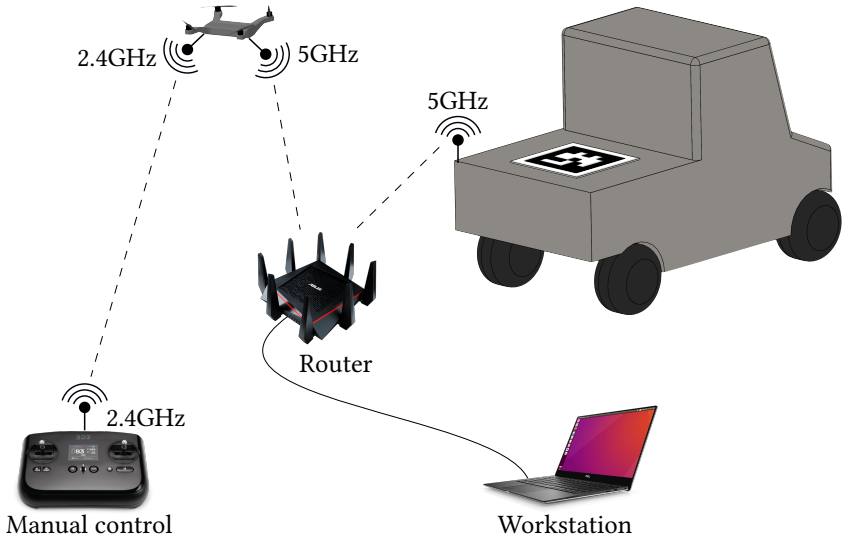


Figure 5.5: Communication setup

Device	Brand	Type nr
Quadcopter	3D Robotics	SOLO Drone
Single Board Computer	Hardkernel	ODROID-C2
Wireless Adapter	Edimax	ew-7811uac
Camera	iDS	UI-3250LE-C-HQ

Table 5.4: List of Devices Quadcopter

5.2.3 Landing Pads

The fiducial marker used in this test setup, is a 80 times 80cm PRiAM tag mounted on a wooden board. Tests have been carried out with the tag placed on the ground, at the back of the ground vehicle Olav and at the deck of the surface vehicle Odin. Both Olav and Odin have accurate navigation systems on board broadcasting the navigation data to the UAV. In order to have a velocity and position measurement of the landing pad when using other landing pads, a CSU where created in this work.

Figure 5.6: Router, UAV, CSU and landing pad
Router, UAV, CSU and landing pad. Total setup for precise, autonomous landing

Figure 5.7: Closeup of the 3DR solo quadcopter

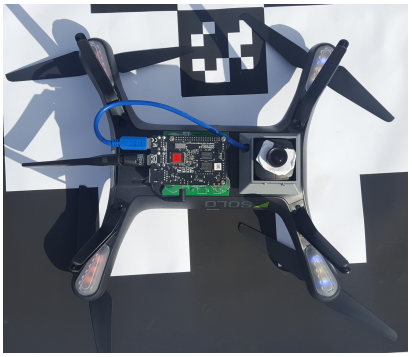


Figure 5.8: SBC, wireless adapter and camera mounted on the UAV

Figure 5.9: UAV landing on the FFI ground vehicle Olav

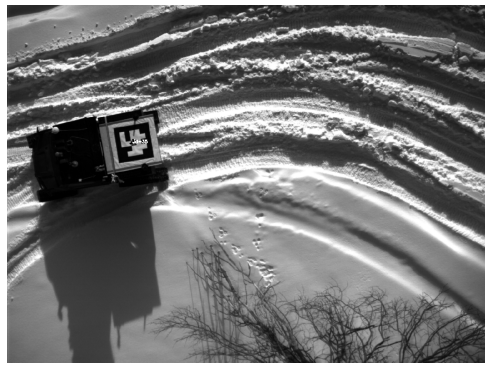


Figure 5.10: Olav detected from UAV camera at 12m

5.2.3.1 Custom Sensor Unit

The CSU is built by a flight controller, GNSS sensor, SBC, wireless adapter, battery and a 3D printed enclosure. Figure 5.13 illustrates the included components in details. Detailed information of the components can be studied in table 5.5. The flight controller is running PX4 flight controller software communicating with the SBC running Ubuntu 16.04 and ROS.

Device	Brand	Type nr
GNSS sensor	mRo	GPS u-Blox Neo-M8N
Flight Controller	mRo	Pixhawk 2.4.6
Single Board Computer	Hardkernel	ODROID-C2
Battery	Zippy Flightmax	3000mAh, 20C, 11.1v
Wireless Adapter	Edimax	ew-7811uac

Table 5.5: List of Devices in Custom Sensor Unit

Figure 5.11: UAV landing on the FFI surface vehicle Odin

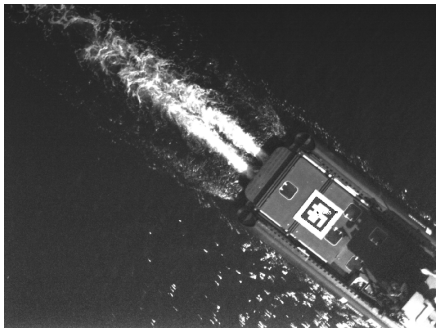


Figure 5.12: Odin detected from UAV camera at 15m

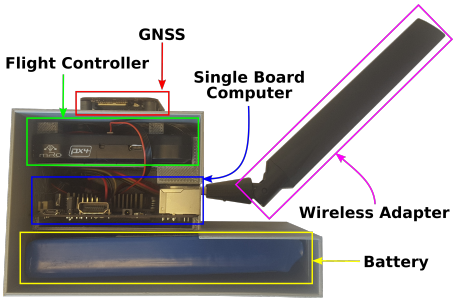


Figure 5.13: Overview of the components included in the SBC

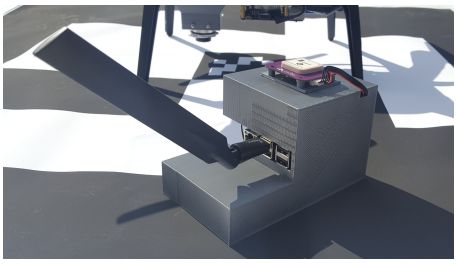


Figure 5.14: Fully assembled SBC

5.3 Simulation Setup

A 2D simulation environment were developed to compare the two guidance methods developed in section 4.2. The simulation environment were developed using Matlab and includes simplified dynamic models of Olav and the UAV. Moreover, Optimal and Parallel Navigation Guidance methods and a graphical illustration to present the results are implemented in the same environment.

The ground vehicle is modeled as the nonholonomic kinematic car based on rolling

without slippering constraints given in Spong et al. (2006) as

$$\dot{\boldsymbol{q}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \phi + \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \frac{1}{d} \tan \phi \\ 0 \end{bmatrix} \boldsymbol{v}^n_{l/n} \tag{5.2}$$

with the state vector $\boldsymbol{q}$

$$\boldsymbol{q} = \begin{bmatrix} p^n_{l/n,x} \\ p^n_{l/n,y} \\ \theta \\ \phi \end{bmatrix} \tag{5.3}$$

θ is the vehicle heading relative to the n frame, ϕ is the vehicle steering angle, $\boldsymbol{p}^n_{l/n} \in \mathbb{R}^2$is the position of the landing pad relative to the n frame, $\boldsymbol{v}^n_{l/n} \in \mathbb{R}^2$ is the landing pad velocity relative ti n and d is the length between the back and front wheels of the ground vehicle.

The UAV is modeled using a simple mass force model combined with a proportional velocity controller. The mass force model ca be given as

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{v}^n_{u/n} \\ \frac{\boldsymbol{f}^n_u}{m} \end{bmatrix} \qquad \boldsymbol{q} = \begin{bmatrix} \boldsymbol{p}^n_{u/n} \\ \boldsymbol{v}^n_{u/n} \end{bmatrix} \tag{5.4}$$

where $\boldsymbol{p}^n_{u/n} \in \mathbb{R}^2$ and $\boldsymbol{v}^n_{u/n} \in \mathbb{R}^2$ represents the UAV position and velocity relative to and given in the n frame. The vector $\boldsymbol{f}^n_u \in \mathbb{R}^2$ represent the force acting on the UAV given in n frame and the constant m is the total mass of the UAV. Furthermore, the proportional velocity controller controlling the UAV velocity with the force acting on the UAV $\boldsymbol{f}^n_u$ is implemented by.

$$\boldsymbol{f}^n_u = K_p(\boldsymbol{v}^n_{u/n} - \boldsymbol{v}^n_{u/n,m}) \tag{5.5}$$

The Optimal Guidance method is implemented by solving the quadratic objective function 4.14 subject to its constraints 4.19-4.24 containing the desired dynamics given

by the matrices 4.28 using the quadprog function included in the Matlab Optimization Toolbox (MATLAB; 2017). While the parallel navigation guidance method is implemented by using the controller given in equation 4.12.

Chapter 6

Results

6.1 Guidance Simulations

This section presents the results from the simulations of the Parallel Navigation- and Model Optimal Guidance methods discussed in section 4.2. The weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ used in the simulations presented are given in 6.1, the constraints restricting the states are given in 6.2 and the constraints limiting the input are given in 6.3.

$$\boldsymbol{Q} = \begin{bmatrix} 1.3 & 0 & 0 & 0 \\ 0 & 1.3 & 0 & 0 \\ 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 500 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

$$\boldsymbol{x}^{low} = \begin{bmatrix} -\infty \\ -\infty \\ -\infty \\ -\infty \end{bmatrix} \qquad \boldsymbol{x}^{high} = \begin{bmatrix} \infty \\ \infty \\ \infty \\ \infty \end{bmatrix} \tag{6.2}$$

$$\boldsymbol{u}^{low} = \begin{bmatrix} u^n_{d/u,x,k} \\ u^n_{d/u,y,k} \\ -15 \\ -15 \end{bmatrix} \qquad \boldsymbol{u}^{high} = \begin{bmatrix} u^n_{d/u,x,k} \\ u^n_{d/u,y,k} \\ 15 \\ 15 \end{bmatrix} \qquad \Delta\boldsymbol{u} = \begin{bmatrix} \infty \\ \infty \\ 2.5 \\ 2.5 \end{bmatrix} \tag{6.3}$$

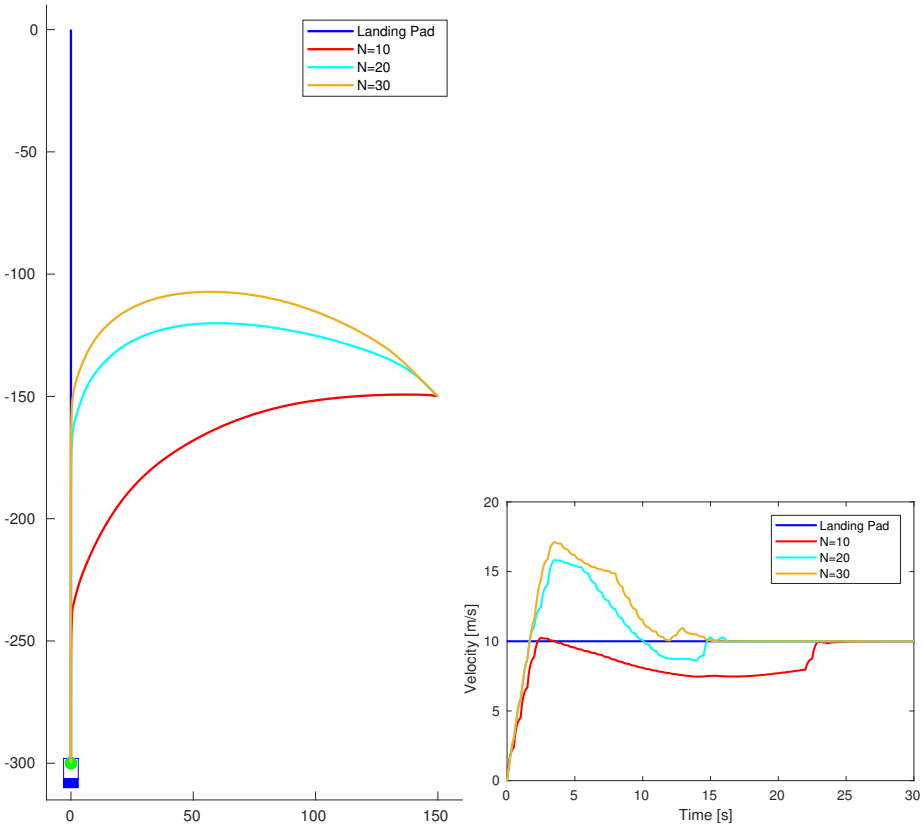
where the states inequality constraints 6.2 gives no restrictions in the flight area. The control constraints in 6.3 restricts the UAV velocity to $\pm 15 m/s$ and forces the optimization algorithm to predict constant velocity equal to the measured velocity $u^n_{d/u,k}$ on the control objective.

Figure 6.1 illustrates how the behavior of the Optimal Guidance Controller changes by using different prediction lengths N. Where the blue line in sub figure 6.1a represents the path driven by the landing pad at constant velocity and the remaining lines represents the path driven by the UAV using Optimal Guidance controller with different prediction lengths. Sub figure 6.1b presents the velocity during the simulation. Further on in the guidance simulations, the prediction length is set to $N = 10$. The parameters used in the Parallel Navigation Guidance controller are given in 6.4.

$$\Delta = 16 \qquad U_{c,max} = 10 \tag{6.4}$$

6.1.1 Constant Velocity

The simulation results given in figure 6.2 presents the behavior of the Optimal- and the Parallel Navigation Guidance tracking a target at constant velocity. There are two simulation cases presented in the figure, one where the UAV's initial positions are set to $[150, -150]$, and one where the initial positions are set to $[150, -50]$. Sub figure 6.2a, illustrates the paths the UAV's and the landing pad have driven, sub figure 6.2b and 6.2c illustrates the corresponding velocities using the initial condition $[150, -150]$ and $[150, -50]$ respectively.



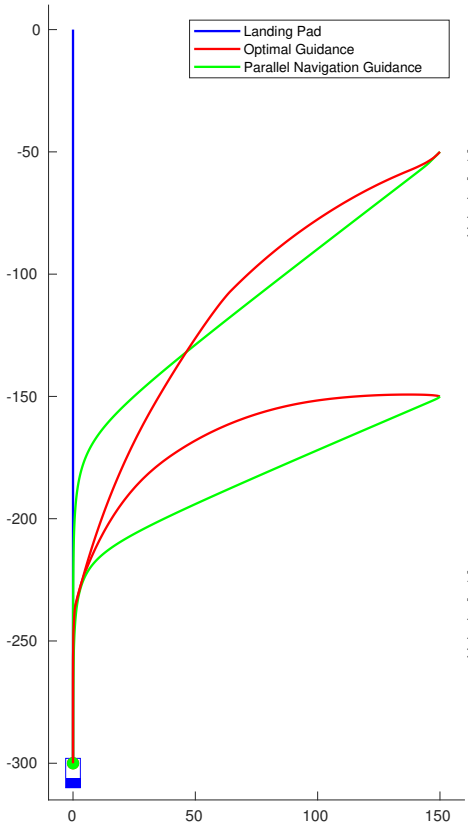
(a) Position track from the LP and UAV's

(b) Velocity from LP and UAV's

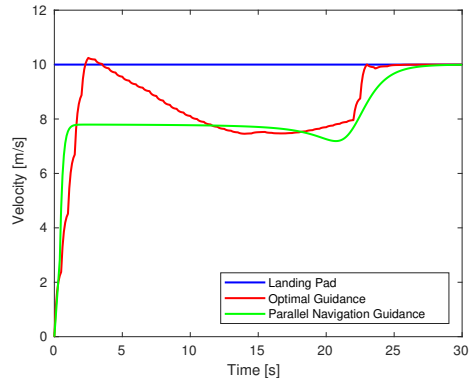
Figure 6.1: Optimal Guidance with different prediction lengths
Optimal Guidance with different prediction lengths. Constant velocity LP

6.1.2 Accelerating

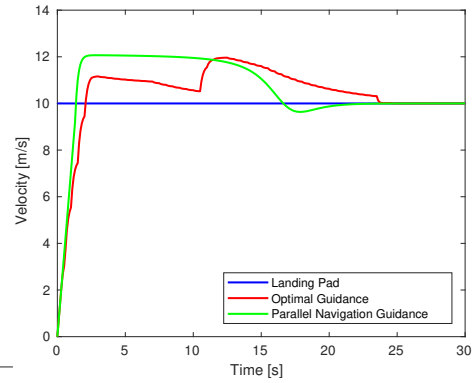
In this simulation, the landing pad is acceleration from 0 to $10m/s$ with an acceleration of $1m/s^2$ and then continues with a constant velocity. Figure 6.3 presents the paths and velocities generated while tacking the landing pad using Optimal- and Parallel Navigation Guidance.



(a) Position track from the LP and UAV's



(b) Initial position [150,-150]

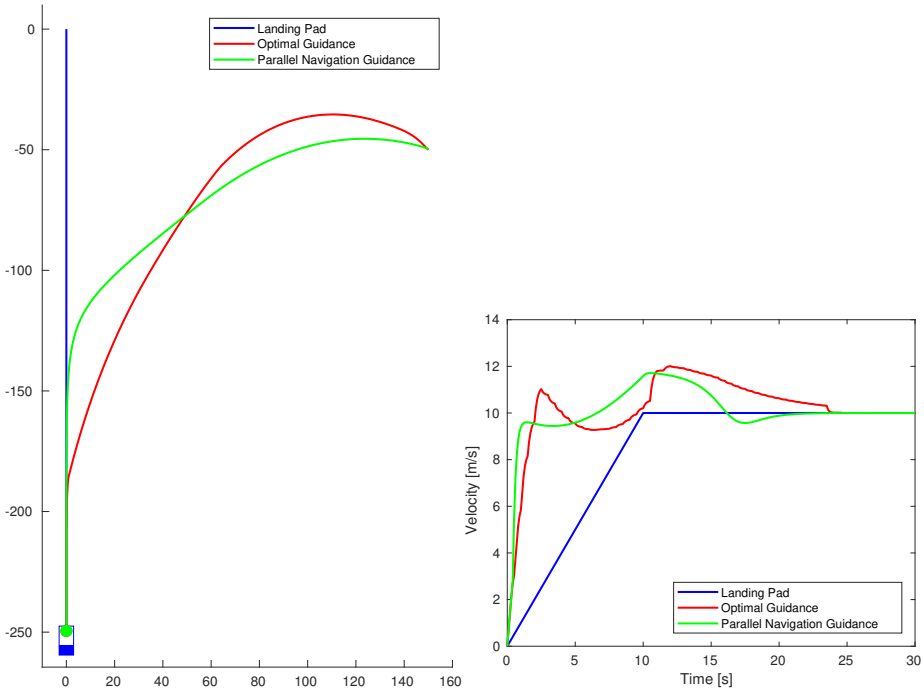


(c) Initial position [150,-50]

Figure 6.2: Optimal and Parallel navigation Guidance with a constant velocity target

6.1.3 Random Driving

In the simulations presented in figure 6.4, random noise is added to the steering angle of the landing pad, while the speed is kept constant. Sub figure 6.4b represents the velocity of the random driving illustrated in 6.4a. Similarly, sub figure 6.4d represents the velocity from the random driving in 6.4c.



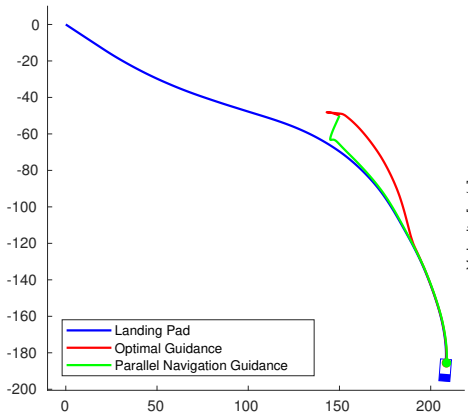
(a) Position track from the LP and UAV's

(b) Velocity from LP and UAV's

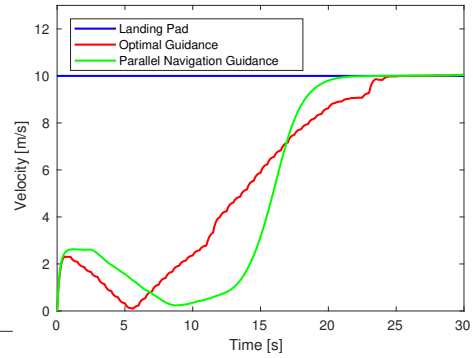
Figure 6.3: Optimal and Parallel navigation Guidance with a accelerating target

6.1.4 Summary

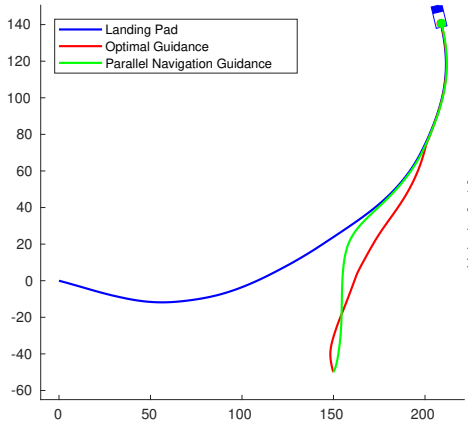
Overall, both the guidance methods compared in this result are able to track the desired tracking point in all the scenarios simulated. The rugged velocity measurements from the Optimal Guidance methods may be possible to smooth out by tuning the controller. However, that is one of the major disadvantage using the Optimal Guidance method. The tuning process is time consuming due to the high number of parameters to be tuned. Moreover, it is far more complex to implement and has a higher computational load compared to the Parallel Navigation Guidance method. The benefit of using the Optimal Guidance method, is the opportunity to set constraints to the states and control



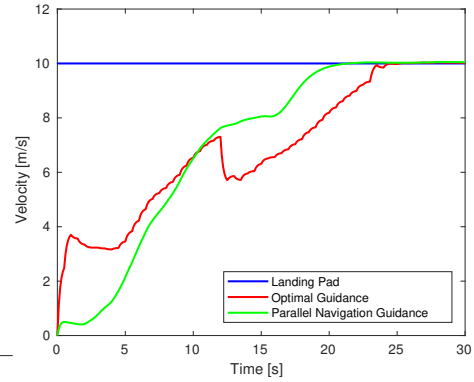
(a) Position track from random steering 1



(b) Velocity from random steering 1



(c) Position track from random steering 2



(d) Velocity from random steering 2

Figure 6.4: Optimal and Parallel navigation Guidance with random steering target

outputs. Such constraints can be restrictions in areas to fly, velocity and acceleration. Due to the simplicity of the Parallel Navigation Guidance method in addition to the proven stability given in section 4.2.1, this is the method implemented in the physical system.

6.2 State estimator tuning

In this section, the results from the implementation and tuning of the sate estimator are presented. The Kalman filter for position, linear velocity and bias state estimation introduced in section 3.3.2 is implemented in the ROS node "node_navigation". All measurements are collected in ROS bags (data logs) during flight and subsequently used to tune parameters in the state estimator.

6.2.1 Static Landing Pad

When tuning the filter for a static landing pad, the prior knowledge of a static landing pad is used to improve the estimates as discussed in 3.3.4. The filter is automatically initialized by adapting the knowledge of using the Landing Pad as takeoff position. The multi marker ArUco tag system is used as fiducial marker and the custom sensor station is used as Landing Pad sensors measurement unit. The filter parameters used in this implementation are given in equation 6.5 to 6.7

Figure 6.5: UAV landing on a static Landing Pad

$$\boldsymbol{Q}_{diag} = \begin{bmatrix} 4.1 & 4.1 & 4.1 & 0 & 0 & 0 & 0.1 & 0.1 & 10 \end{bmatrix} \cdot 10^{-3} \tag{6.5}$$

$$\boldsymbol{R}_{diag} = \begin{bmatrix} 2 & 2 & 11 & 0.06 & 0.06 & 0.025 & 10000 & 10000 & 10000 \end{bmatrix} \tag{6.6}$$

$$\boldsymbol{P}_{0,diag} = \begin{bmatrix} 10 & 10 & 10 & 350 & 350 & 380 & 10 & 10 & 10 \end{bmatrix} \cdot 10^{-3} \tag{6.7}$$

where $\boldsymbol{Q}_{diag}$, $\boldsymbol{R}_{diag}$ and $\boldsymbol{P}_{0,diag}$ are the diagonal elements in the corresponding 9×9 diagonal matrices.

Figures 6.6a, 6.6b and 6.6c present the results from the implemented Kalman

filter state estimator. Where sub figure 6.6a presents the measurements from the ArUco detection, the measurements from the GNSS position sensors, estimated relative position vector $\boldsymbol{p}^n_{l/u}$ and its corresponding 2σ bound. All given in the North, East and Down components. Moreover, the estimated landing pad velocity $\boldsymbol{v}^n_{l/n}$ from the same test are given in sub figure 6.6b. Position bias $\boldsymbol{\beta}^n_{l/n}$ and its corresponding 2σ bound are given in 6.6c. The 2σ bound represents the 95% confidence interval.

Figure 6.7 gives a closed up view of figure 6.6a. Moreover, from figure 6.7 and 6.6c it can be seen how the state estimate corrects, the covariance of the estimate narrows and the estimated bias adjusts as the ArUco tag measurements starts to arrive at $t = 29$.

The figure 6.6a and 6.7 illustrates the importance of supplementing the ArUco measurements to the state estimator, in addition to the GNSS measurements. Without the presence of ArUco measurements, the position estimate gets higher covariance and navigation biases cannot be updated.

6.2.2 Landing Pad in Motion

This section presents the result from the implemented and tuned state estimator in 3.3.1. A multi marker ArUco tag system is placed on the deck of the FFI surface vehicle Odin. The already implemented state-of-the-art navigation estimate of the surface vehicle is broadcasted to the UAV. The filter parameters used during this tests are given in equation 6.8 to 6.10

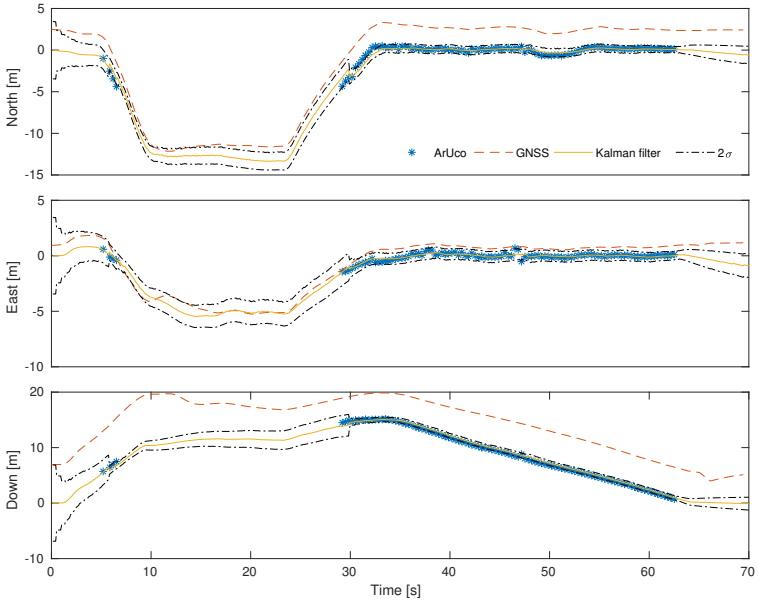
$$\boldsymbol{Q}_{diag} = \begin{bmatrix} 4.1 & 4.1 & 4.1 & 10 & 10 & 10 & 0.1 & 0.1 & 10 \end{bmatrix} \cdot 10^{-3} \tag{6.8}$$

$$\boldsymbol{R}_{diag} = \begin{bmatrix} 2 & 2 & 11 & 0.06 & 0.06 & 0.025 & 0.0024 & 0.0024 & 0.007 \end{bmatrix} \tag{6.9}$$

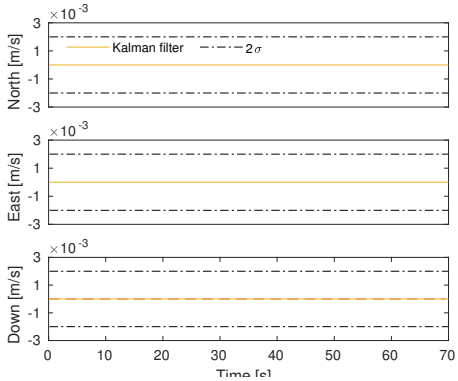
$$\boldsymbol{P}_{0,diag} = \begin{bmatrix} 140 & 140 & 410 & 12.4 & 12.4 & 12.4 & 28 & 28 & 600 \end{bmatrix} \cdot 10^{-3} \tag{6.10}$$

where $\boldsymbol{Q}_{diag}$, $\boldsymbol{R}_{diag}$ and $\boldsymbol{P}_{0,diag}$ are the diagonal elements in the analogous 9×9 diagonal matrices.

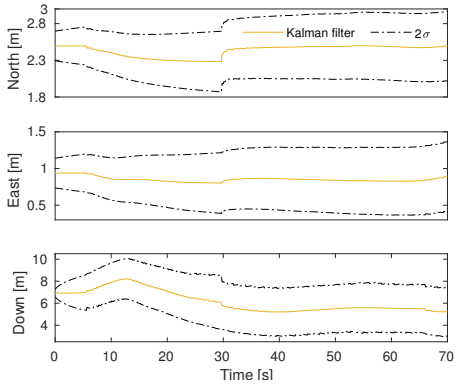
As illustrated in figure 6.8, the test results was conducted with the UAV first hover at a stationary point while the landing pad accelerates up to $1.8 m/s$. The UAV then starts to fly towards the landing pad and matches its velocity before lovering the



(a) State estimates, GNSS- and ArUco measurements of relative position between UAV and LP $\boldsymbol{p}_{l/u}^{n}$



(b) State estimates and measurements of the LP velocity $\boldsymbol{v}_{l/n}^{n}$



(c) State estimates of the position bias $\boldsymbol{\beta}_{l/n}^{n}$

Figure 6.6: Results from the navigation filter tuned for a static landing pad

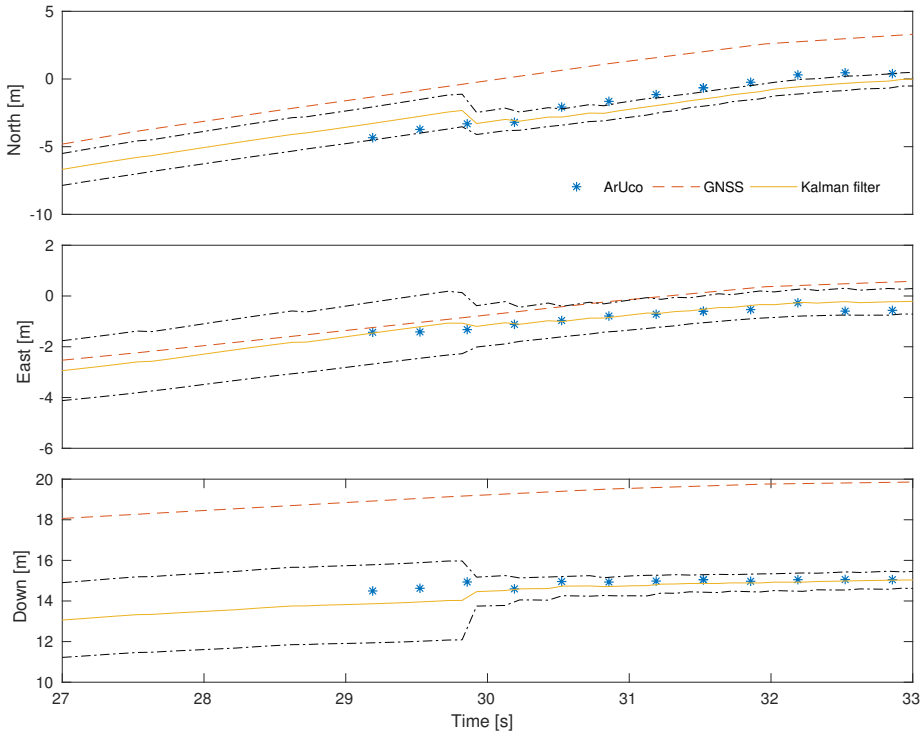


Figure 6.7: How covariance estimate narrows
Covariance estimate narrows as the fiducial marker gets within camera sight

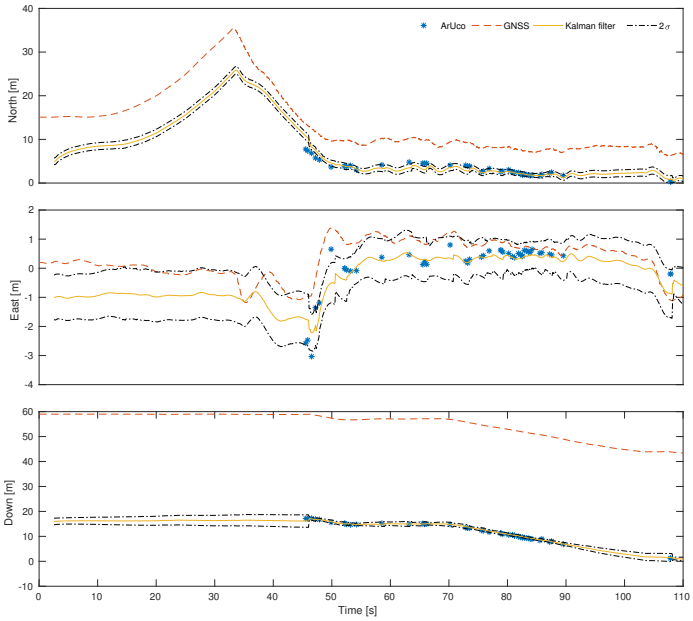
altitude and lands on the landing pad. Figure 6.9 highlights a selection of the ArUco measurements, position estimate and the estimated covariance from figure 6.8 where the ArUco tag were detectable. As figure 6.8 and 6.9 illustrates, the UAV was oscillating back and forward during the flight. The oscillations was casing motion blur on the pictures used for ArUco tracking, making it challenging to detect fiducial markers. From figure 6.9, it can be seen how the lack of tag detections increases the covariance compared with the period without oscillations.

A closeup of the estimated and measured north component of the landing pad velocity $\boldsymbol{v}^n_{l/n,n}$ is given in figure 6.11. As the figure illustrates, there is a time delay of approximate 3 seconds between the changes in the measured velocity and the estimates velocity. This may be caused by the constant velocity model used in the Kalman filter equation derived in section 3.3. To be able to have a more accurate track of the landing pads during acceleration, the Kalman filter can be extended to a constant acceleration model.

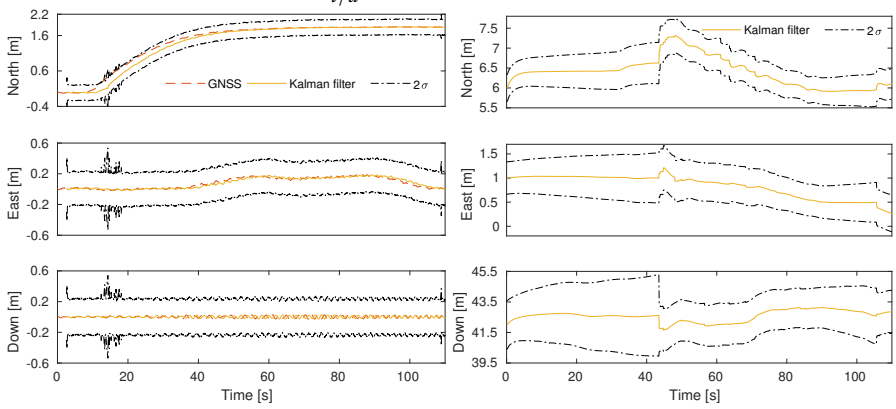
6.2.3 Summary

Since there is no precise reference measurements in the dataset, such as records from a motion capture lab or like, there is no straight forward method to validate the state estimates. However, from the figures in 6.6 and 6.8 it can be seen that the Kalman filter is able to return smooth and reasonable state estimates. Moreover, several dozens of tests have been conducted in varying weather conditions such as windy, windless, snow, bright sun and cloudy. These tests have proven the state estimator to be robust and generate reliable and accurate navigation estimates.

Moreover, on static landing pads, the sensor bias estimated by the state estimator does include the position offset between the fiducial marker and the GNSS receiver. Hence, the local navigation unit can be placed at an arbitrary place near by the landing pad and the position offset will automatically be included in the bias.



(a) State estimates, GNSS- and ArUco measurements of relative position between UAV and LP $\boldsymbol{p}_{l/u}^{n}$



(b) State estimates and measurements of the LP velocity $\boldsymbol{v}_{l/n}^{n}$

(c) Covariance estimate narrows as the fiducial marker gets within camera sight

Figure 6.8: Results from the navigation filter tuned for a landing pad at speed

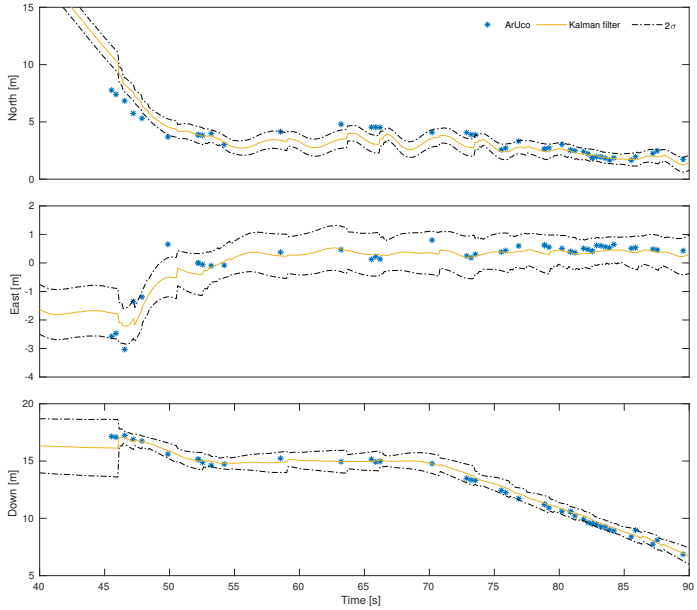


Figure 6.9: Closeup of the ArUco measurements
Closeup of the ArUco measurements, estimated position vector $\boldsymbol{p}_{l/u}^{n}$ and 2σ bound during camera fix

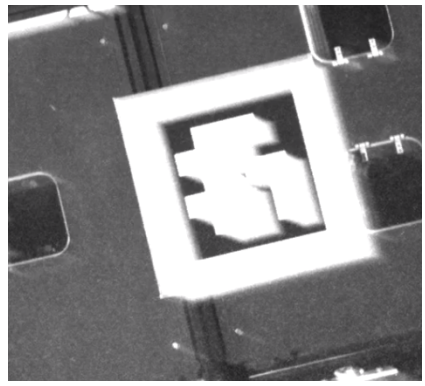


Figure 6.10: Image from the onboard camera affected by motion blur
Image from the onboard camera affected by motion blur due to high angular velocity of the UAV

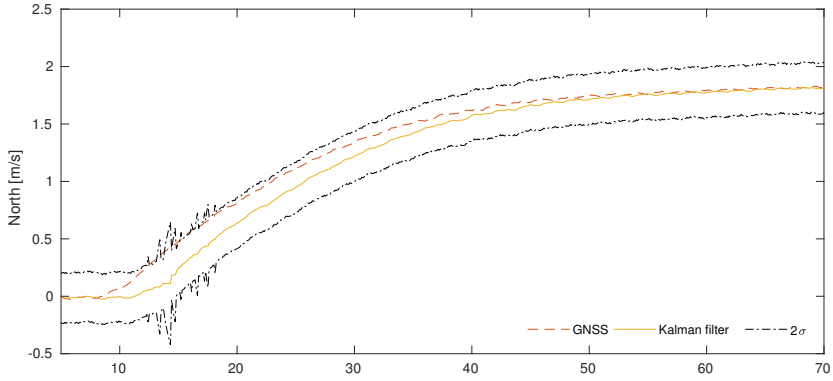


Figure 6.11: Delay between the measured and estimated LP velocity
Delay between the measured and estimated north component of the LP velocity

6.3 Controller Tuning

The software modules node_navigation, node_aruco and node_controller described in section 5.1 are implemented and running in real time locally on the UAV. Tests are conducted by manual takeoff and flight to a starting point at a distance from the

landing pad. The *do land* command is then triggered from the work station and the UAV, running the controller defined in chapter 4 performs autonomous landing. The parameters listed in 6.1 are used in the implemented state machine.

Parameter	Description	Value
$U_{c,max}$	Maximum UAV approach speed	$3m/s$
Δ_1	controller gain	3.0
Δ_2	controller gain after gain adjust	2.0
h	Hover height	$15m$
r_h	Radius of the hover sphere	$0.5m$
h_g	Gain adjust altitude	$8.0m$
h_f	Final stage altitude	$4.0m$
v_d	Main decent velocity	$0.40m/s$
v_f	Final stage decent velocity	$0.30m/s$
c_h	Minimum covariance in hover	100
r_l	Radius of the landing cylinder	$0.20m$
h_l	Height of the landing cylinder	$0.50m$

Table 6.1: Parameters used in the state machine

Figure 6.12 presents a plot of the relative position estimate between the UAV and landing pad $\boldsymbol{p}^n_{l/u}$ calculated by the state estimator during the test. The states from the state machine are included in the figure as vertical lines marked with the state name. Moreover, the *Adjust* and *Final* are abbreviations for *Gain Adjust* and *Final Stage*.

As indicated in figure 6.3, the first 19 seconds of the flight is carried out manually. The state machine then enters the *intercept* state and the UAV starts to fly towards the hovering point. From the *intercept* phase in figure 6.12, there is a "jump" in the position estimate. This is due to filter corrections in the appearance of AruCo measurements. The state machine goes straight from *Intercept* state to *Lower*, not including the *Hover* state. This is a result of the high value set as parameter for the minimum covariance in hover c_h. The oscillations in the north and east position caused by the wind affecting the UAV. During the Gain Adjust phase, the north and east position stabilizes, making the UAV stable trough the *Final* state until it finally triggers the *Land* command.

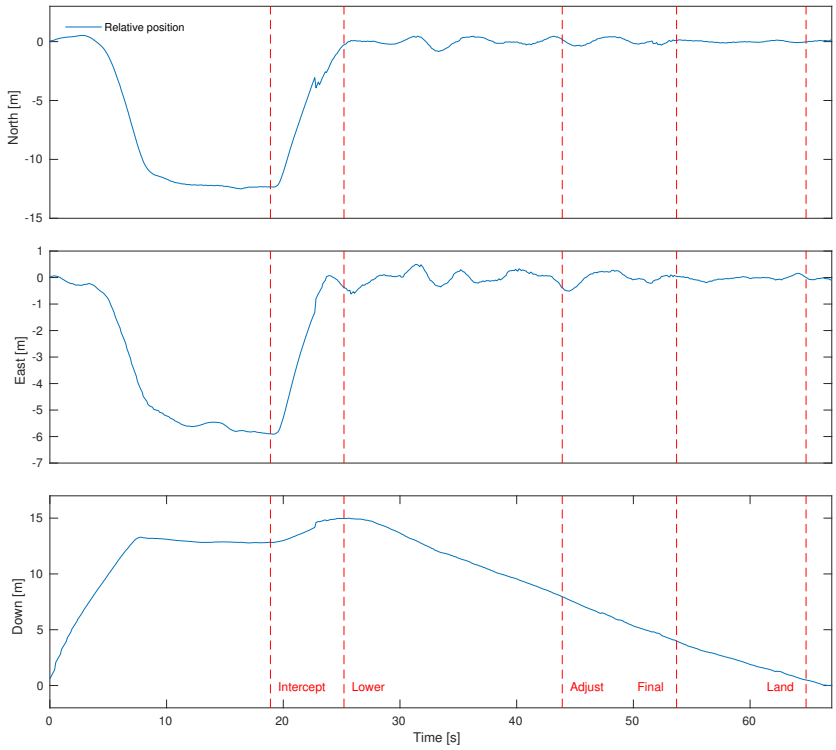


Figure 6.12: Relative position $\boldsymbol{p}_{l/u}^{n}$ during autonomous landing
Relative position $\boldsymbol{p}_{l/u}^{n}$ during autonomous landing. State machine states indicated by vertical lines

6.3.1 Summary

Several dozens of physical real-time test on a static landing pad have been conducted using the implemented software modules node_navigation, node_aruco and node_controller. The autonomous system has given exceptional results, by performing precise and stable landings with high repeatability. A video of from one of the tests conducted can be seen at `https://youtu.be/H2HTWxUuOW8`

Chapter 7

Conclusion and future work

The main objective of this thesis is to conduct autonomous and precise landing on a static landing pad, or at a landing pad in motion.

To measure the pose of the landing pad relative to the UAV, a camera based measurement system for detecting fiducial markers is implemented as an image processing module. Due to limited measuring range in traditional fiducial markers, a multi fiducial marker method is developed in this thesis, extending the measurement range. The imaging processing module has been extensively tested under various scenarios, such as winter conditions, bright summer sunlight at sea and indoor, all given reliable and accurate results.

A state estimator based on a Kalman filter has been derived and implemented in this thesis. The filter has been parameterized and tuned, both for a static landing pad and for a landing pad in motion. The state estimator is able to fuse measurements from the UAV, the landing pad and the camera based measurement system, returning estimates on position, velocity and sensor biases. Never the less, the state estimates have shown to manage sensor fusion of multiple asynchronous measurements in a robust way.

The two guidance methods Parallel Navigation Guidance and Optimal Guidance are discussed and compared using simulations. The Parallel Navigation method is

by far the easiest to implement, have low computational cost and are proven to be UGAS. In the Optimal Guidance method it is possible to add constraints, such as restrictions in the flight area or control output. However, this guidance method has a high computational cost compared to the Parallel Guidance method and it is harder to implement. Hence, the Parallel Navigation method was the method of choice in the implementation on the UAV. Controller logic has been developed to set the desired UAV trajectory, velocities and controller gains.

The image processing module, the state estimator, the Parallel Navigation method and the controller logic are all implemented and running in real time on the UAV. Results from physical tests indicates that the combination of the state estimator and the Parallel Navigation Guidance method ensures robust and accurate autonomous landing on a static target. Additionally, results from the tuned state estimator and simulations indicated that the system is able to carry out autonomous landing on landing pads in motion. Landing has been conducted both on ground and a surface vehicles containing world class navigation system, and on a homemade landing pad containing a low cost navigation system. FFI will use the state estimator, guidance method and the controller logic developed in this project on their attempt to create fully autonomous drone swarms.

Further work

To achieve autonomous landing of a multirotor UAV on a platform in motion, several topics requires further investigation.

- Large-scale testing and tuning of the state estimator, Parallel Navigation method and controller logic on a landing pad in motion. Kalman filter may be tuned by running a full scale tests with a high accuracy motion capture system measuring the pose of the UAV and the landing pad.
- Extending the Kalman filter given in 3.3.2 Position, linear velocity and bias state estimation, by include relative heading between the UAV and landing pad in the filter equations.

- Investigate the multirotor UAV dynamics near touchdown.
- Add additional communication methods between the UAV and the landing pad. Enables the landing pad to broadcast its position to the UAV if the UAV is out of reach on the 5GHz network.
- Derive and implement accurate covariance models for all measurements.
- If the landing pad is autonomous, information from the path planner on the landing pad can be included to improve the UAV controller.
- Add condition monitoring and fault detection system to the UAV controller, detecting and act on unexpected behavior.

References

AG, D. (2017). The mercedes-benz vision van. [Online; accessed 16-December-2017].
URL: https://www.daimler.com/innovation/specials/vision-van/en/

Araar, O., Aouf, N. and Vitanov, I. (2017). Vision based autonomous landing of multirotor uav on moving platform, *Journal of Intelligent & Robotic Systems* **85**(2): 369–384.

Beard, R. W. and McLain, T. W. (2012). *Small unmanned aircraft: Theory and practice*, Princeton university press.

Belleter, D. J. (2016). Control of underactuated marine vehicles in the presence of environmental disturbances.

Borowczyk, A., Nguyen, D.-T., Nguyen, A. P.-V., Nguyen, D. Q., Saussié, D. and Ny, J. L. (2016). Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle, *arXiv preprint arXiv:1611.07329* .

Breivik, M. (2010). Topics in guided motion control of marine vehicles.

Breivik, M. and Fossen, T. I. (2007). Applying missile guidance concepts to motion control of marine craft, *IFAC Proceedings Volumes* **40**(17): 349 – 354. 7th IFAC Conference on Control Applications in Marine Systems.
URL: *http://www.sciencedirect.com/science/article/pii/S1474667015321200*

Brown, R. G. and Hwang, P. Y. (1997). *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*.

Cai, G., Chen, B. M. and Lee, T. H. (2011). *Unmanned rotorcraft systems*, Springer Science & Business Media.

Egeland, O. and Gravdahl, J. T. (2002). *Modeling and simulation for automatic control*, Vol. 76, Marine Cybernetics Trondheim, Norway.

Foss, B. and Heirung, T. A. N. (2013). Merging optimization and control, *Lecture Notes* .

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

Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F. and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion, *Pattern Recognition* **47**(6): 2280 – 2292.
URL: *http://www.sciencedirect.com/science/article/pii/S0031320314000235*

Gibbs, S. and agencies (2016). First successful ship-to-shore drone delivery takes place in new jersey. [Online; accessed 16-December-2017].
URL: `https://www.theguardian.com/technology/2016/jun/24/first-successful-ship-to-shore-drone-delivery-new-jersey`

Groves, P. D. (2013). *Principles of GNSS, inertial, and multisensor integrated navigation systems*, Artech house.

Huang, R., Tan, P. and Chen, B. M. (2015). Monocular vision-based autonomous navigation system on a toy quadcopter in unknown environments, *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, IEEE, pp. 1260–1269.

Itseez (2018). Open source computer vision library, `https://github.com/itseez/opencv`.

Khalil, H. K. (2015). Nonlinear systems.

Line, V. (2017). Autonomous takeoff and landing of a multirotor uav on a platform in motion.

Lozano, R. (2013). *Unmanned aerial vehicles: Embedded control*, John Wiley & Sons.

Martin, P. and Salaün, E. (2010). The true role of accelerometer feedback in quadrotor control, *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, pp. 1623–1629.

MATLAB (2017). Matlab optimization toolbox version 8.0. The MathWorks, Natick, MA, USA.

Nicolai, L. M. and Carichner, G. (2001). *Fundamentals of aircraft and airship design*, American Institute of Aeronautics and Astronautics,.

Open Source Robotics Foundation (2018a). About ros.
URL: *http://www.ros.org/about-ros/*

Open Source Robotics Foundation (2018b). Mavros.
URL: *http://wiki.ros.org/mavros*

Roumeliotis, S. I., Sukhatme, G. S. and Bekey, G. A. (1999). Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization, *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, Vol. 2, IEEE, pp. 1656–1663.

Shneydor, N. (1998a). Chapter 4 - parallel navigation, *in* N. Shneydor (ed.), *Missile Guidance and Pursuit*, Woodhead Publishing, pp. 77 – 100.
URL: *https://www.sciencedirect.com/science/article/pii/B9781904275374500095*

Shneydor, N. (1998b). Introduction, *in* N. Shneydor (ed.), *Missile Guidance and Pursuit*, Woodhead Publishing, pp. xiii – xvi.
URL: *https://www.sciencedirect.com/science/article/pii/B9781904275374500058*

Spong, M. W., Hutchinson, S. and Vidyasagar, M. (2006). *Robot modeling and control*, Vol. 3, Wiley New York.

Tailanian, M., Paternain, S., Rosa, R. and Canetti, R. (2014). Design and implementation of sensor data fusion for an autonomous quadrotor, *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*, IEEE, pp. 1431–1436.

Tiemann, J., Schweikowski, F. and Wietfeld, C. (2015). Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments, *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, IEEE, pp. 1–7.

Tørdal, S. S. and Hovland, G. (2017). Relative vessel motion tracking using sensor fusion, aruco markers, and mru sensors.

Vik, B. (2009). Integrated satellite and inertial navigation systems, *Department of Engineering Cybernetics, NTNU* .

Zou, J.-T., Wang, C.-Y. and Wang, Y. M. (2016). The development of indoor positioning aerial robot based on motion capture system, *Advanced Materials for Science and Engineering (ICAMSE), International Conference on*, IEEE, pp. 380–383.