Object Detection and Tracking Based on Optical Flow in Unmanned Aerial Vehicles

Håkon Hagen Helgesen

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Thor Inge Fossen, ITK
Co-supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Formulation

Thesis Description:

A UAV with daylight camera can be used to observe interesting objects, such as ships, boats, persons, vehicles, marine mammals, ice and environmental spills. The purpose of the thesis is to investigate the effectiveness of optical flow in order to 1) detect moving objects (moving object segmentation), 2) track the detected objects, and 3) estimate relative velocity between the UAV and the background (sea or terrain).

The following items should be considered:

1. Literature survey, considering general methods for detection and tracking of objects and features from a moving platform with ego-motion using optical flow, and related applications such as automotive (pedestrians and other vehicles), aerial ocean surveillance, UAV velocity estimation, etc.
2. Develop algorithms for computing optical flow assuming no moving objects. Use the algorithm to estimate the relative velocity between the UAV and the background. Use the calculated velocity in a nonlinear observer with IMU and GNSS in order to estimate the position, velocity, attitude and gyro bias of the UAV.
3. Develop an algorithm for computing optical flow and segmentation of moving objects assuming one moving object in the image frame.
4. Based on 2 and 3, develop a tracking system for tracking an arbitrary number of objects (from zero to many) from a sequence of images.
5. Use the position of the UAV to compute georeferenced position and velocity over ground estimates for tracked objects.
6. Test the algorithms through computer simulations.

Start date: 2015-01-12
Due date: 2015-06-08
Thesis performed at: Department of Engineering Cybernetics, NTNU
Supervisor: Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU

Abstract

In recent years the Unmanned Aerial Vehicle (UAV) community has discovered the enormous amount of information that can be extracted from a video camera. It can be used for collision avoidance, navigation, velocity estimation, terrain mapping, object detection and object tracking in addition to many other applications. This thesis looks into several different ways a video camera in the payload of a fixed-wing UAV can be utilized. The first part investigates a way to calculate the body-fixed velocity of the UAV with the images captured by the camera. It can be achieved with optical flow and measurements of the roll angle, pitch angle and the altitude of the UAV. The calculated body-fixed velocity can be used as a measurement in the navigation system. This thesis looks into a navigation system based on a nonlinear observer that estimates attitude, position, velocity and gyro bias. In order to estimate these states the nonlinear observer utilizes measurements from an Inertial Measurement Unit (IMU), a Global Navigation Satellite System (GNSS) receiver and a video camera.

The second part looks into moving object detection and tracking. An algorithm for detection of moving objects has been developed. It utilizes the navigation states of the UAV and optical flow in order to extract the moving objects from the images. Furthermore a tracking system based on the moving object detection algorithm has been proposed. The tracking system consists of the moving object detection algorithm, a classifier that describes each object and a discrete Kalman filter that estimates the motion of the object. The objects are tracked in the image plane and the estimates are transformed to the North-East-Down (NED) coordinate system.

A fixed-wing UAV experiment has been carried out to gather images captured from an airborne UAV and collect data from an IMU and a GPS receiver. The navigation system has been evaluated offline by computer simulations based on the data collected at the experiment. The tracking system and the moving object detection algorithm have also been evaluated in computer simulations. Promising and accurate results were shown for both the navigation system and the proposed tracking system.

Keywords: Unmanned Aerial Vehicle, Vision Based Navigation, Vision-Aided Nonlinear Observer, Discrete Kalman Filter, Navigation System, Sensor Fusion, Computer Vision, Optical Flow, Object Detection, Object Tracking

Sammendrag

De siste ti årene har bruk av kamera i ubemannede fly vært et sentralt forskningstema. Ved å montere et kamera i et ubemannet fly kan man benytte det for å unngå kollisjoner, navigere, estimere hastighet, lage terrengprofiler, detektere objekter og målfølge ojekter blant annet. Denne avhandlingen ser på forskjellige anvendelser hvor et kamera montert i nyttelasten til et ubemmanet fly er nyttig. Den første delen undersøker og presenterer en måte bildene fra kameraet kan brukes til å beregne hastigheten til det ubemannende flyet. Dette kan realiseres med optisk flyt, måling av orienteringen til flyet og en høydemåling. Den beregnede hastigheten kan brukes i navigasjonssystemet til det ubemannede flyet. Denne avhandlingen ser nærmere på et navigasjonsssytem som er realisert med en ulineær observer som estimerer orientering, posisjon, hastighet og gyroskopbias. Den ulineære observeren er avhengig av målinger fra en IMU, et globalt posisjonerings system (GPS) og kameraet.

Den andre delen undersøker hvordan objekter i bevegelse kan detekteres og følges ved hjelp av kameraet. En algoritme som finner objekter i bevegelse blir foreslått. Denne algoritmen benytter estimatene fra navigasjonssystemet og optisk flyt for å finne objekter i bevegelse i bildene. I tillegg er et målfølgingssystem som benytter denne algoritmen foreslått. Dette systemet består av algoritmen som finner objekter i bevegelse, en beskrivelse av alle objekter som detekteres slik at de kan skilles fra hverandre og et diskret Kalman filter som estimerer bevegelsen til objektene. Objektene målfølges i bildeplanet og estimatene transformeres til Nord-Øst-Ned (NED) koordinatsystemet.

En testflyvning med et ubemannet fly har blitt gjennomført for å samle bilder og data fra IMU og GPS. Navigasjonssystemet har blitt testet og evaluert gjennom simuleringer. Målfølgingssystemet og algoritmen for å finne objekter i bevegelse har også blitt testet og evaluert gjennom simuleringer. Simuleringene viste lovende og nøyaktige resultater for både navigasjonssystemet og målfølgingssystemet.

Nøkkelord: Ubemannet Fly, Kamerabasert Navigasjon, Kamera-Assistert Ulineær Observer, Diskret Kalman Filter, Navigasjonssystem, Sensorfusjon, Bildebehandling, Optisk Flyt, Objekt Deteksjon, Målfølging

Preface

The work described in this thesis is carried out in the Department of Engineering Cybernetics, at the Norwegian University of Science and Technology, the spring of 2015. It is submitted in partial fulfilment of the requirements for the degree of MSc. at the Norwegian University of Science and Technology. In addition to this thesis a part of the work has been to write two papers in cooperation with fellow MSc. student Jesper Hosen, PhD. candidate Lorenzo Fusini, Professor Thor I. Fossen and Professor Tor A. Johansen. The first paper has been submitted and accepted for the International Conference on Unmanned Systems 2015 (ICUAS'15). The second paper has been submitted to the 2016 American Institute of Aeronautics and Astronautics Science and Technology Forum and Exposition (AIAA SciTech 2016) and acceptance or rejection is expected late in August 2015.

I would like to thank Professor Thor Inge Fossen who trusted me to write this thesis and giving me useful feedback throughout this spring. I would also like to thank Professor Tor Arne Johansen who created the problem formulation and have used a lot of time on proofreading the papers. Furthermore I would like to thank PhD. candidate Lorenzo Fusini for the cooperation and supervision. I have worked closely with fellow MSc. student Jesper Hosen this year and he deserves a lot of credit for a great collaboration. He has given me a lot of inspirational ideas and useful feedback.

Finally, I would like to thank my parents, brother, sister and girlfriend Hilde for their support.

Trondheim, June 2015

Håkon Hagen Helgesen

Table of Contents

List of Tables

List of Figures

Abbreviations

6 DOF	= Six Degrees of Freedom
Abs(x)	= The absolute value of scalar x
CSV	= Comma Separated Files
Diag($[a_1..a_n]$)	= Diagonal Matrix with a_1 to a_n on the diagonal and zeros in the rest of the matrix
DOF	= Degrees of Freedom
ECEF	= Earth-Centered Earth-Fixed
EKF	= Extended Kalman Filter
GNSS	= Global Navigation Satellite System
GPS	= Global Positioning System
I3	= The 3×3 identity matrix.
IMU	= Inertial Measurement Unit
INS	= Inertial Navigation System
KF	= Discrete Kalman Filter
Max(x,y)	= Extracts the maximal value of two scalars x and y
NED	= North-East-Down
OF	= Optical Flow
RMSE	= Root Mean Squared Error
SIFT	= Scale Invariant Feature Transform
SURF	= Speeded-Up Robust Features
UAV	= Unmanned Aerial Vehicle
UTC	= Coordinated Universal Time

Part I

Preliminaries

Chapter 1

Introduction

This master thesis explores the possibility of using a commercial video camera to estimate the linear and angular velocity of a fixed-wing unmanned aerial vehicle (UAV) with a principle called optical flow (OF). The estimated linear velocity is used as a measurement in a nonlinear observer to estimate the ego-motion of the fixed-wing UAV. In addition an approach for detection and tracking of moving objects in images captured from a fixed-wing UAV is developed. This approach also depends on OF and is therefore related to the estimation of the ego-motion in this manner. Furthermore the detection and tracking of moving objects depend on the UAV states and thus a state estimator is necessary. Finally, the estimated ego-motion of the UAV is used to compute georeferenced position and velocity over ground estimates for the detected objects. This chapter contains the following topics:

- Section 1.1 provides a short introduction to the research field with UAVs as the main focus. Furthermore the motivation behind this thesis is presented with some of the main challenges.

- Section 1.2 presents the contributions of this thesis and specifies the problem formulation.

- Section 1.3 presents an outline of the contents in this thesis.

1.1 Historical View and Motivation

The use of UAVs, for both military and civil purposes, has received a lot of interest in the last decade, and already plays a major role in military use. The field of applications for UAVs will grow even more in the future, and the demands for reliable systems are considered to be crucial. Safe navigation is one of the most important topics when working with UAVs since they should be able to move safely without an operator controlling the aircraft. Camera based navigation is a globally prioritized research field and expected to be useful in civil and military applications for the next decades. Integration of a camera in the navigation system increases the redundancy in the navigation system. This can be especially important for UAVs since magnetometer measurements of the heading might be heavily affected by electromagnetic fields from other electrical components. The limited amount of space in the UAV increases this problem since it is difficult to shield the magnetometer from other electrical components.

Cameras can be useful for surveillance, obstacle avoidance, velocity estimation, object detection and object tracking in addition to several other applications. Therefore it is a sensor that can gather an enormous amount of information. In this thesis the images captured from a camera strapped to the UAV will be used to calculate the linear and angular velocity of the UAV, and in addition find and track moving objects. This illustrates the diversity of the applications for which a video camera can be useful. The use of cameras for navigational purposes is expected to grow fast since video cameras are lightweight, and the prices are constantly decreasing. Therefore cameras with great quality can be used in commercial applications as well.

The knowledge of camera based navigation is not comprehensive enough yet and there is a research space ready to be examined. This is especially the case for applications concerning UAVs. UAVs are moving fast which causes the images to be more contaminated by noise than images captured from a camera at rest. Moreover the displayed environment in the images changes rapidly when the velocity is large. Therefore suitable algorithms that can handle these situations are important to identify or develop. Detection and tracking of moving objects are very mature research fields, but most work is developed for static environments where the camera is at rest. Therefore most approaches developed so far are not suitable for UAVs and other approaches are necessary to identify.

Other challenges with image processing applications in UAVs, are the limited weight and space available for sensors in the payload. It is always desirable to have small, light weight and power efficient sensors on an airborne vehicle. A trade-off between price, performance, size and weight is necessary when choosing a video camera. Therefore there is a limited set of appropriate cameras available. The processing capacity of on-board processors today is limited, and image processing algorithms are known to be computationally heavy. Thus the use of a video camera in the navigation system requires algorithms that can be processed in an efficient manner for online applications. The need for accurate measurements is a contradiction to the desire for fast algorithms, and this is one of the main challenges for real-time applications. This however is not the main focus in this thesis since the algorithms will be tested offline.

1.2 Problem Formulation Specification and Contributions

This thesis has two main objectives. The first objective is to estimate the ego-motion of a fixed-wing UAV with a nonlinear observer. The nonlinear observer [24] has earlier been used with a magnetometer, but in this thesis the magnetometer is replaced by a video camera. The nonlinear observer with the camera has earlier been proved to be uniformly semi-globally stable and verified through a computer simulation [21]. In this thesis a real experiment has been conducted to test the nonlinear observer on real data. Several OF algorithms have been implemented and the angular and linear velocity have been calculated with OF. The implementation of the OF algorithms has been conducted in cooperation with fellow MSc. student Jesper Hosen. OF and velocity calculation from OF are described in Chapter 2. The calculated velocities have been used as a measurement in the nonlinear observer. The observer has been implemented and verified through offline simulation of real data from a flight experiment conducted at Eggemoen February 2015.

The nonlinear observer has been simulated with the data gathered at the experiment at Eggemoen, and the results have been published in a paper accepted for the International Conference on Unmanned Aerial Vehicles 2015 (ICUAS'15) in cooperation with PhD. student Lorenzo Fusini, MSc. student Jesper Hosen, Professor Tor Arne Johansen and Professor Thor Inge Fossen. The accepted paper is attached in Appendix A of this thesis. A second paper has been written and submitted to the 2016 American Institute of Aeronautics and Astronautics Science and Technology Forum and Exposition (AIAA SciTech 2016) and is based on the data gathered from the same flight test. This paper focuses on a different way to calculate the linear body-fixed velocity with OF. Furthermore the nonlinear observer is evaluated with simulated and real data and compared with the observer from the first paper. The second paper is attached in Appendix B and will be accepted or rejected late in August 2015. The second paper is also written in cooperation with the aforementioned authors. The nonlinear observer is described more closely in Chapter 3.

The second objective of this thesis is detection and tracking of moving objects. A comprehensive study of existing methods and important theory are presented in Chapter 4. A new algorithm for detection of moving objects with OF has been developed in this thesis and is also described more closely in this chapter. It has been implemented and evaluated through computer simulations. The algorithm has not been verified on real data since the images from the experiment at Eggemoen did not contain moving objects with known paths. Furthermore it was not possible to perform a new experiment within the time limit. However objects have been edited into the images captured at Eggemoen. In this manner the objects can be detected and tracked in relevant images and the real data of the UAV have been used. Therefore the results from the simulation should illustrate the performance of the detection algorithm in a trustworthy way.

The detected moving objects are tracked in the image plane in order to create a time dependent trajectory for the motion of the objects. A tracking system utilizing the moving object detection algorithm, a classifier to describe each object and a Kalman filter has been developed. OF and the position of the detected objects have been used as measurements of the velocity and the position in the image plane. The estimated ego-motion of the UAV (from the nonlinear observer) has been used to compute the position and velocity of the moving objects in the North-East-Down (NED) coordinate frame [20, Chapter 2]. The tracking system has also been implemented and tested through computer simulations. Different paths in NED have been created and objects inserted into the images at the correct position in the image. The estimated trajectory in NED has been compared with the true paths to evaluate the tracking system.

The following contributions have been made in this thesis:

- Co-author on paper accepted for ICUAS'15. The paper evaluates the nonlinear observer on real data, and is attached in Appendix A.
- Co-author on paper submitted for AIAA SciTech 2016. The paper is attached in Appendix B and proposes a new way to utilize the video camera in the nonlinear observer. The paper evaluates the nonlinear observer on real and simulated data.
- Derivation of how to calculate the body-fixed velocity from OF with a camera pointing straight towards the ground (with zero roll and pitch for the UAV). This is derived in Section 2.4.
- Development of an algorithm for detection of moving objects utilizing OF and the navigation states of the UAV. This is described in Section 4.2.
- Development of a tracking system for multiple moving objects utilizing the moving object detection algorithm. This is described in Section 4.2-4.5.
- Implementation of the nonlinear observer and the tracking system. This is described in Chapter 5.
- A comparison of the performance of the nonlinear observer with a EKF based on real data. Furthermore the performance of the observer with the camera is

compared to the performance of the nonlinear observer with a magnetometer. This is presented in Section 7.2-7.3.

- Case studies evaluating the performance of the moving object detection algorithm and the tracking system. This is presented in Section 7.4-7.6.

1.3 Outline

This thesis is organized in eight chapters and three appendices. The chapters are numbered 1-8 and the appendices are numbered A-C. A short description of the contents is given below:

- Chapter 2 presents the camera system. An introduction to OF is given and the OF algorithms used in this thesis are described. Furthermore the calculation of body-fixed linear and angular velocity from OF is derived.
- Chapter 3 presents the nonlinear observer used to estimate the ego-motion of a fixed-wing UAV. The equations for the observer are presented and different configurations of the observer described.
- Chapter 4 presents the moving object detection algorithm and tracking system. This includes the moving object detection algorithm, object classification, object tracking and transformation of estimates in the image plane to NED (georeferencing). Related work is also comprehensively described.
- Chapter 5 describes the software implemented in this thesis. This includes the implementation of OF algorithms, velocity calculation from OF, moving object detection, tracking system and other related software.
- Chapter 6 describes the UAV experiment, case studies and simulations conducted in this thesis. This chapter is the basis for understanding the results presented in Chapter 7.
- Chapter 7 presents the results and discussion for the different case studies. The result and discussion for each case study are presented separately.
- Chapter 8 presents a conclusion and the main findings of this thesis. Future work is also described in this chapter.
- Appendix A is the paper accepted for ICUAS'15.
- Appendix B is the paper submitted to AIAA SciTech 2016.
- Appendix C shows how the transformation between NED and the camera-fixed frame is calculated using the symbolic toolbox in Matlab.

Part II

Theory

Chapter 2

Image Processing

Video cameras can be used for a lot of different applications including surveillance, obstacle avoidance, safe navigation, object detection and object tracking. In recent years commercial video cameras have been added to the payload of many UAVs since it is a cheap and light weight sensor which can capture a huge amount of information. In this chapter a video camera will be used to calculate the body-fixed velocity of the UAV with a principle called OF. This chapter looks into the calculation of velocity from OF and contains the following topics:

- Section 2.1 is an introduction to image processing and important for readers with little experience with image processing and computer vision.
- Section 2.2 defines OF and introduces the research field.
- Section 2.3 describes different OF methods used in this thesis.
- Section 2.4 presents the calculation of body-fixed velocity from OF.

2.1 Introduction to Image Processing

Video recordings are simply a sequence of images captured closely in time. Therefore a sequence of images is the same as a video recording. An image is a representation of the perceived area in the field of view of the camera captured at a time instant. Thus an image is a discrete representation and can contain different amounts of noise. This is why two images of the exact same area, captured in a short time period, never look exactly equal. There exist several different color representations for images. In this thesis color images stored with the RGB format and monochrome images will be considered [48, Chapter 2].

Color images are based on detection of three different colors, namely red, green and blue. Other colors are represented as a combination of red, green and blue. A camera consists of an image sensor with a finite amounts of pixels. Color cameras normally use an image sensor where each pixel is divided in four regions. Two of these regions detect the color green since the human eye is most sensitive to green. The two other regions detect red and blue. Such a pattern is called a Bayer pattern and is displayed in Figure 2.1. Each color gets a value that describes the intensity (amount) of the color. Monochrome images express each pixel as a single value which determines the brightness of the image. The intensity value is normally an eight-bit number with values from 0 to 255.

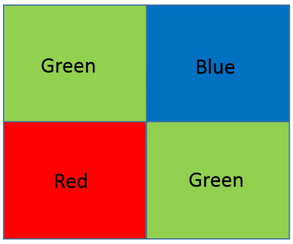


Figure 2.1: The Bayer filter at each pixel.

2.1.1 The Pinhole Camera Model

A camera model can be defined as

Definition 2.1 A camera model is a mathematical model describing the relationship between the depicted object and the perceived image. It is a mathematical model relating the position of objects in the 2D image plane to the position in a 3D coordinate frame fixed to the camera.

There are several camera models, but one of the simplest with high validity is the pinhole camera model [29]. In order to relate pixel coordinates to other coordinate systems, it is necessary to map a point in the 3D camera-fixed frame to the 2D image plane. The pinhole camera model is a well known camera model which defines this mapping in a simple manner. It is displayed in Figure 2.2. The following two definitions define coordinates in the image plane and the camera-fixed frame:

Definition 2.2 Superscript ^{c} refers to the camera-fixed frame. The origin of this frame is placed in the lens aperture. Position coordinates in the camera-fixed frame are expressed as $\mathbf{p}^c = (x^c, y^c, z^c)$.

Definition 2.3 r and s is the horizontal and vertical pixel coordinate in the image plane respectively.

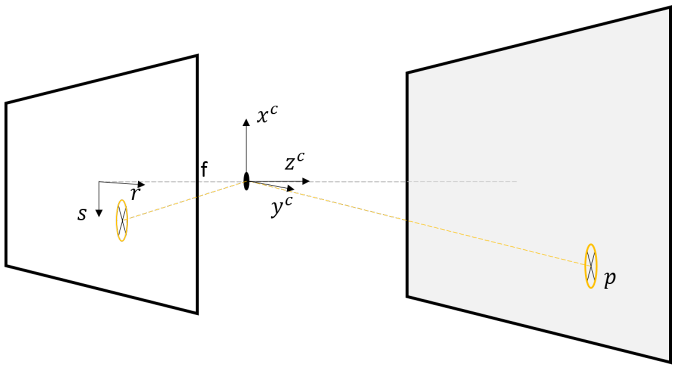


Figure 2.2: The pinhole camera model maps a perceived point in the camera-fixed frame to the image plane.

By geometric considerations (similarity of form) one may see from Figure 2.2 that the relationship between r and y^c and the relationship between s and x^c can be written as

$$\frac{r}{f} = \frac{y^c}{z^c} \tag{2.1a}$$

$$\frac{s}{f} = -\frac{x^c}{z^c} \tag{2.1b}$$

where f is the focal length of the camera given by the lens specification. (2.1a) and (2.1b) might be combined and expressed as

$$\begin{bmatrix} r \\ s \end{bmatrix} = \frac{f}{z^c} \begin{bmatrix} y^c \\ -x^c \end{bmatrix}, z^c \neq 0 \tag{2.2}$$

(2.2) describes the relationship between the image plane coordinates (r, s) and the camera-fixed coordinates (x^c, y^c, z^c). z^c is the distance between the lens aperture and the plane the captured point is located in. Obviously the coordinate z^c can never be zero, but this is not a problem for UAVs since z^c is proportional to the altitude of the UAV. A necessary assumption for the pinhole model to be valid is:

Assumption 2.1 The distance z^c is known such that a one-to-one mapping between the image plane and the camera-fixed frame is given by (2.2).

The pinhole camera model will later be used to derive the calculation of velocity from OF. Therefore the validity of the model should be considered and necessary assumptions stated. The pinhole camera model is a good approximation, but it assumes that the images are not affected by lens distortion and that the image projection is continuous. In general all cameras are prone to lens distortion to some extent. Lens distortion could e.g. be identified as straight lines appearing as bended lines. This is because a lens has defects that lead to blur, color changes or geometric distortion from the ideal ray [48, Chapter 3]. As the geometric distortion is most present in the peripheral of an image, the assumption of no lens distortion might be valid if one considers a region of the image extending some distance from the centre of the image. For simplicity the following assumption is made in this thesis:

Assumption 2.2 The lens used does not introduce significant distortion to the images.

The distortion of a camera can be tested with the Computer Vision toolbox in Matlab. Since the number of pixels in the image plane is finite, the image plane consists of a fixed set of points. Therefore the image plane is not continuous, but discrete. Thus the points in the 3D plane, which is continuous, are not mapped perfectly to the image plane. In order to find the velocity from OF the pinhole camera model needs to be differentiated and this is derived in the continuous time space. Therefore the following assumption is necessary:

Assumption 2.3 The image projection is continuous such that the image coordinates can be differentiated with respect to time.

With the large number of pixels in cameras today, the assumption of continuous image projection is not far from reality.

2.2 Optical Flow

OF has been studied for nearly 50 years and belongs to the research field of computer vision. This section will define OF and explain the most important topics within this research field.

2.2.1 Definition of Optical Flow

[5] and [37] defines OF as a velocity field that transforms one image into the next image in a sequence of images. It is therefore necessary to have two subsequent images to calculate OF. In this thesis another, but closely related, definition will be considered. OF can be understood as the difference between two images. It says something important about how images in a sequence change to create the next image. Each image consists of a finite number of OF vectors where the resolution of the image is an upper bound for the number of vectors. For simplicity a single OF

vector can be considered as the vector between the same feature in two consecutive images. It has a magnitude and a direction. This is illustrated in Figure 2.3. It is important to notice that different features in the same image can have different OF vectors. This can be the case if one of the features is located closer to the camera than other features or if the image contains moving objects. In this report OF will be defined as

Definition 2.4 An optical flow vector is the change in position for a feature displayed in two consecutive images. Optical flow of an image is the set of all calculated OF vectors in the image.

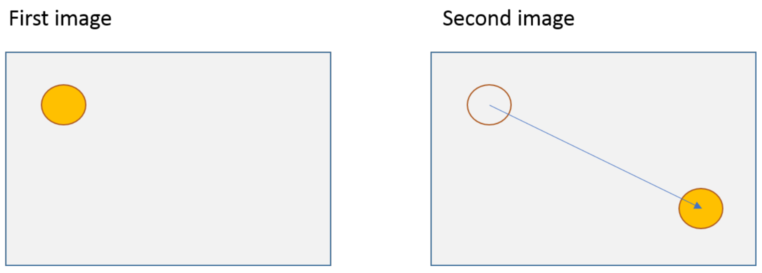


Figure 2.3: The circle, which is a feature, changes position in two consecutive images and the arrow is used to illustrate the optical flow.

Dense and Sparse Optical Flow Algorithms

In Figure 2.3 OF was illustrated as the movement of a common point in two consecutive images. This however is just the OF of a single point (pixel) in the image. In Section 2.4 it is justified that at least three OF vectors from each image are necessary to compute the angular and linear velocity from OF. Different OF methods have different approaches to decide the number of calculated OF vectors. The methods can be classified as either dense or sparse. The dense methods calculate OF at each and every pixel of the image. They are in general computationally demanding since the number of OF calculations is equal to the image resolution. The sparse methods on the other hand calculate OF at a subset of the pixels. These pixels can be picked randomly or be identified through a feature detector for instance. Feature detectors locate parts in an image that are easy to identify. This can for example be regions with large contrast in intensity, edges or corners.

Definition 2.5 Dense optical flow methods calculate optical flow at each and every pixel of the image. Sparse optical flow methods calculate optical flow at a subset of the pixels.

It is important to notice that dense methods not necessarily are more computationally demanding than sparse methods. This is because the methods can use different approaches for calculating OF at a single pixel. Therefore, a dense method that calculates OF at each pixel very efficiently might be less computationally demanding than sparse methods that requires long computation time at each pixel.

A problem with dense algorithms in UAV applications may be when flying over areas with smooth surfaces or over dynamic surfaces which are changing. Such a surface may be the ocean. When flying over a homogeneous surface such as still water one would expect non-zero OF vectors since subsequent images change. However the algorithms may propose that the OF is zero. This is because the image might look equal at several image regions. Then a change at one pixel might be neglected since a new equal pixel is moved to the same location as the original point. Another possibility is that the algorithms are unable to distinguish between two equal pixels. Therefore the pixels are mixed together and wrong OF vectors are generated. An illustration of this problem is given in Figure 2.4. This can lead to OF vectors that are wrong both in magnitude and direction.

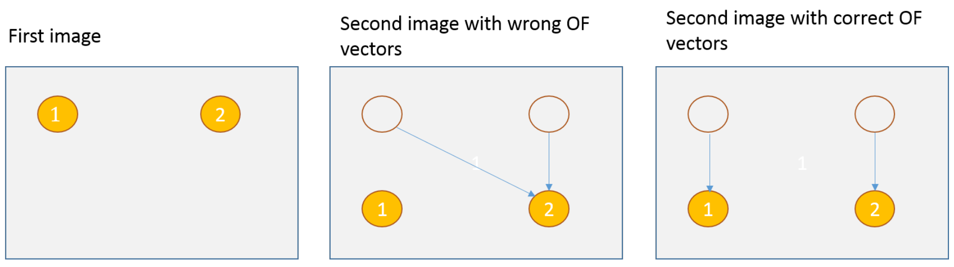

First image
1
2
Second image with wrong OF vectors
1
2
Second image with correct OF vectors
1
2

Figure 2.4: Two equal points (pixels) mixed together and erroneous OF vectors are computed.

Sparse algorithms on the other hand are free to choose points in the image (regions of interest), and thereby avoid using homogeneous areas when calculating the OF. For this to work it is necessary to avoid areas of an image that are inappropriate for OF calculation. This is a very challenging problem and it is impossible to find a solution that works in all cases. Therefore it is impossible to be completely protected from erroneous optical flow vectors. [56], [19] and [5] are all examples of dense methods, while OF calculation based on [33] and [3] are examples of sparse methods.

2.3 Optical Flow Methods

The algorithms for calculating OF have evolved greatly over the last decades. This section will describe some of the methods implemented in this thesis. For a more comprehensive study of available methods the reader should investigate [2, 37].

The algorithms implemented in this thesis are chosen based on the findings in related literature and experiences from when the project report was written. [37] evaluated some algorithms in an offline evaluation based on images captured from a UAV. Other researchers [9, 54, 58] have used a lot of resources on evaluating different algorithms in UAV applications. In this thesis the following methods are considered:

- Template matching
- Feature-matching based OF methods

A single gradient [6] method is implemented in this thesis, but not used in the UAV experiment in Chapter 6 due to the poor results in [37] and findings in the project report. Thus the method will not be described in this thesis. Template matching and feature-matching based methods will be examined further in this section and later used in a UAV experiment.

2.3.1 Template Matching

Template matching [51] is also referred to as region matching. A single OF vector is calculated by identifying the displacement of a small region between subsequent images. This can be conducted for several image regions the get the desired number of OF vectors for every image. The process of calculating a single OF vector is illustrated in Figure 2.5. A region around the pixel of interest, called a template, from the first image is searched for in the second image. The goal is to locate the same region in the second image. The OF vector is the displacement of the template from the first to the second image in the image plane. This process is for the rest of this thesis referred to as template matching.

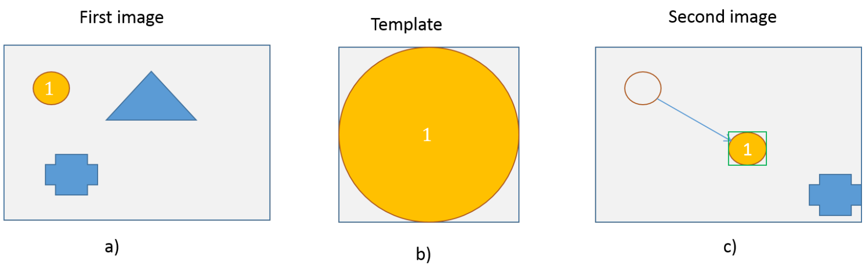


Figure 2.5: a) A template is extracted from the first image. b) The template by itself. c) The template is matched at the marked region in the second image and displacement calculated. The arrow is the OF vector.

Template matching will in this thesis be based on normalized cross correlation.

This process can be mathematically described as

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y')I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \sum_{x',y'} I(x+x',y+y')^2}} \tag{2.3}$$

where $R(x,y)$ is the result which indicates the quality of the match, $T(x,y)$ is the template, $I(x,y)$ is the image where the template is searched for and (x,y) is a pixel position. In practice the template is placed over a part of the image (with the same size as the template) and compared with that part. If the areas are similar, $R(x,y)$ will be equal to one. To find the best match this operation must be performed for every pixel (x,y) in the image $I(x,y)$. Thus the number of times (2.3) is computed is related to the image resolution. The process is illustrated in Figure 2.6.

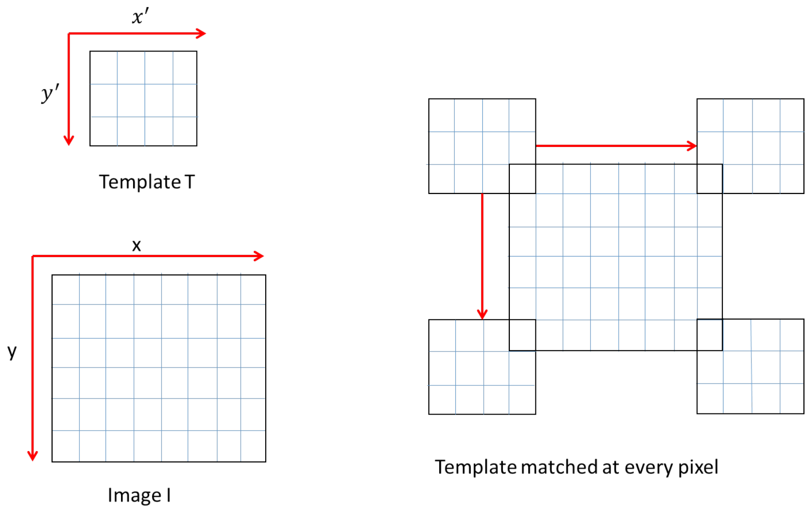


Figure 2.6: Pattern for performing normalized cross correlation at every pixel.

Figure 2.7 shows an example of template matching performed on a real image captured by a UAV. In practice the template matching is performed on a single color channel or a monochrome image. This is because matching of several color channels is to computationally expensive. In this thesis it is only performed template matching on monochrome images which in practice means that the brightness of the image is the matching criterion.

Advantages and Disadvantages With Template Matching

The images in Figure 2.7 illustrates an environment that might be a problem for template matching techniques. If the extracted template displays a homogeneous area, such as snow with no texture, several locations might match the template.

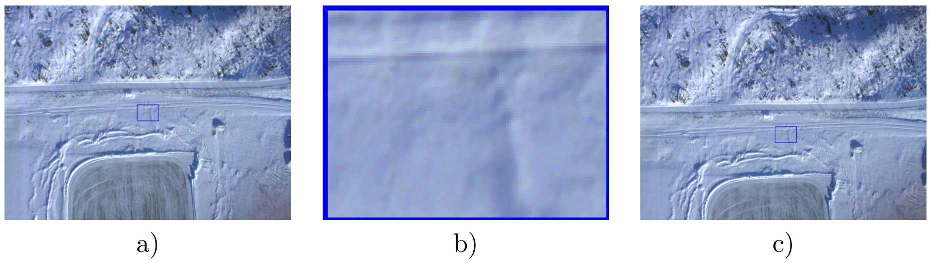


Figure 2.7: a) Template is extracted from the first image. b) The template. c) The template is matched at the marked region in the second image.

Therefore erroneous OF vectors might be created. These techniques are therefore in general not that reliable in homogeneous environments. A possibility for increasing the reliability is to match every color channel independently, and reject the results if every color channel do not provide the same result. This however is computationally demanding since it requires three operations with template matching instead of one.

The main advantage with the region-based methods is their performance for fast motion. A disadvantage is that the procedure is computationally demanding and not appropriate for real-time applications on computers with limited processing power. The template matching is computationally heavy, both in time and space as illustrated in Figure 2.6. Another issue with template matching is that the template should look equal in both images in order to find it. Therefore the displayed area in the template should have the same angle with respect to the camera, and the same number of pixels covering the region, as well as equal light conditions. The problem is minimized by a sufficiently high frame-rate for the camera.

One solution to increase the speed, is to divide the image in smaller regions and search for the template in each region. It is then possible to use parallelism and find the most likely regions for the correct match. A more detailed search can be conducted in the best regions. Decreasing the template size is also a solution to increase the computational speed. Larger templates increase the search time, but gives more reliable results since the search algorithm has more information about the area of interest (a larger number of pixels). Neither of these strategies are used further in this thesis.

How to Choose Templates

One challenge with these methods is to find pixels where OF should be calculated. One approach is to divide the image in regions, and find the OF of the centre of each region. Naturally the region must be larger than the size of the template for this to be reliable. The regions should be equal in size to get OF vectors from all parts of the image. Another approach is to use a feature detector (see section 2.3.2) to choose appropriate features to match between consecutive images. However there will always be a risk of detecting features in one area of the image only. In the worst-case scenario the detector cannot find any features, and thus not a single OF vector is calculated. An example of an OF field created with template matching, is displayed in Figure 2.8.

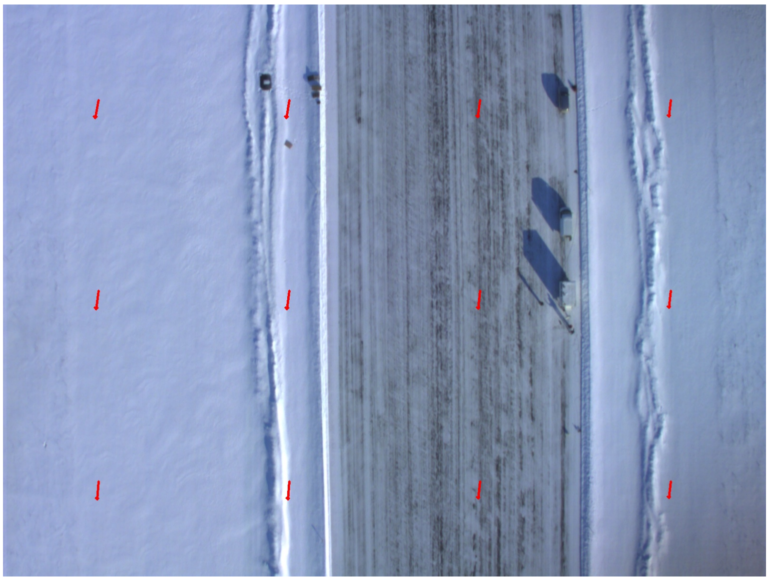


Figure 2.8: Optical flow calculated with template matching.

2.3.2 Feature-Matching Based Methods

Feature-matching based methods calculate OF in three steps:

1. Find interest points, so called features, with a feature detector.

2. Connect a descriptor to each feature that describes the neighbourhood of the feature. The descriptor is used to separate different features and match features together.

3. Match all descriptors from two consecutive images together to find common features. OF vectors are created by the displacement of each feature located in both images.

The process is illustrated in Figure 2.9. Typical features extracted in the first step are edges, lines and corners. The most important property of the detector is the repeatability. The repeatability describes the detectors ability to identify the same interest points in different viewing conditions. These conditions can include properties such as light conditions, angle of rotation, different scale (changes in size) and of course displacements compared to the environment.

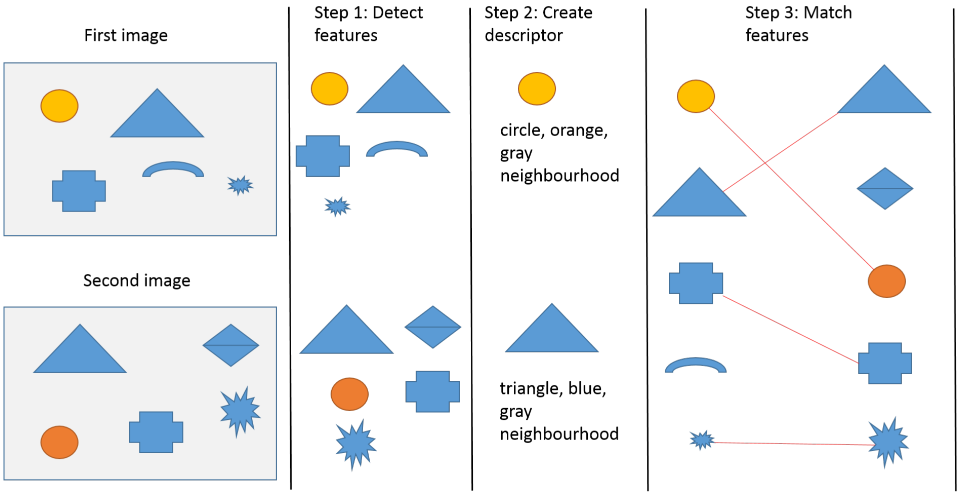

First image
Second image
Step 1: Detect features
Step 2: Create descriptor
circle, orange, gray neighbourhood
triangle, blue, gray neighbourhood
Step 3: Match features

Figure 2.9: Illustration of Feature-Based Matching Methods.

The descriptor from step two in Figure 2.9 is often a vector describing the pixels in the neighbourhood of the feature. The vector could include color information, intensity information, information about the gradient of the colors and so on. The descriptors most important property are to uniquely describe each feature. It needs to be robust to noise and other disturbances such as image distortions. Distinctive descriptors for each interest point is critical such that different interest points are not matched together.

The matching technique in step three is in most cases based on least squares or a similar error minimization method. If the similarity is above a given threshold, the descriptors are considered to be a match. The threshold can be adjusted by the best match to exclude insecure matches. The Harris Algorithm [10], SURF [3] and SIFT [33] are famous feature-matching based OF algorithms. Both SIFT and SURF have been implemented, but only SIFT has been used in the case studies described in Chapter 6.

Advantages and Disadvantages

The main disadvantage with these methods is that they are computationally demanding, and the performance depends heavily on the matching technique chosen. However template matching at several locations is more computationally demanding. The main advantage is that several of these methods are scale and rotation invariant. This means that the features can change in size and rotation, but still be detected by the feature detector and given the same descriptor. Furthermore these methods have the same advantage as template matching with respect to fast motion as well.

Scale Invariant Feature Transform (SIFT)

The SIFT algorithm [33] was the first scale and rotation invariant OF algorithm. It is also partially invariant to brightness changes. The features are detected by locating maxima and minima of a difference-of-Gaussian [33] function at different image resolutions. A vector of 64 or 128 elements is used as the descriptor. An example of OF vectors calculated on a real image with SIFT is displayed in Figure 2.10. SIFT is often able to identify several features as illustrated by the number of OF vectors in the image.

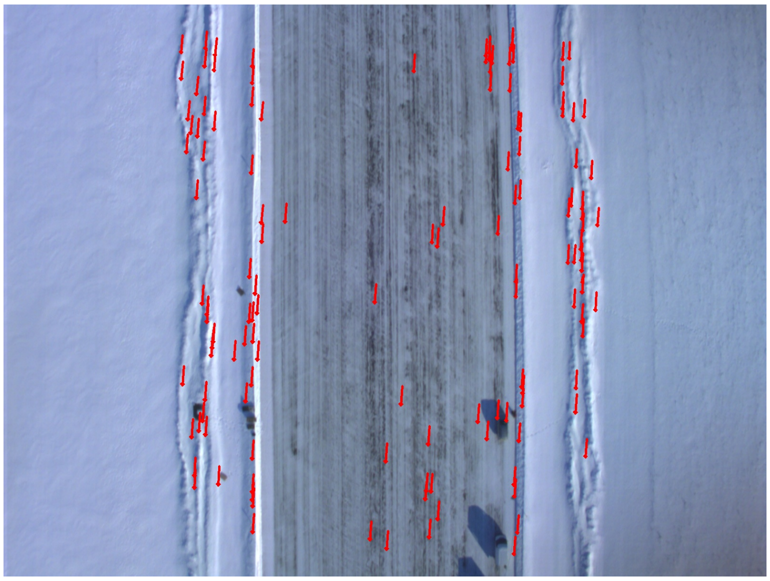


Figure 2.10: Optical flow vectors calculated with the SIFT algorithm.

2.4 Calculation of Velocity From Optical Flow

This section presents a way to calculate the six-degrees of freedom (6 DOF) body-fixed velocity of a UAV based on OF. The method described in this chapter utilizes

OF vectors, roll, pitch and altitude of the UAV in order to calculate the linear and angular body-fixed velocity. This section begins with an introduction to why it is desirable to calculate velocity from OF. Thereafter a relationship between OF vectors and 6 DOF motion of a UAV is derived.

OF based velocity calculation has received a lot of interest the latter years, and multiple researchers have published work in this field [37, 54]. The motivation for studying velocity calculation from OF is mostly justified in the sense of robustness. A vision-aided navigation system should be able to maintain reliable estimates of the states of the UAV, even in the case of GPS drop-outs or magnetometer disturbances.

2.4.1 Calculating the Velocity of the Terrain With Respect to the Camera

The problem of recovering velocity from OF has been studied in [37], [54] and [21] among others. When dealing with Euler angles, the formulas from the aforementioned researchers would yield a singularity when pointing the camera downwards (pitch at 90 degrees). In order to solve the singularity problem with the Euler angles, the transformation between OF and velocity is derived in this thesis, assuming that the camera on the UAV is pointing straight down. This leads to a slightly different transformation from OF to velocity. This transformation was also published in the paper submitted to ICUAS'15 (Appendix A). The placement of the camera is illustrated in Figure 2.11 The camera-fixed coordinate system is depicted in Figure 2.2.

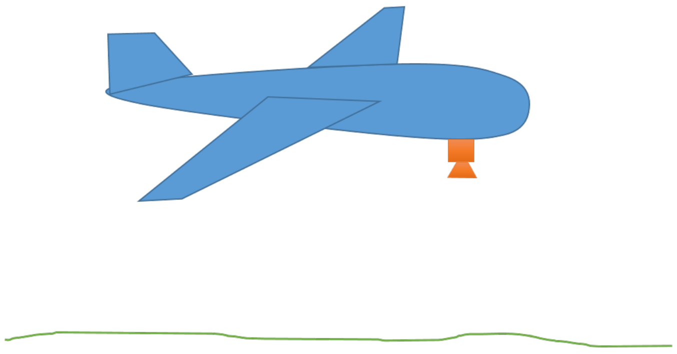


Figure 2.11: The placement of the camera in the UAV. In reality the camera is placed closer to the center of gravity.

From now on vectors and matrices are written as bold upper-case and lower-case letters respectively ($\mathbf{r}$ and $\mathbf{R}$), while scalars are written as non-bold letters (r). In Section 2.1.1 a relationship between points in the 2D image plane and the 3D camera-fixed coordinate frame was established through the pinhole camera model. By assuming that every feature corresponding to an OF vector is at rest, (2.2) can be differentiated with respect to time in order to derive the relationship between velocity and OF (by assumption 2.3).

$$\begin{bmatrix} \dot{r} \\ \dot{s} \end{bmatrix} = \frac{f}{z^c} \begin{bmatrix} 0 & 1 & -\frac{y^c}{z^c} \\ -1 & 0 & \frac{x^c}{z^c} \end{bmatrix} \begin{bmatrix} \dot{x^c} \\ \dot{y^c} \\ \dot{z^c} \end{bmatrix} \tag{2.4}$$

Definition 2.6 The vector $[\dot{r}, \dot{s}]^T$ is the optical flow vector at pixel (r, s) in the image plane, and is a measure of how fast a feature in the image plane moves in horizontal (r) and vertical (s) direction between consecutive images.

The vector $[\dot{x^c}, \dot{y^c}, \dot{z^c}]^T$ of the right hand side can be recognized as

$$\begin{aligned} \dot{\mathbf{p}}^c &= \begin{bmatrix} \dot{x^c} \\ \dot{y^c} \\ \dot{z^c} \end{bmatrix} \\ &= \mathbf{v}^c_{T/c} + \boldsymbol{\omega}^c_{T/c} \times (\mathbf{p}^c - \mathbf{o}^c_T) \end{aligned} \tag{2.5}$$

where the following definitions are necessary:

Definition 2.7 $\mathbf{v}^c_{T/c}$ is the linear velocity of the terrain with respect to the camera represented in the camera-fixed coordinate system.

Definition 2.8 $\boldsymbol{\omega}^c_{T/c}$ is the angular velocity of the terrain with respect to the camera represented in the camera-fixed coordinate system.

Definition 2.9 $\mathbf{o}^c_T$ is the terrain origin/point of rotation in camera coordinates. All rotations of the terrain seen in the image will be rotations about the camera, hence the rotation point $\mathbf{o}^c_T$ coincides with the origin of the camera frame.

Definition 2.10 $\mathbf{p}^c$ is the position of the OF vector in the camera-fixed coordinate system.

(2.4) might be rewritten by inserting (2.5)

$$\begin{bmatrix} \dot{r} \\ \dot{s} \end{bmatrix} = \frac{f}{z^c} \begin{bmatrix} 0 & 1 & -\frac{y^c}{z^c} \\ -1 & 0 & \frac{x^c}{z^c} \end{bmatrix} (\mathbf{v}^c_{T/c} + \boldsymbol{\omega}^c_{T/c} \times (\mathbf{p}^c - \mathbf{o}^c_T)) \tag{2.6}$$

$$= \frac{f}{z^c} \left[\begin{matrix} 0 & 1 & -\frac{y^c}{z^c} \\ -1 & 0 & \frac{x^c}{z^c} \end{matrix} \;\middle|\; \begin{matrix} 0 & 1 & -\frac{y^c}{z^c} \\ -1 & 0 & \frac{x^c}{z^c} \end{matrix}\right] \begin{bmatrix} \mathbf{v}^c_{T/c} \\ \boldsymbol{\omega}^c_{T/c} \times (\mathbf{p}^c - \mathbf{o}^c_T) \end{bmatrix} \tag{2.7}$$

$$= \frac{f}{z^c} \left[\mathbf{A} \;\middle|\; \mathbf{A}\right] \begin{bmatrix} \mathbf{v}^c_{T/c} \\ \boldsymbol{\omega}^c_{T/c} \times (\mathbf{p}^c - \mathbf{o}^c_T) \end{bmatrix} \tag{2.8}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & -\frac{y^c}{z^c} \\ -1 & 0 & \frac{x^c}{z^c} \end{bmatrix}$$

By the properties of the crossproduct

$$\begin{aligned} \boldsymbol{\omega}^c_{T/c} \times (\mathbf{p}^c - \mathbf{o}^c_T) &= -(\mathbf{p}^c - \mathbf{o}^c_T) \times \boldsymbol{\omega}^c_{T/c} \\ &= -\mathbf{S}(\mathbf{p}^c - \mathbf{o}^c_T)\boldsymbol{\omega}^c_{T/c} \end{aligned} \tag{2.9}$$

where $\mathbf{S}(\mathbf{v})$ is the skew-symmetric matrix

$$\mathbf{S}(\mathbf{v}) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{2.10}$$

It is now possible to rewrite (2.8) as

$$\begin{aligned} \begin{bmatrix} \dot{r} \\ \dot{s} \end{bmatrix} &= \frac{f}{z^c} \begin{bmatrix} \mathbf{A} & \mid & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{v}^c_{T/c} \\ -\mathbf{S}(\mathbf{p}^c - \mathbf{o}^c_T)\boldsymbol{\omega}^c_{T/c} \end{bmatrix} \\ &= \frac{f}{z^c} \begin{bmatrix} \mathbf{A} & \mid & -\mathbf{A} \cdot \mathbf{S}(\mathbf{p}^c - \mathbf{o}^c_T) \end{bmatrix} \begin{bmatrix} \mathbf{v}^c_{T/c} \\ \boldsymbol{\omega}^c_{T/c} \end{bmatrix} \\ &= \mathbf{M}'(f, \mathbf{p}^c, \mathbf{o}^c_T) \begin{bmatrix} \mathbf{v}^c_{T/c} \\ \boldsymbol{\omega}^c_{T/c} \end{bmatrix} \end{aligned} \tag{2.11}$$

where $\mathbf{p}^c = [x^c, y^c, z^c]^T$, $\mathbf{o}^c_T = [x^c_0, y^c_0, z^c_0]^T$ and

$$\mathbf{M}'(f, \mathbf{p}^c, \mathbf{o}^c_T) = \tag{2.12}$$

$$\frac{f}{z^c} \begin{bmatrix} 0 & 1 & -\frac{y^c}{z^c} & -\frac{y^c(y^c - y^c_0)}{z^c} - (z^c - z^c_0) & \frac{y^c(x^c - x^c_0)}{z^c} & x^c - x^c_0 \\ -1 & 0 & \frac{x^c}{z^c} & \frac{x^c(y^c - y^c_0)}{z^c} & -\frac{x^c(x^c - x^c_0)}{z^c} - (z^c - z^c_0) & y^c - y^c_0 \end{bmatrix} \tag{2.13}$$

A relationship between linear and angular velocity and OF is now established through (2.11). It is necessary to argue for the centre of rotation $\mathbf{o}^c_T$. All rotations of objects seen in the image will be rotations about the UAV, hence the rotation point $\mathbf{o}^c_T$ coincides with the origin of the body frame. As the camera is placed close to the centre of Gravity of the UAV the following assumption is reasonable:

Assumption 2.4 The camera-fixed coordinate system coincides with the body-fixed reference frame.

By this assumption

$$\mathbf{o}^c_T = [0, 0, 0]^T \tag{2.14}$$

which means that the centre of rotation coincides with the origin of the camera-fixed coordinate system (body-fixed reference frame of the UAV). Inserting (2.14) into (2.11) one finally arrive at an expression for the linear and angular velocity of the terrain with respect to the camera in the camera-fixed frame

$$\begin{aligned}\begin{bmatrix}\dot{r}\\ \dot{s}\end{bmatrix} &= \frac{f}{z^c}\begin{bmatrix}0 & 1 & -\frac{y^c}{z^c} & -\frac{y^c(y^c)}{z^c} - z^c & \frac{y^c x^c}{z^c} & x^c \\ -1 & 0 & \frac{x^c}{z^c} & \frac{x^c y^c}{z^c} & -\frac{x^c(x^c)}{z^c} - z^c & y^c\end{bmatrix}\begin{bmatrix}\mathbf{v}^c_{T/c}\\ \boldsymbol{\omega}^c_{T/c}\end{bmatrix} \\ &= \mathbf{M}(f, \mathbf{p}^c)\begin{bmatrix}\mathbf{v}^c_{T/c}\\ \boldsymbol{\omega}^c_{T/c}\end{bmatrix}\end{aligned} \tag{2.15}$$

There is one problem with (2.15). There are six unknowns ($\mathbf{v}^c_{T/c}$ and $\boldsymbol{\omega}^c_{T/c}$), but only two equations. Thus the system is underdetermined. The solution is to extended the equation to concern N points $(r_1, s_1) \dots (r_N, s_N)$:

$$\begin{bmatrix}\dot{r_1}\\ \dot{s_1}\\ \vdots \\ \dot{r_N}\\ \dot{s_N}\end{bmatrix} = \begin{bmatrix}\mathbf{M}(f, \mathbf{p}^C_1)\\ \vdots \\ \mathbf{M}(f, \mathbf{p}^C_N)\end{bmatrix}\begin{bmatrix}\mathbf{v}^c_{T/c}\\ \boldsymbol{\omega}^c_{T/c}\end{bmatrix} \tag{2.16}$$

This is possible since the velocity of the terrain with respect to the UAV is fixed, but the OF might vary at different points in the terrain. By calculating the pseudoinverse of $\mathbf{M}$ in (2.16) the angular and linear velocity can be computed as

$$\begin{bmatrix}\mathbf{v}^c_{T/c}\\ \boldsymbol{\omega}^c_{T/c}\end{bmatrix} = \begin{bmatrix}\mathbf{M}(f, \mathbf{p}^c_1)\\ \vdots \\ \mathbf{M}(f, \mathbf{p}^c_N)\end{bmatrix}^+ \begin{bmatrix}\dot{r_1}\\ \dot{s_1}\\ \vdots \\ \dot{r_N}\\ \dot{s_N}\end{bmatrix} = \mathbf{D}^+ \begin{bmatrix}\dot{r_1}\\ \dot{s_1}\\ \vdots \\ \dot{r_N}\\ \dot{s_N}\end{bmatrix} \tag{2.17}$$

where

$$\mathbf{D} = \begin{bmatrix}\mathbf{M}(f, \mathbf{p}^c_1)\\ \vdots \\ \mathbf{M}(f, \mathbf{p}^c_N)\end{bmatrix} \tag{2.18}$$

$\mathbf{D}^+$ is the pseudoinverse of $\mathbf{D}$ and can be calculated as $(\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T$. It exists if $\mathbf{D}^T\mathbf{D}$ has full rank for all states. This can only happen if the number of OF vectors are greater or equal to three. In addition the three vectors cannot be located on the same line. This is because the system of equations will be linearly dependent in this case and the system undetermined. A minimum demand is therefore that at least three OF vectors must be present when calculating velocities, and the vectors cannot be located on the same line. Nevertheless more than three OF vectors are desired in order to find a more accurate solution. The following assumption is made for the rest of this thesis:

Assumption 2.5 The matrix $\mathbf{D}$ is assumed to have full rank as long as the number of OF vectors is sufficient. Thereby it is possible to calculate the linear and angular velocity from (2.17)

2.4.2 Calculating the Velocity of the Terrain With Euler Angles and Altitude

The matrix $\mathbf{D}$ in (2.17) depends on $\mathbf{p}^c$. $\mathbf{p}^c$ is the position of a feature corresponding to an OF vector in the camera-fixed frame. However it is more convenient to express $\mathbf{A}$ as a function of the roll, pitch and altitude of the UAV instead of a position in the camera-fixed coordinate frame. This is because only the position in the image plane for the feature is known. Therefore it is necessary to find a relationship between the velocity and the states of the UAV. This section will derive a relationship between $\mathbf{p}^c$ and the focal length of the camera and roll, pitch and altitude of the UAV.

The rotation matrix $\mathbf{R}_c^n(\Theta)$ and displacement vector $\mathbf{c}^n$ represent a rotation and a translation between NED and the camera-fixed frame. They can be merged to form a homogeneous 4×4-transform $\mathbf{T}_c^n$ between NED and the camera-fixed frame

$$\mathbf{T}_c^n = \begin{bmatrix} \mathbf{R}_c^n(\Theta) & \mathbf{c}^n \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.19}$$

If $\mathbf{t}^n = [x^n, y^n, z^n, 1]^T$ is a vector expressed in NED and $\mathbf{t}^c = [x^c, y^c, z^c, 1]$ a vector expressed in the camera-fixed frame, the relationship between the vectors is given as

$$\mathbf{t}^c = (\mathbf{T}_c^n)^{-1}\mathbf{t}^n \tag{2.20}$$

where the inverse of the homogeneous transformation is given as

$$(\mathbf{T}_c^n)^{-1} = \begin{bmatrix} (\mathbf{R}_c^n(\Theta))^T & -(\mathbf{R}_c^n(\Theta))\mathbf{c}^n \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.21}$$

By (2.20), $\mathbf{t}^c$ is now a function of $\mathbf{c}^n$ and Θ. That are the UAV position in NED and the Euler angles (attitude of UAV). By inserting (2.20) into the pinhole camera model (2.2) one can solve the equation with respect to x^n and y^n by assuming that the Down position of the feature z^n and image coordinates (r, s) are known. The solution is defined as x_T^n and y_T^n. The equation can be solved in Matlab using the symbolic toolbox (Appendix C). By constructing a new vector with the solution $\mathbf{t}_T^n = [x_T^n, y_T^n, z^n, 1]^T$, one can calculate the corresponding solution in the camera-fixed frame. $\mathbf{t}_T^c$ can be calculated by employing the 4×4-transformation again.

$$\mathbf{t}_T^c = (\mathbf{T}_c^n)^{-1}\mathbf{t}_T^n \tag{2.22}$$

Now by defining $\mathbf{p}^c$ as the three first rows of $\mathbf{t}_T^c$, one ends up with

$$\mathbf{p}^c = \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = \begin{bmatrix} \frac{s(c_z^n - z^n)}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{r(c_z^n - z^n)}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{f(c_z^n - z^n)}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \end{bmatrix} \tag{2.23}$$

where the rest of the terms in (2.23) are defined as:

Definition 2.11 c_z^n is the z-component of the camera position in NED frame. This is the altitude of the UAV (with a minus sign since positive direction is downwards). ϕ and θ is the roll and pitch angles of the UAV respectively.

The Down position z^n of the feature is unknown. The altitude of the UAV is on the other side known and thus the following assumption is made:

Assumption 2.6 The terrain is flat such that all features are located in the same plane. In practice this means that $z^n = 0$.

A relationship between OF and roll, pitch and altitude of the UAV is derived by inserting (2.23) into (2.15). The roll and pitch angles can be measured by an inclinometer and the altitude by GPS or altimeter. Therefore the velocity calculation from OF can be performed with available sensors and not depend on a state estimator.

2.4.3 Calculating the Body-fixed Velocity of the UAV With Euler Angles and Altitude

(2.17) expresses the velocity of the terrain with respect to the camera-fixed frame ($\mathbf{v}^c_{T/c}$ and $\boldsymbol{\omega}^c_{T/c}$). However an expression for the body-fixed velocity of the UAV is desirable. This is the same as the camera-fixed velocity with respect to the terrain given in the camera-fixed coordinate frame, namely $\mathbf{v}^c_{c/T}$ and $\boldsymbol{\omega}^c_{c/T}$. Since it is assumed that the body-fixed frame coincides with the body-fixed frame, $\{c\} = \{b\}$ and the body-fixed velocity is given as

$$\begin{bmatrix} \mathbf{v}^c_{c/T} \\ \boldsymbol{\omega}^c_{c/T} \end{bmatrix} = \begin{bmatrix} \mathbf{v}^b_{b/T} \\ \boldsymbol{\omega}^b_{b/T} \end{bmatrix} = - \begin{bmatrix} \mathbf{v}^c_{T/c} \\ \boldsymbol{\omega}^c_{T/c} \end{bmatrix} \tag{2.24}$$

This results in the following transformation for body- fixed velocity based on OF:

$$\begin{bmatrix} \mathbf{v}^b_{b/T} \\ \boldsymbol{\omega}^b_{b/T} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \mathbf{v}^c_{c/T} \\ \boldsymbol{\omega}^c_{c/T} \end{bmatrix} = - \begin{bmatrix} \mathbf{M}(f, \phi, \theta, c_z^n, r_1, s_1) \\ \vdots \\ \mathbf{M}(f, \phi, \theta, c_z^n, r_N, s_N) \end{bmatrix}^+ \begin{bmatrix} \dot{r_1} \\ \dot{s_1} \\ \vdots \\ \dot{r_N} \\ \dot{s_N} \end{bmatrix} \tag{2.25}$$

For clarity the $\mathbf{M}$-matrix is now defined:

Definition 2.12 The matrix $\mathbf{M}(f,\phi,\theta,c_z^n,r,s)$ relates optical flow to angular and linear velocity

$$\mathbf{M}(f,\phi,\theta,c_z^n,r,s) = \frac{f}{z^c}\begin{bmatrix} 0 & 1 & -\frac{y^c}{z^c} & -\frac{y^c(y^c)}{z^c} - z^c & \frac{y^c x^c}{z^c} & x^c \\ -1 & 0 & \frac{x^c}{z^c} & \frac{x^c y^c}{z^c} & -\frac{x^c(x^c)}{z^c} - z^c & y^c \end{bmatrix}$$

$$\begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = \begin{bmatrix} \frac{s c_z^n}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{r c_z^n}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{f c_z^n}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \end{bmatrix}$$

How Velocity Calculation is Affected by Difference in Terrain Elevation

For the derived relationship to be valid it was necessary to assume that the terrain being recorded is flat. This section looks into the ramifications if the assumption is violated. This can be that the level of the ground is constant, but that the UAV is flying above some trees. Then the distance down to the perceived object (here the trees) would be less than the distance to the ground. This problem will now be illustrated through an example, in order to illustrate how the calculated velocities are influenced when assuming wrong distance down to the perceived object.

Example 2.1 Assume that the UAV is flying 150 meters above the ground level. At time t, the UAV flies over a forest. The height of the trees in the forest is 20 meters. Thus the distance from the camera to the object perceived by the camera is 130 meters. The UAV has roll and pitch equal to zero. Furthermore assume that the UAV is moving straight forward with zero angular velocity. Hence $v = w = p = q = r = 0$. The relationship between optical flow and velocity can then be simplified to:

$$\begin{aligned} \begin{bmatrix} \dot{r} \\ \dot{s} \end{bmatrix} &= \begin{bmatrix} 0 \\ \frac{f}{z^c} \end{bmatrix} u \\ &= \begin{bmatrix} 0 \\ \frac{f}{-\frac{f c_z^n}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))}} \end{bmatrix} u \\ &= \begin{bmatrix} 0 \\ -\frac{f}{c_z^n} \end{bmatrix} u \\ &= \begin{bmatrix} 0 \\ -f\frac{1}{c_z^n} \end{bmatrix} u \end{aligned} \tag{2.26}$$

Now for simplicity assume focal length of camera lens $f = 0.01m$, and that the OF algorithm measures an optical flow of $[\dot{r} = 0, \dot{s} = 2.3 * 10^{-3}] m/sec.$
Since one are expecting the UAV to be 150 meters (meaning $c_z^n = -150$) above the ground, the speed is calculated to be

$$u = -\dot{s} \times c_z^n \times \frac{1}{0.01} = -2.3 * 10^{-3} \times (-150) \times 100 = 34.5[m/s]$$

However, since the true distance down to the perceived object is only 130 meters, the real linear forward velocity u is:

$$u = 2.3 * 10^{-3} \times 130 \times 100 = 29.9[m/s]$$

This shows that the calculated velocity is actually greater than the real velocity by a factor of $\frac{15}{13}$.

In general the calculated linear velocity is scaled by a linear scale factor

$$\text{scale factor in linear velocity} = \frac{\text{assumed height above perceived object}}{\text{real height above perceived object}}$$

This would in fact mean that when flying over trees the linear velocities will appear larger if the variations in terrain elevations are not compensated for. However the calculated angular velocity is not affected by a change in elevation.

When calculating the velocity from OF the terrain has been used to find the velocity. Thus only OF vectors that belongs to the terrain should be used. Moving objects are not moving in the same manner as the terrain with respect to the camera. Therefore using OF vectors from moving objects might cause the calculated velocity to be erroneous. A simple outlier detector developed in the project report is utilized to remove erroneous OF vectors.

Chapter 3

State Estimation With a Nonlinear Observer

Navigation can be defined as the problem of determining the position, velocity or attitude of an object in motion by combining measurements from several sensors [25]. Nonlinear observers and the well known Kalman filter [31] are possible solutions for the navigation problem. Nonlinear observers have received a considerable amount of attention in recent years and is a way to estimate the unknown parameters of interest. This thesis focuses on a nonlinear observer for fixed-wing UAVs that estimates position, velocity, attitude and gyro bias with measurements from different sensors. The Kalman filter is an expensive estimator with respect to computationally complexity. Furthermore the tuning of the Kalman filter is time consuming. Nonlinear observers on the other hand are less demanding and suits systems with low computational power, such as the on-board computer in a UAV. The nonlinear observers often can be proved to be globally or semi-globally stable which in theory guarantees convergence for the estimates. This is especially an advantage for nonlinear systems since the Extended Kalman Filter cannot be proven to be optimal in the same way as the linear Kalman filter. Exponential stability also increases the resistance to environmental disturbances and uncertain initial values [21].

This chapter will present a nonlinear observer which is proven to be semi-globally exponentially stable. The observer uses inertial measurements of acceleration, angular velocity and GNSS (GPS) measurement of position and velocity. In addition both a video camera and a magnetometer can be used, but not in combination. Therefore the equations with a magnetometer and a video camera will be presented. Both versions of the nonlinear observer will be evaluated through an experiment.

Altimeter and inclinometer measurements of the roll and pitch angle are necessary if the camera is used. Furthermore the results from the camera-based observer is used as a basis for the paper submitted and accepted for ICUAS'15. The paper is attached in Appendix A, but the main findings will be presented later in this thesis as well. The following topics will be the main focus of this chapter:

- Section 3.1 provides an introduction to nonlinear observers. Some history will be presented as well as related work. Important definitions for the rest of the chapter will also be given in this section.
- Section 3.2 presents the equations for the nonlinear observer and the necessary measurements. Furthermore a theorem regarding the stability of the observer is given.
- Section 3.3 contains information about the adjustable parameters in the observer.

3.1 Introduction

Nonlinear observers have been studied for the last two decades for navigation purposes. The first nonlinear observer for attitude estimation was presented in [44]. It was extended in [52] to include estimation of gyro bias and linear velocity by utilizing GPS measurements. The nonlinear observer studied in this thesis uses an attitude observer first proposed in [36]. It uses inertial measurements to find the attitude, and global stability results were proved for the observer (attitude) error. The attitude observer was developed further in [22] to get rid of some of the assumptions made in [36]. It was expanded to to estimate position, velocity, attitude and gyro bias in [25]. In addition global exponential convergence was proved for the observer. The observer utilizes measurements from GNSS, inertial measurements and a magnetometer. This observer estimated attitude through a rotation matrix between the body-fixed frame and NED, and neglected the rotation of the earth (local navigation). [23] simplified the attitude estimation to use quaternions instead of the rotation matrix and considered the effect of the rotation of the earth.

The work in this thesis is closely related to the articles mentioned above. [21] uses the observer presented in [25], but replaced the magnetometer measurement with measurements from a video camera. The observer with the video camera is the main focus of this chapter. However the same observer with magnetometer instead of camera is also examined.

The estimates can be represented in other reference frames than the measurements. Therefore the different reference frames need to be defined. Two coordinate frames are of interest in this chapter. That is NED (earth-fixed) which can be assumed to be inertial if the rotation of the earth is neglected. This is often assumed in local navigation applications since the deviation caused by the rotation of the earth is limited locally, and in a limited time period.

Definition 3.1 A vector represented in NED is given with superscript n. A position vector in NED is represented as $\mathbf{p}^n$. The gravity vector $\mathbf{g}$ expressed in NED is $\mathbf{g}^n = [0, 0, 9.81]^T$.

The other reference frame of interest is the body-fixed frame which is attached to the UAV. The body-fixed reference frame is illustrated in Figure 3.1. The rotation between the body-fixed frame and NED is given by the Euler angles. The rotation matrix between these frames are given in [20, Chapter 2].

Definition 3.2 A vector represented in the body-fixed frame is given with superscript b. The body-fixed linear velocity is represented as $\mathbf{v}^b$.

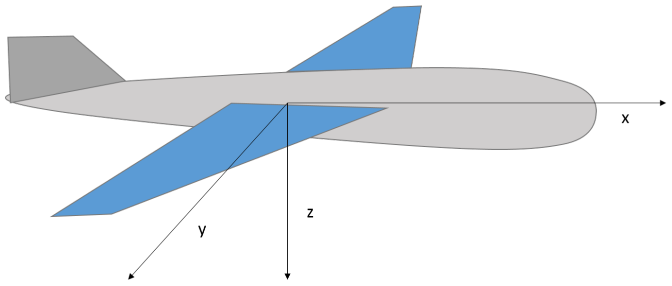


Figure 3.1: The body-fixed reference system.

Matrices are written as bold upper-case letters, vectors as bold lower-case letters and scalars as non-bold letters. The skew-symmetric matrix $\mathbf{S}$ of a vector $\mathbf{x} \in \mathbb{R}^3$ is defined in Section 2.4.1. The opposite operation is denoted vex($\mathbf{x}$) which means that vex($\mathbf{S}(\mathbf{x})) = \mathbf{x}$. The skew-symmetric part of a square matrix $\mathbf{A}$ is expressed as $\mathbb{P}_a(\mathbf{A}) = \frac{1}{2}(\mathbf{A} - \mathbf{A}^T)$. Sat($\mathbf{A}$) is an element-wise saturation of a vector or matrix $\mathbf{A}$ in the interval [-1,1]. The parameter projection Proj$(\cdot, \cdot)$ is defined in [25] as

$$\text{Proj}(\hat{\mathbf{b}}^b, \boldsymbol{\tau}) = \begin{cases} \left(\mathbf{I} - \frac{c(\hat{\mathbf{b}}^b)}{\|\hat{\mathbf{b}}^b\|^2} \hat{\mathbf{b}}^b \hat{\mathbf{b}}^{bT}\right) \tau, & \|\hat{\mathbf{b}}^b\| \geq L_b,\ \hat{\mathbf{b}}^{bT} \boldsymbol{\tau} > 0 \\ \boldsymbol{\tau}, & \text{otherwise} \end{cases}$$

where $c(\hat{\mathbf{b}}^b) = \min\{1, (\|\hat{\mathbf{b}}^b\|^2 - L_b^2)/(L_{\hat{\mathbf{b}}^b}^2 - L_{b^b}^2)\}$ and $\| \cdot \|$ denotes the standard euclidean norm.

3.2 A Nonlinear Observer for Attitude, Position, Velocity and Gyro Bias Estimation

As mentioned in Section 3.1 an observer that estimates position, velocity, attitude and gyro bias is of interest. The necessary measurements for this to be possible are

- NED position $\mathbf{p}^n$ from GPS.
- NED velocity $\mathbf{v}^n$ from GPS.
- Angular velocity in body relative to NED with bias $\boldsymbol{\omega}^b_{imu} = \boldsymbol{\omega}^b_{b/n} + \mathbf{b}^b$.
- Acceleration from accelerometer in body $\mathbf{a}^b_{imu}$. This measurement is related to the NED by the rotation matrix $\mathbf{R}^n_b$, later referred to as $\mathbf{R}$.
- Camera computed body-fixed velocity $\mathbf{v}^b_{b/n}$ through OF. This measurement can be replaced by a body-fixed magnetometer measurement $\mathbf{m}^b$ related to the earth magnetic field in NED through $\mathbf{R}$.
- Altitude from an altimeter (or GPS) and roll and pitch angles from an inclinometer to compute the body-fixed velocity from the OF vectors.

Furthermore two assumptions are necessary for the observer to guarantee stability.

Assumption 3.1 The gyro bias $\mathbf{b}^b$ is constant and bounded by a known upper limit L_b such that $\|\mathbf{b}^b\| < L_b$

The first assumption depends on the gyro bias. The upper limit can often be found in the data sheet of the gyro. Furthermore the gyro bias is often constant for limited time periods and thus the assumption is assessed to be reasonable.

Assumption 3.2 The body-fixed velocity (or magnetometer measurement) computed from the OF vectors and acceleration are linearly independent.

The second assumption is related to the reference vectors used to find the attitude. Since the accelerometer measurement in NED contains the gravity vector, the only possibility for the velocity and acceleration to be linearly dependent is with motion straight up or down. This can be the case for helicopters, but not fixed-wing UAVs since they always have forward motion. Therefore the second assumption is also reasonable.

3.2.1 Observer Equations

This section presents the observer equations. The equations for the observer with camera and magnetometer are almost similar. However the equations for both versions of the observer will be presented for clarity. Furthermore the differences will be highlighted.

Camera-Aided Nonlinear Observer

The observer with the camera is illustrated in Figure 3.2. The equations for the nonlinear observer with camera are

$$\Sigma_1 \begin{cases} \dot{\hat{\mathbf{R}}} = \hat{\mathbf{R}}\mathbf{S}(\boldsymbol{\omega}^b_{imu} - \hat{\mathbf{b}}^b) + \sigma \mathbf{K}_P \hat{\mathbf{J}} \\ \dot{\hat{\mathbf{b}}}^b = \mathrm{Proj}(\hat{\mathbf{b}}^b, -k_I \mathrm{vex}(\mathbb{P}_a(\hat{\mathbf{R}}_s^T \mathbf{K}_P \hat{\mathbf{J}}))) \end{cases} \tag{3.1}$$

$$\Sigma_2 \begin{cases} \dot{\hat{\mathbf{p}}}^n = \hat{\mathbf{v}}^n + \mathbf{K}_{pp}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{pv}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \dot{\hat{\mathbf{v}}}^n = \hat{\mathbf{a}}^n + \mathbf{g}^n + \mathbf{K}_{vp}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{vv}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \dot{\boldsymbol{\xi}} = -\sigma \mathbf{K}_P \hat{\mathbf{J}} \mathbf{a}^b + \mathbf{K}_{\xi p}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{\xi v}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \hat{\mathbf{a}}^n = \hat{\mathbf{R}} \mathbf{a}^b_{imu} + \boldsymbol{\xi} \end{cases} \tag{3.2}$$

$$\mathrm{OF} \begin{cases} \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} = - \begin{bmatrix} \mathbf{M}(f, \phi, \theta, c^n_z, r_1, s_1) \\ \vdots \\ \mathbf{M}(f, \phi, \theta, c^n_z, r_N, s_N) \end{bmatrix}^+ \begin{bmatrix} \dot{r_1} \\ \dot{s_1} \\ \vdots \\ \dot{r_N} \\ \dot{s_N} \end{bmatrix} \end{cases} \tag{3.3}$$

$$\hat{J} \begin{cases} \hat{\mathbf{J}}(\mathbf{v}^b_{b/n}, \quad \hat{\mathbf{v}}^n, \mathbf{a}^b, \hat{\mathbf{a}}^n, \hat{\mathbf{R}}) := \hat{\mathbf{A}}_n \mathbf{A}_b^T - \hat{\mathbf{R}} \mathbf{A}_b \mathbf{A}_b^T \\ \mathbf{A}_b := [\mathbf{a}^b,\ \mathbf{a}^b \times \mathbf{v}^b_{b/n},\ \mathbf{a}^b \times (\mathbf{a}^b \times \mathbf{v}^b_{b/n})] \\ \hat{\mathbf{A}}_n := [\hat{\mathbf{a}}^n,\ \hat{\mathbf{a}}^n \times \hat{\mathbf{v}}^n,\ \hat{\mathbf{a}}^n \times (\hat{\mathbf{a}}^n \times \hat{\mathbf{v}}^n)] \end{cases} \tag{3.4}$$

The observer can be divided in two parts. Σ_1 in (3.1) is the attitude estimator. It is based on biased angular velocity measurements from the IMU, the velocity measurement from the camera and accelerometer measurements from IMU (through $\hat{\mathbf{J}}$). Σ_1 estimates the rotation matrix $\mathbf{R}$ (between NED and the body-fixed frame) and the gyro bias $\hat{\mathbf{b}}^b$. The matrix $\hat{\mathbf{J}}$ in (3.4) is an injection term constructed from the camera velocity measurement used to stabilize the attitude estimates. $\mathbf{K}_p$ is a symmetric positive matrix with gains for the injection term $\hat{\mathbf{J}}$. $\sigma \geq 1$ is a constant adjusted to achieve stability. K_I is a positive scalar which determines how fast the estimate of the gyro bias is allowed to change. $\hat{\mathbf{R}}_s$ is the saturation sat($\hat{\mathbf{R}}$).

Σ_2 in (3.2) estimates the translational motion. That is the position and velocity in NED, the acceleration in NED and $\dot{\boldsymbol{\xi}}$ is a term to correct the position and velocity

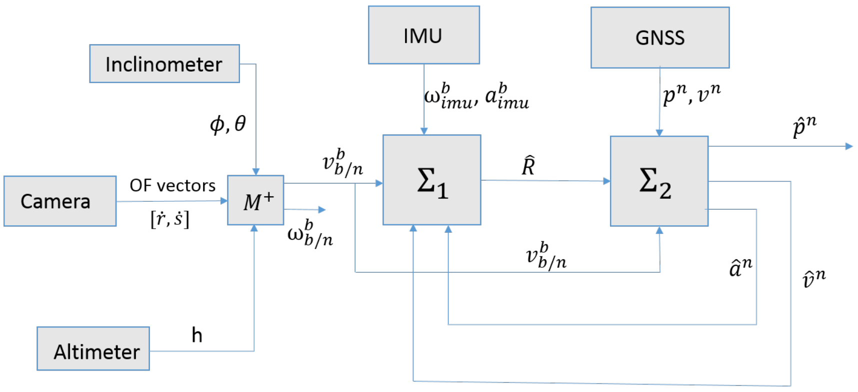


Figure 3.2: Observer with body-fixed velocity from camera.

estimates through the acceleration $\hat{\mathbf{a}}^n$. $\mathbf{K}_{pp}$, $\mathbf{K}_{pv}$, $\mathbf{K}_{vp}$, $\mathbf{K}_{vv}$, $\mathbf{K}_{\xi p}$ and $\mathbf{K}_{\xi v}$ are diagonal matrices used to tune the translational part of the observer.

OF in (3.3) is the calculation of linear and angular velocity from OF vectors described more closely in Section 2.4. $\mathbf{v}^b_{b/n}$ is the linear velocity computed from the camera by OF and considered as a measurement in the observer. The angular velocity calculated from the OF vectors are not used since the angular velocity is measured with the gyro. Only the direction of the linear velocity is of interest. Therefore the linear velocity used in (3.4) is normalized and the same is true for the other vectors in $\hat{\mathbf{J}}$.

Feedback of Roll, Pitch and Altitude to Find Velocity From OF

In Section 2.4 a transformation between OF, roll, pitch and altitude and velocity was derived. It was mentioned that the transformation required measurements of the roll and pitch angles from an inclinometer and altitude from GPS or altimeter. However estimates of the altitude, roll and pitch exist and they are in general more accurate than measurements of the same states. Therefore a possibility for calculating velocity from OF in (3.3) is to use the estimates in the calculation instead of measurements. This is illustrated in Figure 3.3. By using feedback of the estimates, (3.3) is changed to

$$\text{OF}\left\{\begin{bmatrix}\mathbf{v}^b_{b/n}\\ \boldsymbol{\omega}^b_{b/n}\end{bmatrix} = -\begin{bmatrix}\mathbf{M}(f,\hat{\phi},\hat{\theta},\hat{c^n_z},r_1,s_1)\\ \vdots\\ \mathbf{M}(f,\hat{\phi},\hat{\theta},\hat{c^n_z},r_N,s_N)\end{bmatrix}^+\begin{bmatrix}\dot{r_1}\\ \dot{s_1}\\ \vdots\\ \dot{r_N}\\ \dot{s_N}\end{bmatrix}\right. \tag{3.5}$$

while the rest of the equations are equal.

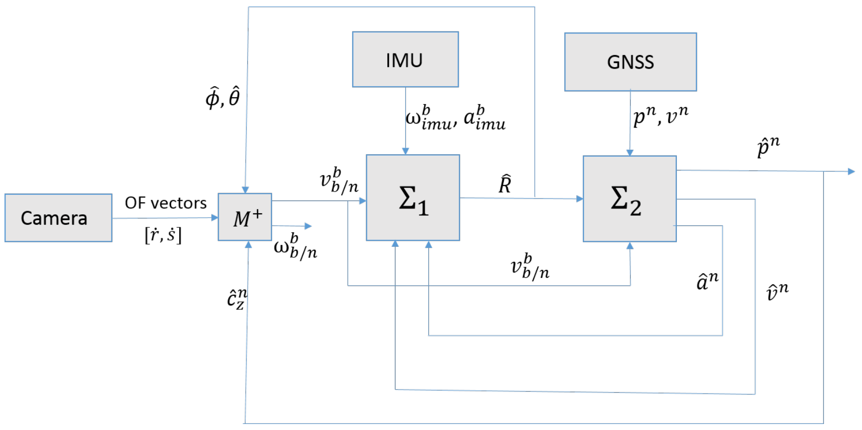


Figure 3.3: Observer with camera and feedback.

Magnetometer-Aided Nonlinear Observer

The velocity computed from the camera by OF can be replaced by magnetometer measurements. This is fairly simple and only the injection term $\hat{\mathbf{J}}$ needs to be changed. The velocity $\mathbf{v}^b_{b/n}$ in $\mathbf{A}_b$ are replaced by the body-fixed magnetometer measurement $\mathbf{m}^b$. Furthermore the estimate of $\hat{\mathbf{v}}^n$ in $\hat{\mathbf{A}}_n$ is replaced by the local magnetic field in NED $\mathbf{m}^n$. The equations for the observer with a magnetometer are then

$$\Sigma_1 \begin{cases} \dot{\hat{\mathbf{R}}} = \hat{\mathbf{R}}\mathbf{S}(\boldsymbol{\omega}^b_{\text{imu}} - \hat{\mathbf{b}}^b) + \sigma\mathbf{K}_P\hat{\mathbf{J}} \\ \dot{\hat{\mathbf{b}}}^b = \text{Proj}(\hat{\mathbf{b}}^b, -k_I\text{vex}(\mathbb{P}_a(\hat{\mathbf{R}}_s^T\mathbf{K}_P\hat{\mathbf{J}}))) \end{cases} \tag{3.6}$$

$$\Sigma_2 \begin{cases} \dot{\hat{\mathbf{p}}}^n = \hat{\mathbf{v}}^n + \mathbf{K}_{pp}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{pv}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \dot{\hat{\mathbf{v}}}^n = \hat{\mathbf{a}}^n + \mathbf{g}^n + \mathbf{K}_{vp}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{vv}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \dot{\boldsymbol{\xi}} = -\sigma\mathbf{K}_P\hat{\mathbf{J}}\mathbf{a}^b + \mathbf{K}_{\xi p}(\mathbf{p}^n - \hat{\mathbf{p}}^n) + \mathbf{K}_{\xi v}(\mathbf{v}^n - \hat{\mathbf{v}}^n) \\ \hat{\mathbf{a}}^n = \hat{\mathbf{R}}\mathbf{a}^b_{\text{imu}} + \boldsymbol{\xi} \end{cases} \tag{3.7}$$

$$\hat{J} \begin{cases} \hat{\mathbf{J}}(\mathbf{m}^b, \quad \hat{\mathbf{m}}^n, \mathbf{a}^b, \hat{\mathbf{a}}^n, \hat{\mathbf{R}}) := \hat{\mathbf{A}}_n\mathbf{A}_b^T - \hat{\mathbf{R}}\mathbf{A}_b\mathbf{A}_b^T \\ \mathbf{A}_b \quad := [\mathbf{a}^b,\ \mathbf{a}^b \times \mathbf{m}^b,\ \mathbf{a}^b \times (\mathbf{a}^b \times \mathbf{m}^b)] \\ \hat{\mathbf{A}}_n \quad := [\hat{\mathbf{a}}^n,\ \hat{\mathbf{a}}^n \times \mathbf{m}^n,\ \hat{\mathbf{a}}^n \times (\hat{\mathbf{a}}^n \times \mathbf{m}^n)] \end{cases} \tag{3.8}$$

The only other consideration is the adjustable parameters for the observer. Some of the gains should possibly have different values for the magnetometer in comparison with the camera. The nonlinear observer with magnetometer instead of camera is displayed in Figure 3.4.

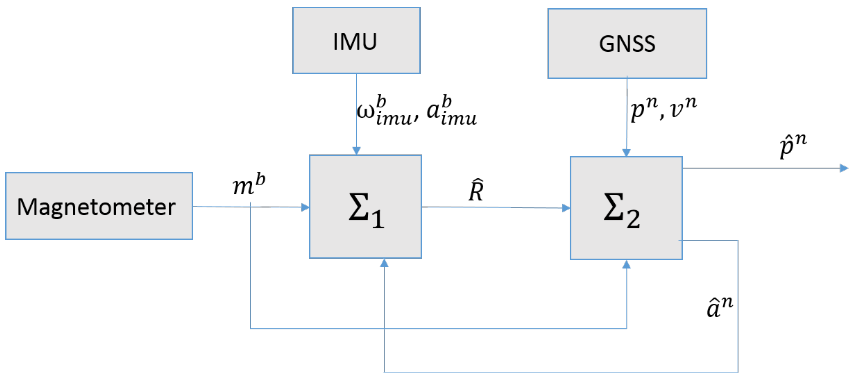


Figure 3.4: Observer with magnetometer.

3.2.2 Stability of the Nonlinear Observer

The stability properties of the nonlinear observers are often more conclusive than the EKF since global stability properties can be proved. The stability properties for the observer with camera and magnetometer will be given in this section.

Theorem 3.1 The error dynamics of the nonlinear observer in 3.2.1 with body-fixed velocity measurements from the camera are uniformly semi-globally exponentially stable.

Proof 3.1 The proof of the observer with the camera is given in [21].

Theorem 3.2 The error dynamics of the nonlinear observer in 3.2.1 with body-fixed velocity measurements from the camera replaced by magnetometer measurements are globally exponentially stable.

Proof 3.2 The proof is given in [25].

The stability proofs show exponential convergence rate for the estimates. This in theory guarantees that the estimates converge to the true value in finite time. The same properties cannot be proven for the EKF and is one of the advantages for nonlinear observers. However it is worth noting that the proofs only guarantee that there exist tuning values that ensure the stability properties. In practice it is necessary to find values that works for the application of interest, and it might be hard to find the best values.

It is important to notice that stability of the nonlinear observer with feedback of roll, pitch and altitude to the M-matrix is not proved. A proof for the stability is not the focus of this thesis and the feedback will only be tested in practice. Nevertheless it is highly probable that the stability properties are equally good, especially after convergence for the estimates.

3.3 Parameter Tuning

This section discusses the adjustable parameters in the nonlinear observer in Section 3.2.1. The best values can vary in different conditions. Therefore the purpose of this section is to provide some knowledge about what the gains affect. The nonlinear observer with camera and magnetometer both consist of the same tuning parameters and they will not be described separately.

Tuning of the attitude part of the observer

The attitude part of the observer is given by (3.1). The adjustable parameters are σ, $\mathbf{K}_p$ and k_I. $\mathbf{K}_p$ is a 3×3 matrix and normally diagonal (and symmetric). For the stability proof to hold σ needs to be chosen according to $\sigma > 1$. $\mathbf{K}_p$ decides the influence of the injection term and how much the gyro is trusted compared to the body-fixed velocity measurement from the camera or equivalently magnetic field from magnetometer. Since the gyro normally runs at a much higher frequency than the camera, the gyro decides the high frequency motion and the camera decides the main trend. Higher gains on the diagonal let the camera have higher influence. Different values on the diagonal give different weighting to the different parts of the rotation matrix. Therefore decreasing the diagonal elements in the dimensions where the camera provide inaccurate measurements might be beneficial. k_I is used to update the gyro bias and decides how fast the estimate is allowed to change. Increasing the value increases the rate of change.

Tuning the translational motion part of the observer

The translational part of the observer consists of six adjustable matrices. That is $\mathbf{K}_{pp}$, $\mathbf{K}_{pv}$, $\mathbf{K}_{vp}$, $\mathbf{K}_{vv}$, $\mathbf{K}_{\xi p}$ and $\mathbf{K}_{\xi v}$. They are often a scalar multiplied by the identity matrix. $\mathbf{K}_{pp}$ decides how fast the position measurement from GPS is allowed to correct the position estimate. $\mathbf{K}_{pv}$ decides how fast the position estimate is corrected by the measured velocity from the GPS. $\mathbf{K}_{pp}$ is often much larger than $\mathbf{K}_{pv}$ since the position measurement should affect the position estimate more than the velocity measurements. By the same logic $\mathbf{K}_{vp}$ and $\mathbf{K}_{vv}$ are used to correct the velocity estimate. However $\mathbf{K}_{vv}$ should be larger than $\mathbf{K}_{vp}$ since the velocity measurement should affect the velocity estimate more than the position measurement. $\boldsymbol{\xi}$ is used to correct the estimated acceleration. $\mathbf{K}_{\xi p}$ and $\mathbf{K}_{\xi v}$ are used to correct the estimated acceleration. These values are often smaller than the other adjustable matrices in the translational part of the observer.

Chapter 4

Object Detection and Tracking

Object detection and tracking are two of the most important topics in computer vision. Object detection is the process of automatically detecting objects of importance with respect to some criterion in an image. Object detection can for instance be used to find humans in an image [15]. Object tracking is the process of generating a trajectory for an object detected in several image frames. Therefore it is important to uniquely describe objects in order to identify common objects in different images and match objects in different images together. This is called object classification. Several important research areas require automatic detection and tracking of objects. Surveillance [28], search and rescue applications (human body detection and geo-localization) [17, 43], collision avoidance [1], driver assistance [26], face detection [45] and vehicle tracking [11] are all common examples.

This chapter looks into the problem of object detection, classification and tracking in images captured from an airborne fixed-wing UAV. This can be extremely challenging because the projected area displayed in the image will change rapidly because of the large velocity of the UAV. The possibility of using OF for object detection and tracking will be the main focus in this chapter which contains the following topics:

- Section 4.1 provides an introduction to object detection and tracking. This includes related work and important definitions for the rest of the chapter.
- Section 4.2 contains image segmentation and object detection. How can objects be detected automatically in images? This section includes a description of some important methods in addition to development of a new algorithm that utilizes OF.

- Section 4.3 contains object classification. How can objects from different image frames be associated? This includes a description of several object classification methods.
- Section 4.4 contains development of a tracking system. How can position trajectories be generated for objects detected in several image frames? The tracking system utilizes the moving object detection algorithm.
- Section 4.6 contains an approach for transforming the tracking estimates in the image plane to position and velocity in NED.

4.1 Introduction to Object Detection and Tracking

Cameras, including both infra-red cameras and regular color cameras, can gather an enormous amount of information about the flight area. In this thesis it is assumed that the images contain information about the terrain below the UAV. In other words, the camera points towards the ground. This is illustrated in Figure 4.1. The images captured of the ground can be used for object detection and tracking. One of the greatest challenges in object detection is to extract the same kind of information that a human operator would do in real time. This is crucial to solve before computer vision can replace human operated surveillance tasks.

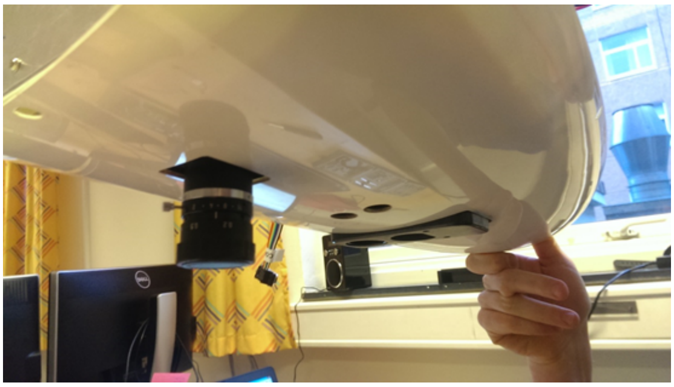


Figure 4.1: Camera placement in the UAV. It points straight towards the ground as long as the UAV has zero roll and pitch.

There are a lot of issues when object detection and tracking are performed on images captured from a UAV. The UAV moves with large velocity. Since a finite amount of images are captured each second, the velocity and altitude determines how much consecutive images change. Larger altitude leads to less change. Unfortunately larger altitudes also causes each object in the image to be smaller and harder to detect and distinguish. An example of a image captured at a height of approximately 140 meters above ground level is given in Figure 4.2. The objects in the image are in fact large cars and people. The cars are not very easy to see in the image because of the altitude, but also because of the white color. This illustrates the problem an algorithm might have to find objects, and especially smaller objects than cars.

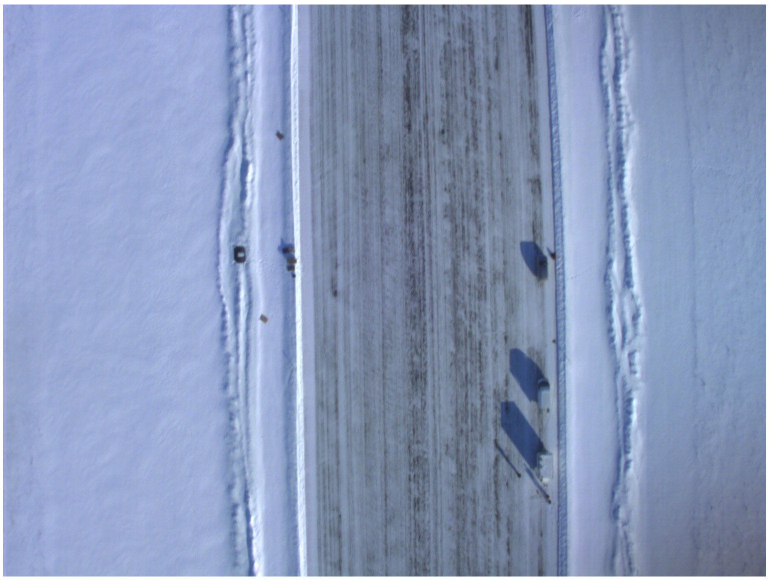


Figure 4.2: Image captured at an altitude of approximately 140 meters. Two large cars and a trailer are visible on the right hand side of the runway.

For tracking purposes it is beneficial to have the object of interest inside the field of view of the camera during the tracking period. A UAV moving with large velocity will capture images where the object of interest is visible for a limited amount of image frames only. This is especially an issue for objects moving in the opposite direction of the UAV. In many situations only moving objects are of interest. Slowly moving objects might be hard to detect, since the static parts of the image will change almost equally much as the object. Therefore it might hard to separate slowly moving objects from noise. The static part of the image is defined as

Definition 4.1 The static part of an image, also referred to as the background, is the parts of the image that belong to the terrain.

In practice this means that the static part is every pixel that belongs to something at rest, like the terrain. However light conditions or noise might still change the pixel.

The OF is a measure of the motion field between consecutive images (Section 2.2.1). Therefore, in theory, it should be possible to separate moving objects from the background with OF. For this to work OF vectors must be present on the moving objects. A dense OF method should be used in order to evaluate every single pixel in the image. This removes the chance of missing OF vectors on the moving objects. Sparse methods might miss some objects since objects might be located where the OF is not calculated. However the dense methods are often inaccurate [37] or too computationally expensive. Therefore a sparse approach will be presented in the next section.

4.2 Moving Object Detection

Object detection can be defined as

Definition 4.2 Object detection is the art of detecting interesting features or areas in an image with respect to a set of criteria.

One might ask what an object is? An object could actually be anything in an image. It could be every place with a red color, every shadow, every human face or a boat. The main concern is to find only the objects of interest and this might be the main reason for the many different existing approaches. The most suitable method for object detection depends on the application. If the main goal is to detect and track humans, the object detector should utilize the unique characteristics of a human and look for familiar shapes. If the goal is to track every moving object in a scene, the motion field of the image might be the best way to detect objects. The first part of this section will classify and say something about different approaches for object detection. The second part will focus on object detection from UAVs.

4.2.1 Related Work

Every object detection algorithm utilizes information given in the image. It could be the intensity value (color intensity values in color images) or the gradient of the intensity value. The difference lies in the way the raw image information is used. Some methods might benefit from image smoothing [48, Chapter 5] before object detection, while some methods use the image directly. The goal of this section is to describe different object detection approaches and say something about the most suitable application for each approach. In this way the reader will be provided with knowledge that explains and highlights the choices made later in this chapter. For a more comprehensive study the reader should investigate [55].

The process of looking for objects in an image can be defined as a segmentation process. Motion segmentation, which is the process of finding moving objects, is particularly interesting for this thesis. For a static camera, background segmentation is the most common way to locate moving objects [55]. The purpose is to generate a model of the fixed background and subtract the background from the current image. This is called image differencing [48]. With this approach the moving objects will be the only thing left if the background is static. This is however not appropriate for UAV applications because the UAV moves and a background model can not be used. Another possibility is to look for differences in consecutive image frames. This is also not appropriate for UAV since every pixel will change because of the movement of the UAV. Adaptive background models, which continuously updates the background model, might work for small velocities, but the fixed-wing UAV velocity is too large for this to work. Mean shift segmentation [13] is a famous segmentation algorithm.

Point detectors locate pixels in an image with a clear texture with respect to its environment. Objects of just a single pixel might be detected with the necessary texture. Famous point detectors include the Harris corner and edge detector [27], SIFT [33], Haar-like features [32] and the KLT detector [46]. Point detectors might be one of the best ways to detect objects from a UAV since the detection is performed independently of the previous images and the background. They simply look for objects in every image. The main challenge with these methods is their repeatability which can be stated as the ability to detect the objects for consecutive images.

Supervised learning [55] uses a training set to detect objects. The detection consists of a training stage where the detector learns different object views automatically from a set of examples. This approach needs a priori information to work correctly. More specifically information about the appearance of the objects of interest are necessary. So for this to work it is necessary to know if boats or humans should be detected for instance. Supervised learning makes it possible to separate the same types of objects. If boats are of interest, it is possible to distinguish large vessels from smaller vessel. Therefore, if a single type of boat should be detected a detection algorithm based on the features of this type of vessel can be trained. [53] uses supervised learning to detect walking humans. A method for supervised learning is the support vector machine [48]. Supervised learning is not appropriate in this case since all moving objects are of interest, and it is naturally impossible to find training sets for every possible object. A training set for a human might consists of up to ten thousand images where at least half of them have a human in the image.

OF can in theory be used as a segmentation approach even in the case of camera motion. For a flat static background the flow field should be homogeneous. Therefore moving objects might be detected by looking for image patches where the OF differs from the neighbourhood. Important objects might be very small because of the altitude of the UAV. An example of OF based segmentation is given in [38]. [40] presents a method of object detection from a UAV for collision avoidance. [16] presents a method of object tracking through motion segmentation and OF. However a fixed camera is assumed.

4.2.2 Detection of Moving Objects in Images Captured From a UAV

How can reliable object detection be achieved from a fixed-wing UAV moving at large velocities? This is an important question for many applications. Search and rescue applications in areas where there are limited accessibility or too dangerous for humans, might require reliable human detection. This thesis focuses on detection of moving objects. There are some issues that need to be examined. Most importantly, no prior knowledge of the examined area is available. The objects of interest can be humans, boats, animals or other vehicles. Secondly, the projected area changes rapidly which excludes simple background segmentation methods.

So which methods are left then? Point detectors and dense OF methods are still not excluded. However, as mentioned before, the dense OF algorithms are prone to errors. Therefore dense OF algorithms are considered to be too unreliable. Three possible approaches will be described in this section. One of them is implemented and described thoroughly. Before the different methods are described the following definition is necessary:

Definition 4.3 Theoretical flow is defined as the inverse transformation of (2.25). Thus it is the OF calculated by the M-matrix from Section 2.4 given roll, pitch, altitude, linear velocity, angular velocity and a pixel position (r,s). It results in OF vectors where all features are assumed to be at rest.

It is possible to compare the measured OF from the images with the theoretical flow which is given by the states of the UAV, and look for differences.

[12] investigates the possibility of using OF for object detection from a moving airborne platform. Their approach is to compensate for the UAV motion and use OF to find moving objects after the compensation. Each image is transformed to a reference frame. A possible approach is to find the geometric transform between two successive images. The second image can be transformed to the reference frame of the first image. OF can be used to find moving regions since the OF should be zero elsewhere. This approach requires an accurate geometric transformation and images with small amount of noise. Background segmentation could also be used instead of OF, but also requires an accurate transform.

[49] uses another approach for object detection from a moving robot (not airborne). The 6-DOF motion of the robot is calculated. The theoretical flow field can be calculated based on the 6-DOF motion. The difference between theoretical and measured flow can be calculated and the difference should be non-zero in every location where a moving object is present.

The third approach, which is proposed in this thesis, is very similar to the second approach. A point detector like SURF [3] or SIFT [33] is used to find features in the image. These detectors find features independently of their motion. Therefore static features have the same probability of being detected as moving features. OF can be measured at the position of the extracted features and compared with the theoretical flow at these positions. Moving objects can then be located by using the statistical properties of the difference between theoretical and measured OF for every OF vector in the image. The noise level of the measured OF in UAVs are much larger than for a robot at rest, and thus it is not appropriate to just look for non-zero differences. An algorithm based on the difference and simple statistical properties of the OF vectors is described more closely in the next section.

4.2.3 An Algorithm for Moving Object Detection

This section is going to describe an algorithm that have been developed in this thesis to extract only the moving objects of an image. It is one of the contributions of this thesis. In the problem formulation the following assumption was made

Assumption 4.1 Only one moving object appears in each image.

Assumption 2.1 is used to develop an algorithm for extracting a moving object from an image. The assumption results in OF vectors where most of them belongs to the background. Therefore the standard deviation of the difference between the theoretical and measured OF should be very low. This is because the majority of the OF vectors belongs to features at rest.

Even though Assumption 2.1 states that only one moving object is present in each image, there are no restrictions on the number of objects that are classified as moving. Therefore the algorithm should be able to find more than one moving object if it exists. The algorithm is illustrated in Figure 4.3 and described more closely in Table 4.1.

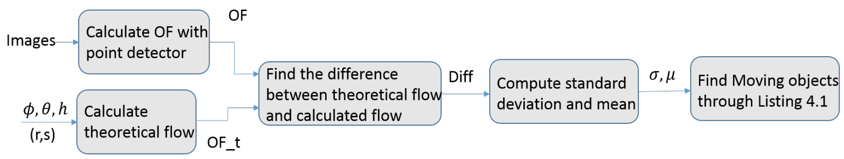

Images
Calculate OF with point detector
OF
φ, θ, h
(r,s)
Calculate theoretical flow
OF_t
Find the difference between theoretical flow and calculated flow
Diff
Compute standard deviation and mean
σ, μ
Find Moving objects through Listing 4.1

Figure 4.3: Flow chart for algorithm for moving object detection

For this algorithm to be reliable the following assumption is made:

Assumption 4.2 The feature detector is able to find at least one feature on the moving objects of interest.

To the authors knowledge this algorithm is the first one that combines a point detector and theoretical flow to find moving objects from a UAV.

Table 4.1: Moving Object Detection Algorithm

#	**Description**
1.	Use a point detector, such as SURF [3] or SIFT [33], to find every feature of interest in two consecutive images. This includes both static and moving objects.
2.	Measure the OF for every feature appearing in both images.
3.	Calculate the theoretical flow according to definition 4.3 at every pixel where the OF is measured in step 2.
4.	Calculate the difference between the theoretical flow and the measured OF by SURF or SIFT.
5.	Find the mean μ and standard deviation σ of the difference between the theoretical flow and measured OF.
6.	Classify every pixel with an OF vector as moving or static by the pseudo-code given in Listing 4.1.
7.	Combine OF vectors that are classified as moving if they are located closely together.

Step six in the algorithm is the most important factor for the performance of the algorithm. The parameters K_σ, K_μ, L_σ and L_μ in Listing 4.1 can be adjusted to decide how large the difference between theoretical flow and computed flow is before it is classified as a moving object. The algorithm can be replaced by more advanced statistical methods to find moving objects. Furthermore results from several image frames could be combined before a decision is made of whether an object is moving or not. However this was considered to be outside of the scope of this thesis.

K_σ is a tuning variable used to decide how large the difference between theoretical and measured flow should be compared to the standard deviation before it is considered to be a moving object. L_σ is an lower limit for the difference before it is considered to be a moving object. In the same manner K_μ decides how large the difference should be compared to the mean difference of all vectors before it is considered to be a moving object. L_μ is a value used for thresholding. If the difference between theoretical and measured flow exceeds the mean difference plus a threshold, the vector is considered to be moving. The implementation of the algorithm is described in Chapter 5 and evaluated in Chapter 7.

Different tests are conducted to check if an OF vector belongs to a moving object. The difference in vertical and horizontal direction are checked independently. Thus a large difference in one direction is enough before it is classified as moving. Furthermore both a pure test of the difference and a test related to the standard deviation of the difference are used to check if the object is moving.

The algorithm has one main limitation. Only moving points can be detected. Therefore it is impossible to say something about what the moving point physically is. It can be a boat or a person. It is also not possible to say something about the shape and size of the object since only a pixel position is identified through the algorithm. Different techniques exist for extracting objects from an area. The complete shape of the object surrounding a moving point could be extracted through thresholding [48, Chapter 6] combined with erosion and dilation [48, Chapter 13]. This is however not the main focus in this thesis and will not be examined further.

Listing 4.1: Pseudo code showing how moving objects are detected

```
%% Classify one vector as static or moving.

% 1. OFdiff_r is the difference between theoretical and
%    measured OF in r for a single OF vector.
% 2. OFdiff_s is the difference between theoretical and
%    measured OF in s for the same vector.
% 3. $\sigma_r$ and $\sigma_s$ are the standard deviation in r and s between
%    measured and theoretical flow, for all OF vectors
% 4. $\mu_r$ and $\mu_s$ are the mean difference between measured
%    and theoretical flow in r and s, for all OF vectors

% Choose gains $K_\sigma$, $L_\sigma$, $K_\mu$ and $L_\mu$ before start.

movingObject = false

if (abs(OFdiff_r) > max($K_\sigma$*$\sigma_r$, $L_\sigma$))
        movingObject = true
end

if (abs(OFdiff_s) > max($K_\sigma$*$\sigma_s$, $L_\sigma$))
        movingObject = true
end

if (abs(OFdiff_r) > max(abs($K_\mu$*$\mu_r$), abs($\mu_r$) + $L_\mu$))
        movingObject = true
end

if (abs(OFdiff_s) > max(abs($K_\mu$*$\mu_s$), abs($\mu_s$) + $L_\mu$))
        movingObject = true
end

return movingObject
```

An example of how the algorithm works is shown in Figure 4.4. Areas where movement is detected are marked with a green rectangle. The black car is detected as a moving object in this particular example. The red arrows are the OF vectors measured by the SIFT algorithm. Every vector not belonging to the car have the approximately same magnitude and direction. The vectors fixed to the car on the other hand have another magnitude and direction because the car moves. This is in fact why the detection algorithm is able to find moving objects.

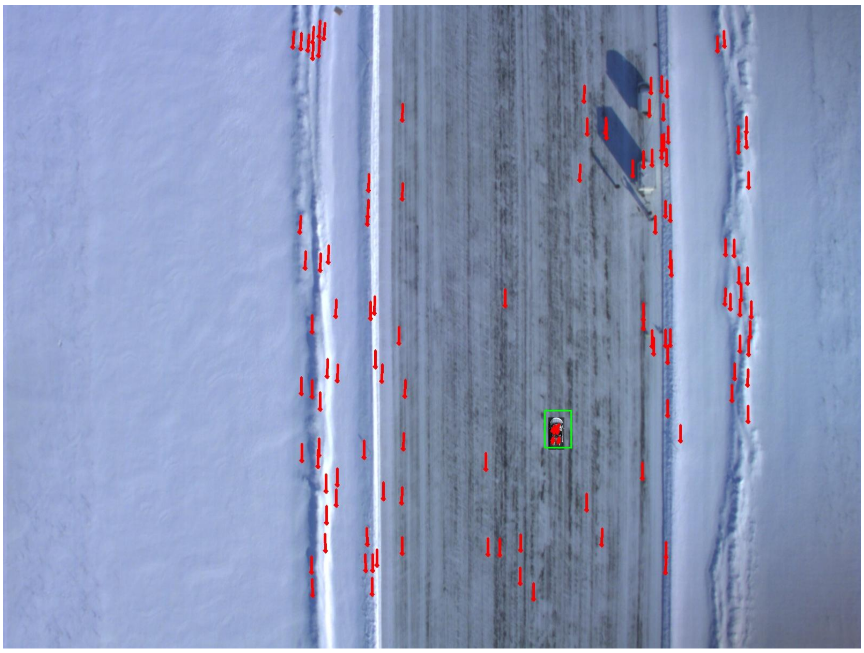


Figure 4.4: Example of the moving object segmentation for a single image.

4.3 Object Classification

Assuming that several objects are detected in different images. How can these objects be separated? How can the same objects in different images be classified as the same object? This can be achieved with object classification.

Definition 4.4 Object classification is the process of assigning a unique description to objects of interest. The description can be a set of features related to each object, such as color, edges, OF, intensity, the intensity gradient or a template of the image. The appropriate choice depends on the application and type of object.

The main focus of this thesis is not object classification, but detection and tracking. Therefore it will not be given that much attention. However a short description of related work and possibilities for UAV applications will be given in the following sections. Furthermore a classifier for the tracking system is chosen since it is necessary to associate measurements of detected objects with tracked objects.

4.3.1 Related Work

There are several approaches for object description and classification. A review is given in [30] and classification is also described in [28], [42] and [55]. SURF [3] and SIFT [33] also provide a way to represent features in addition to detect them. A combination between Mean-shift and SIFT is presented in [57]. Adaboost [50] is a method for creating a strong classifier based on several weak classifiers.

4.3.2 Object Classification for UAV Applications

What is the most important factor for object classification in images captured from a UAV? The most important factor for all classifiers are their ability to uniquely describe each object. The ability to describe the same objects in the same manner, even during different conditions (translation, rotation with respect to the camera, brightness and so on), is especially important. This is because the object might change its illumination when the UAV moves. If a unique classifier is achieved, a similarity measure can be used to connect the same objects from different images. This is crucial for tracking applications.

One important consideration for UAV applications is that the objects are smaller for large altitudes. Therefore less information might be available compared to other applications. Since classification is not the main focus of this thesis, simple approaches are considered.

The first obvious possibility is to use a nearest neighbour approach to connect objects from different images. However this might cause problems for objects entering or leaving the image between successive frames. Nevertheless this might be sufficient for object detection that yields few objects with a minimum distance between each detected object. Furthermore the tracking algorithm might be accurate enough to decide when objects are supposed to leave the images. For this application this approach is not considered to be reliable enough since the number of possible objects to track is unlimited and there are no constraints on the placement of the objects.

Another possibility is to use the classifier in SURF [3] or SIFT [33] directly. This is a simple approach since the SURF and SIFT detector are used to find objects. One disadvantage is the lack of possibility to classify the objects as vehicles or humans for example without adding another description layer for each object. Since several OF vectors might be combined to form a moving object, a single SIFT descriptor is not reliable. The main challenge is that the classifier will be related to a single feature of a moving object. This feature might only be visible for a subset of the images. Therefore a more general description of the object is desired.

For simplicity a template matching approach is chosen for this thesis. A rectangular template (part of the image) is extracted from the areas where moving objects are detected. Every detected moving object gets a template as its classifier. For every new image this template can be matched with areas where movement is detected to check for common objects. It is the same approach as the template matching for OF calculation described in Section 2.3.1. However the results of the template matching is only used to associate objects in this case and not for OF calculation.

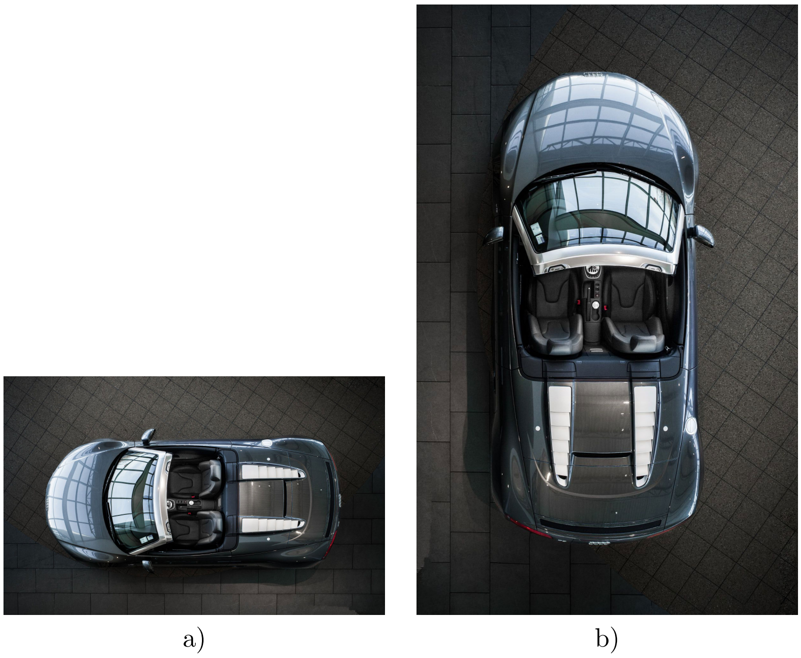


Figure 4.5: a) Template at the beginning b) Template rotated.

There are two main issues with the template matching approach. First of all, the template is an image of the object extracted from a single image. Therefore it is very sensitive to changes in the appearance of the object. Rotations and changes in size might destroy the possibility of associating the same object in different images. This is illustrated in Figure 4.5. The car is equal in images, but if the templates are matched they will not be classified as the same object because of the rotation. However this problem is minimized by updating the template for every new image with match, such that the template at all times represents the latest appearance of the object. The second issue is that the size of the objects is unknown. Therefore it is hard to know how large the template should be. The template should be large enough to easily separate different objects, but small enough to minimize the amount of background in the template. The size of the template can be adjusted in the tracker to find the best performance.

4.4 Object Tracking

If several objects are detected and classified, what is the next step? For many applications it would be interesting to locate objects of interest for a certain time period. This is the art of object tracking.

Definition 4.5 Object tracking is to locate objects through several image frames and generate a time dependent position trajectory for each object.

Object tracking is possible if the same objects can be located in several image frames. Therefore the importance of accurate object classification and the uniqueness of the classifier are easily illustrated. The location of each object in the image can be used to find the position in other coordinate frames such as NED or ECEF if the position of the UAV is known. This is called georeferencing and is described in Section 4.6. Object tracking is of great importance for many applications. Surveillance operations such as vehicle or human tracking, collision avoidance and surveillance of human behaviour are just a few examples of how object tracking is utilized. This section seeks to find an appropriate way to track moving objects detected from a UAV. The first part will present related work and well-known tracking principles. The latter part will focus on developing an approach for tracking of moving objects from a UAV.

4.4.1 Related Work

Tracking can either be performed jointly with detection or done individually. In the joint case the tracking layer can be used to narrow down the search for the object in the next image. This will obviously limit the computational load. However a narrow search will not guarantee that objects entering the image will be detected. Therefore it is most appropriate for applications where a fixed set of objects are tracked. [4] is an example of a multi object tracker. [55] divides existing tracking methods in three categories. That is point trackers, kernel-based trackers and silhouette trackers.

Point trackers consider the detected objects to be a point, for example the centre of the object. This is represented as a pixel in the image. Occlusions, misdetections and objects leaving or entering the image are challenges with point based trackers. Point trackers can either be deterministic or statistical-based. Deterministic methods uses a set of constraints and minimizes an associated cost. The constraints can be based on maximum velocity, displacement or bound on acceleration for example. Statistical methods can use a Kalman filter to track each object [48]. A new instance of the Kalman filter can be created for each object. In this case the position of the detected objects is considered as a measurement with Gaussian noise. Therefore the position trajectory will be the optimal combination of the previous states and the new measurement with respect to minimum variance. [47]

is an example of a point based tracker which in addition uses OF for segmentation. However the background needs to be static.

Kernel tracking is based on the motion of each object. Each object is classified by a simple region (a rectangle or a circle and for example) and the motion of each region is estimated. In the simplest case template matching (see Section 2.3.1) can be used for kernel tracking. OF algorithms can also be used for Kernel tracking. [14] is an example of a Kernel-based object tracker. Another example of a Kernel-based tracker is [41] where OF is used to predict the motion of the objects. However the approach assumes that the objects of interest are manually defined through a simple region in the first image frame.

Silhouette trackers use accurate shape descriptions of the objects. While kernel trackers uses simple geometric forms for object tracking, silhouette trackers use more advanced shapes. An object model is generated and then the tracker tries to locate this shape in the next images. The model can be updated online such that changes in the appearance are accounted for. Silhouette tracking is appropriate when the whole region of the object should be tracked. This can for example be the case for collision avoidance to make sure that the boundary (and not just the centre) avoids collision. [12] and [35] are examples of silhouette based trackers.

4.4.2 Object Tracking for UAV Applications

One of the main approaches described in the previous section needs to be utilized to design a tracker. The objects are tracked in the image plane for simplicity. The position of the UAV can then later be used to compute georeferenced position and velocity trajectories for the tracked objects (Section 4.6). Since every moving object appearing in the scene should be tracked, the objects are not known a priori. Furthermore the objects are not in the scene for long time periods, unless they have the same motion as the UAV. The objects are assumed to be covered by the camera for a limited time-period. Thus silhouette trackers are assessed to be too complicated. They normally need some time to develop a model of a silhouette and this is not available here. Furthermore the silhouette might change rapidly when the UAV turns for example. This might be a problem for kernel-based trackers as well. Furthermore the algorithm for moving object detection results in a point where a moving object should be present. Therefore a point-tracker seems to be the best choice. There are no limits on the motion for the moving objects and a deterministic point tracker is not considered to be appropriate. The obvious alternative is to use the Kalman filter [31]. A new instance of the Kalman filter is created for each detected object. Tracking of objects with a Kalman filter is described in the next section.

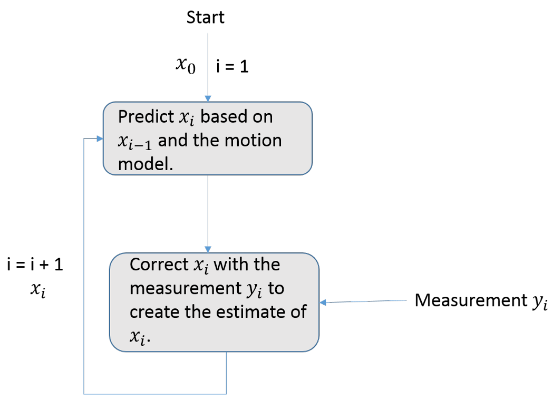
4.4.3 The Kalman Filter

It is assumed that the object only has linear translational motion in the image plane. Thus a linear Kalman filter can be used for tracking. Object tracking in images is a discrete process. This is because the frame rate of video cameras is limited. Thus new images will appear at discrete time instants. A discrete Kalman filter (from now on referred to as KF) is therefore going to be utilized. The KF is a recursive process that tries to estimate a set of states related to the process. Measurements exposed to noise are combined with a motion model to create estimates that are optimal with respect to minimum variance. These estimates are more accurate than the measurements. The optimality criterion holds if the following assumptions are satisfied

Assumption 4.3 The process noise and measurement noise are white and Gaussian. The process noise and measurement noise are uncorrelated. The initial state is Gaussian.

Assumption 4.4 The system is linear and observable.

In addition to the optimality criterion, the estimates are unbiased and asymptotically stable when the assumptions are satisfied. The KF is a combination of two steps. First, the motion model is used to predict the next state based on the the previous state. Then, a correction is performed based on available measurements. The process is displayed in Figure 4.6. If the measurements are lost for a while, the KF can be used to predict the states. The states can be corrected when measurements are available again.



Start
x_0
i = 1
Predict x_i based on x_{i-1} and the motion model.
i = i + 1
x_i
Correct x_i with the measurement y_i to create the estimate of x_i.
Measurement y_i

Figure 4.6: Illustration of correction step in Kalman filtering. x_i is the state that is estimated.

Motion Model

In order to use a KF for tracking a motion model must be identified. Both the position and velocity in the image plane can be measured and should therefore be estimated. The detected objects move around in the real world, and are projected into the 2D image-plane. Motion is represented as a displacement in the image plane. Since the frame rate of the camera is quite large, the following assumption is made

Assumption 4.5 The motion of the detected objects in the image plane is fairly smooth. This means that the change in velocity between two consecutive images is close to zero.

With this assumption the following motion model for the horizontal position r and vertical position s in the image plane can describe the motion of an object

$$\begin{aligned} r[i+1] &= r[i] + u_r[i] + w_r[i] \\ s[i+1] &= s[i] + u_s[i] + w_s[i] \end{aligned} \tag{4.1}$$

where i refers to a discrete time step. w_r and w_s are process noise for the position in the image plane (assumed to be white) and u_r and u_s are velocity in the image plane between consecutive images (OF). By the assumption of smooth motion the model for the velocity in the image plane might be described as

$$\begin{aligned} u_r[i+1] &= u_r[i] + w_{u_r}[i] \\ u_s[i+1] &= u_s[i] + w_{u_s}[i] \end{aligned} \tag{4.2}$$

where w_{u_r} and w_{u_s} are process noise for the velocity in the image plane, assumed to be white.

The moving object detection algorithm locates a pixel position for a moving object. Furthermore the OF of that point is available. This is the velocity between the images. Therefore OF is a measurement of the velocity in the image plane. Both the measured OF and location are contaminated by noise since they are inaccurate. By assuming that the noise is white the following measurement model can be used to describe the measurements

$$\begin{aligned} y_r[i] &= r[i] + v_r[i] \\ y_s[i] &= s[i] + v_s[i] \\ y_{u_r}[i] &= u_r[i] + v_{u_r}[i] \\ y_{u_s}[i] &= u_s[i] + v_{u_s}[i] \end{aligned} \tag{4.3}$$

The motion model can now be expressed in the standard matrix form for linear systems

$$\begin{aligned} \mathbf{x}[i+1] &= \mathbf{A}\mathbf{x}[i] + \mathbf{B}\mathbf{u}[i] + \mathbf{w}[i] \\ \mathbf{y}[i] &= \mathbf{C}\mathbf{x}[i] + \mathbf{v}[i] \end{aligned} \tag{4.4}$$

The process is assumed to be stationary and therefore the matrices are constant. Furthermore the matrix **B** is zero since no external forces (input) affect the system. Thus the system is merely driven by the previous state and the noise. By combining (4.1), (4.2) and (4.3) the system matrices are

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{x} &= \begin{bmatrix} r \\ s \\ u_r \\ u_s \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_r \\ w_s \\ w_{u_r} \\ w_{u_s} \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_r \\ v_s \\ v_{u_r} \\ v_{u_s} \end{bmatrix}
\end{aligned} \tag{4.5}$$

This model is called a constant velocity model. A constant jerk or acceleration model could also have been used. Furthermore the statistical properties of the noise is given by

$$\begin{aligned}
E[\mathbf{w}(k)\mathbf{w}^T(j)] &= \begin{cases} \mathbf{Q}_k, j = k \\ 0, j \neq k \end{cases} \\
E[\mathbf{v}(k)\mathbf{v}^T(j)] &= \begin{cases} \mathbf{R}_k, j = k \\ 0, j \neq k \end{cases} \\
E[\mathbf{w}(k)\mathbf{v}^T(j)] &= 0, \forall i, k
\end{aligned} \tag{4.6}$$

The matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ need to be symmetric and positive definite. A common choice can therefore be to make the matrices diagonal with entries on the diagonal larger than zero. The system is linear and by investigating the observability matrix the system is observable. This is not a surprise since every state can be measured. Thus the assumptions for the Kalman filter to be optimal is partly fulfilled. The process and measurement noise are assumed to be white to fulfil the rest of the assumptions, but it can affect the performance of the filter.

Equations for the Discrete Kalman Filter

The optimality of the KF will not be proven in this thesis. A proof is given in [8, Chapter 4]. However the equations for the KF will be given in Table 4.2. These equations are simplified to be customized to the system in (4.4)

Tracked objects may sometimes not be detected in every image. Therefore a measurement of the object is not always available and a prediction must be used. In practice this is solved by adjusting the corresponding elements in the matrix **C** to zero. Thus the state estimate is simply equal to the prediction. An illustration of how the Kalman filter is recursively updated is given in Figure 4.7.

Table 4.2: Equations for the Linear Discrete Kalman Filter.

Initial Conditions	$\bar{\mathbf{x}}(0) = \mathbf{x}_0$ $\bar{\mathbf{P}}(0) = E[(\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^T]$
Kalman gain	$\mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{C}^T(k)[\mathbf{C}(k)\bar{\mathbf{P}}(k)\mathbf{C}^T(k) + \mathbf{R}(k)]^{-1}$
State estimate update	$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{C}(k)\bar{\mathbf{x}}(k)]$
Error covariance update	$\hat{\mathbf{P}}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{C}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{C}(k)]^T$ $+ \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}^T(k)$
State prediction	$\bar{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k)$
Error covariance prediction	$\bar{\mathbf{P}}(k+1) = \mathbf{A}\hat{\mathbf{P}}(k)\mathbf{A}^T + \mathbf{Q}$

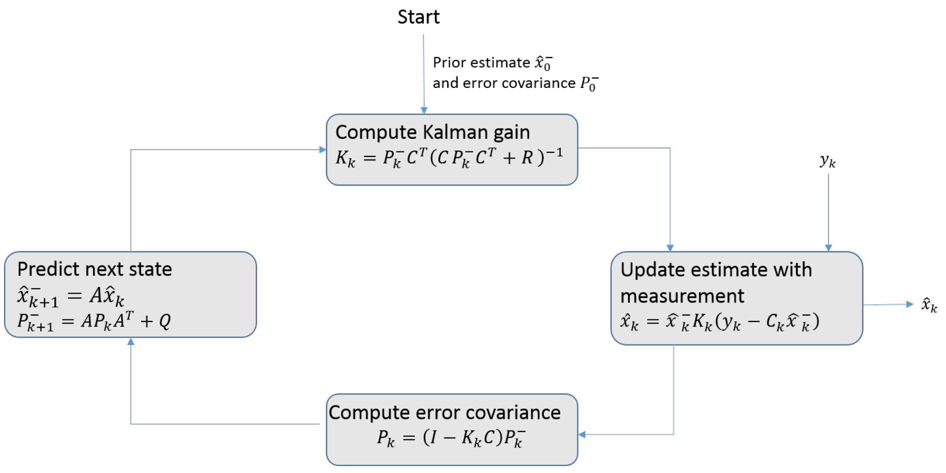

Start
Prior estimate $\hat{x}_0^-$
and error covariance P_0^-
Compute Kalman gain
$K_k = P_k^- C^T (C P_k^- C^T + R)^{-1}$
y_k
Predict next state
$\hat{x}_{k+1}^- = A\hat{x}_k$
$P_{k+1}^- = AP_kA^T + Q$
Update estimate with measurement
$\hat{x}_k = \hat{x}_k^- K_k(y_k - C_k\hat{x}_k^-)$
$\hat{x}_k$
Compute error covariance
$P_k = (I - K_kC)P_k^-$

Figure 4.7: Illustration of the discrete Kalman filter.

4.5 How to Combine Object Detection and a Kalman Filter for Object Tracking

This section will sum up the previous sections and describe the process of tracking an unknown number of moving objects. The process is illustrated in Figure 4.8 and described more closely in Table 4.3. The implementation of the tracking system is described in Chapter 5.

Table 4.3: Tracking System for several moving objects.

#	**The following steps are conducted for every image**
1.	Calculate the optical flow and use the technique described in Section 4.2 to locate moving objects. This step finds a measurement for the position and velocity of moving objects in the image plane.
2.	Create a classifier for each object with a technique from Section 4.3. The classifier can be a template centered around the detected point.
3.	Iterate through every detected moving object and compare each classifier with objects already being tracked. Find out if detected objects are being tracked already.
4.	Update the position and velocity of all detected objects that already are tracked with the new measurements. If a detected object is not tracked before, it is the first detection of the object. Create a new instance of the Kalman filter to start tracking of the object. The initial position is given by the detection.
5.	Update every object being tracked, but not detected in the last image. These objects must be updated with a prediction step only since the position and velocity cannot be measured.
6.	Stop tracking of objects not detected for a long time (described more closely in Section 5.6.

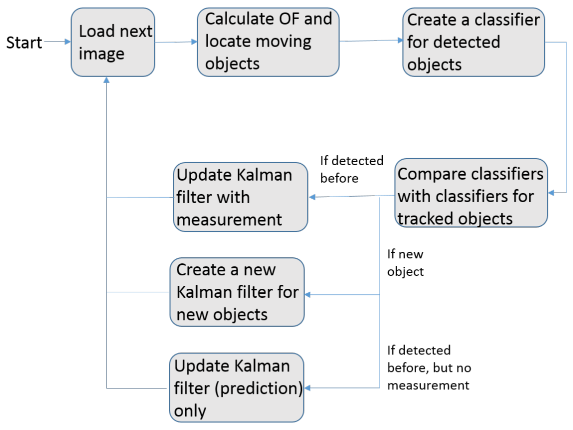

Start
Load next image
Calculate OF and locate moving objects
Create a classifier for detected objects
Compare classifiers with classifiers for tracked objects
If detected before
Update Kalman filter with measurement
If new object
Create a new Kalman filter for new objects
If detected before, but no measurement
Update Kalman filter (prediction) only

Figure 4.8: Illustration of the tracking system.

4.6 Transformation of Estimates From the Image Plane to NED

Section 4.4 looked into the problem of tracking moving objects in the image plane. However a position and velocity trajectory in the image plane are not that interesting. The position and velocity in a world-fixed frame, such as NED, are on the other hand much more interesting. This is especially the case for UAV applications. This is because the UAV is constantly moving, which causes the image plane to change location with respect to NED. Therefore the position trajectory in the image plane is of less use since it is directly related to the UAV position and not the object position. A transformation between the position in the image plane and in NED is therefore desired. This is actually already described in Chapter 2. For clarity the transformation will be given in this chapter as well. It is illustrated in Figure 4.9. In order to transform a trajectory in the image plane to NED the following assumptions are made

Assumption 4.6 The UAV position in NED, velocity and attitude are known.

Assumption 4.6 can be handled with the nonlinear observer. A pixel in the image plane can be converted to the camera-fixed frame through the pinhole camera model (2.2). The body-fixed frame coincides with the camera-fixed frame and thus the position in body-frame can be calculated. However to convert a pixel position to body a conversion between meters and pixels is necessary. Lets assume that a pixel position (r, s) in the image plane of a moving object is known. The goal is to find the position of the object in NED. A transformation from the image plane to the body-fixed frame can be calculated by rearranging (2.2) (by assuming that the body-fixed frame coincides with the camera-fixed frame)

$$\begin{bmatrix} x^b \\ y^b \end{bmatrix} = m\frac{z^b}{f}\begin{bmatrix} -s \\ r \end{bmatrix} \tag{4.7}$$

where m is meters per pixel and f is the focal length of the camera. z^b is the z-position of the object in the body-fixed frame and given by (2.23) as

$$z^b = z^c = -\frac{fc_z^n}{s\sin(\theta) + \cos(\theta)(f\cos(\phi) + r\sin(\phi))} \tag{4.8}$$

where c_z^n is the altitude of the UAV, ϕ is the roll angle and θ is the pitch angle. It is necessary to assume that the terrain is flat for this to be valid. The position of the object in the body-fixed frame is now known. A transformation from body to NED is given by the homogeneous transform (2.19). Thus the position in NED is calculated as

$$\mathbf{p}^n = \begin{bmatrix} x^n \\ y^n \\ z^n \end{bmatrix} = \mathbf{T}_b^n(\Theta, \mathbf{c}^n)\mathbf{p}^b = \begin{bmatrix} \mathbf{R}_b^n(\Theta) & \mathbf{c}^n \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}\begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} \tag{4.9}$$

where $\mathbf{R}_b^n(\Theta)$ is the rotation matrix from body to NED given by the Euler angles and $\mathbf{c}^n$ is the UAV position in NED. The velocity of the object might also be of interest. However since the estimate of the velocity in the image plane is far more inaccurate than the position the velocity will be calculated from the change in position. Thus the velocity is

$$\mathbf{v}^n[i] = \frac{\mathbf{p}^n[i] - \mathbf{p}^n[i-1]}{\delta t} \tag{4.10}$$

where δt is the time between consecutive images and $[i]$ refers to a discrete time step. Only the position in North and East are of interest since the terrain is assumed to be flat. Thus the velocity in down position is assumed to be zero.

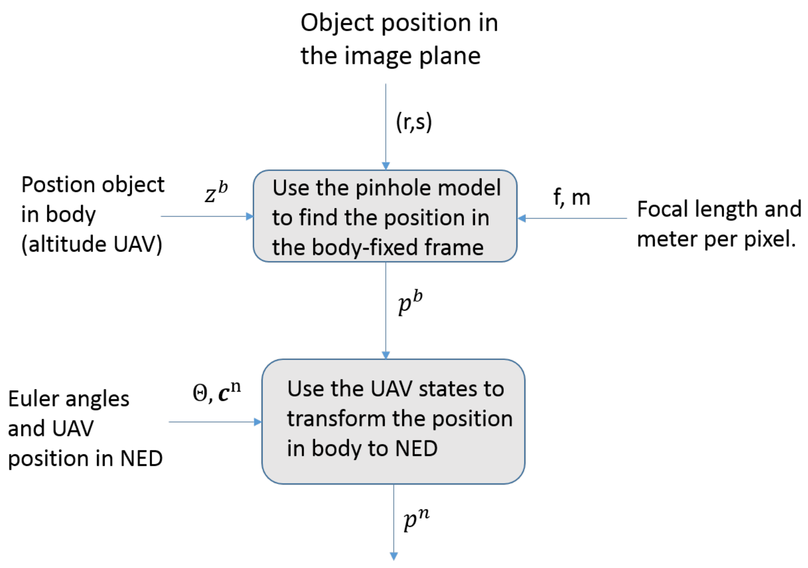

Object position in the image plane
(r,s)
Postion object in body (altitude UAV)
z^b
Use the pinhole model to find the position in the body-fixed frame
f, m
Focal length and meter per pixel.
p^b
Euler angles and UAV position in NED
$\Theta, \boldsymbol{c}^n$
Use the UAV states to transform the position in body to NED
p^n

Figure 4.9: Illustration of how to calculate NED position of object from position in the image plane.

Part III

Methods

Chapter 5

Software Implementation

Extensive theory has been presented so far in this thesis. It is time to use this knowledge in practice through experiments. However before this is possible it is necessary to implement the theory in software (SW). Therefore this chapter will look into the implemented SW and is divided in the following sections:

- Section 5.1 gives a brief introduction to the overall SW architecture.
- Section 5.2 describes the synchronization of different sensors.
- Section 5.3 describes the implementation of OF algorithms.
- Section 5.4 describes the implementation of the nonlinear observer.
- Section 5.5 describes the implementation of the moving object detection algorithm.
- Section 5.6 describes the implementation of the tracking system.

5.1 Overview

This section seeks to provide the reader with a brief description of the SW architecture. The architecture is illustrated in Figure 5.1. The SW is implemented in C++ and Matlab. The OF algorithms and the tracking system are implemented in C++. The nonlinear observer and the detection of moving objects are implemented in Matlab. The use of two different frameworks leads to some challenges, but is a consequence of simplicity and work started in the project report. Every part of this thesis is implemented on a computer with specification given by Table 5.1.

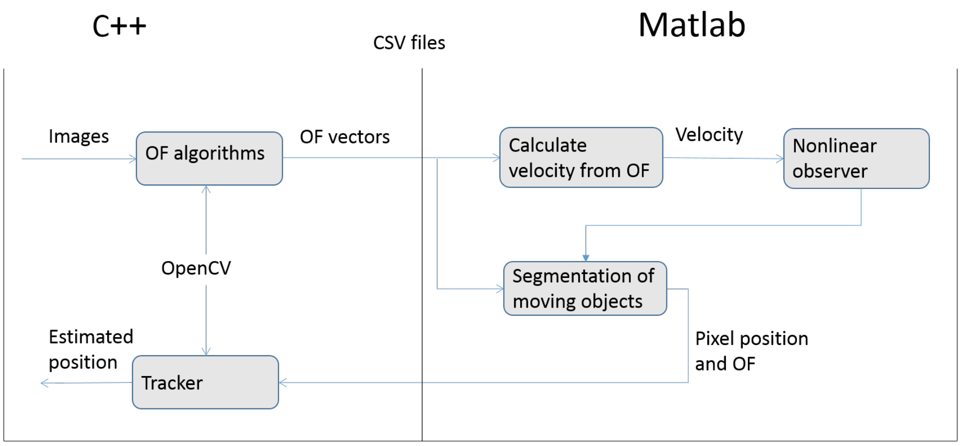


Figure 5.1: Software architecture.

Table 5.1: Computer specification.

Manufacturer	Dell
Operative system	Windows 7 Enterprise
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz
Installed memory(RAM)	8GB
System type	64 bit

5.2 Synchronization of Data

One of the largest challenges related to the implementation is the synchronization of data. As previously mentioned the observer depends on different sensors that log data at different rates. More information about the sensors is given in Chapter 6. All of these measurements need to be stored with a time stamp. Both the accelerometer, inclinometer and the gyroscopes are measured from the IMU. Thus they share a common time frame. The GPS and the video camera has their own time frame. The GPS is synchronized with the IMU through a synchronization board in the payload. Therefore the GPS and IMU stores data in the same time frame. The camera however can not be synchronized with the IMU as easily as the GPS.

The camera images are only time stamped with the Coordinated Universal Time (UTC). The time stamps of the images have a resolution of one hundredth of a second. The GPS also logs the UTC time with the same resolution. Thus the camera images can be synchronized with the GPS through the UTC time. However since the clocks can have an offset, it is necessary to define a common point in time that can be recognized on both sensors. Take-off, landing, power-up or power-off are all possible intuitive reference points. However the power-up is not reliable since the GPS needs lock before the logging begins, and the camera also needs some calibration before images are captured. Furthermore it can be hard to spot the exact take-off and landing on the images. This is because the camera is pointed straight down on the asphalt, and it is hard to spot the take-off without at least a second in uncertainty. Power-off is not a reliable choice as well, since it is impossible to be certain that the sensors stop logging at the same time.

Another and more accurate way to synchronize data were chosen based on experience from a previous experiment. A reference point can be created before take-off. A visible item, such as a screw-driver, can be placed in the field of view of the camera. Then the tip of the UAV can be lifted quickly up and down one time such that the movement is detected by the IMU. It is illustrated in Figure 5.2. The maneuver can easily be recognized in the images, by studying the visible item, down to an accuracy of one image. Furthermore the gyroscope registers the change in pitch. Thus the IMU could be synchronized with the images at the reference point with a low uncertainty. Since the IMU and GPS are synchronized through the synchronization board, the GPS and camera have a common reference point. By calculating the offset between the GPS UTC time and the camera UTC time at the reference point, it is possible to synchronize each image. However for the synchronization to be reliable the following assumption is necessary

Assumption 5.1 The clock offset between the GPS and the camera is constant during the whole flight.

This assumption is considered to be reliable since the clock offset should be constant in short periods of time.

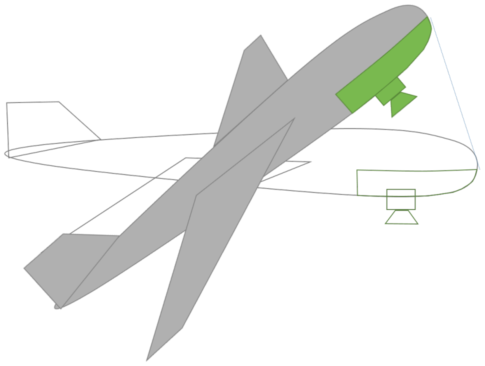


Figure 5.2: UAV maneuver for data synchronization.

5.3 Optical Flow Algorithms

The OF algorithms are implemented in C++, utilizing the Open Source Computer Vision Library[7]. This library has support for several OF algorithms. Furthermore it is documented properly and highly respected. Matlab also have some OF methods, but openCV and C++ were assessed to be the most appropriate framework. The following OF methods have been implemented for this thesis

- SIFT (Section 2.3.2)
- SURF [3]
- Fixed-point template matching (Section 2.3.1)
- Feature-based template matching (Section 2.3.1)
- A sparse version of the Lucas-Kanade algorithm [6]
- Farneback [18]

All of the algorithms follow the scheme in Figure 5.3. The images are read one by one. Thus the computation time for each iteration consists of the sum of reading images, convert the images to the desired format and run the OF algorithm. The images are stored as matrices in OpenCV. They are converted to the desired resolution and converted to necessary color representation. For the Lucas-Kanade algorithm, Farneback, feature based template matching and fixed-point template matching the images needs to be monochrome. The rest of this section will describe the implementation of SIFT and fixed-point template matching since these methods are used later for the UAV experiment in Chapter 6.

Fixed-Point Template Matching

Fixed-point template matching is implemented with functions from OpenCV. There are some important parameters to consider. The horizontal and vertical dimension of the template are important variables. They should be adjusted with the desired image resolution. The template should decrease with the resolution of the image. The fixed-point template matching divides the image in regions as illustrated in Figure 5.4. The number of regions can be chosen by the user. A template, shaped as a rectangle, is extracted from the centre of each region. Naturally the templates should be smaller than the regions. The maximal template size in horizontal direction (width) is given by the following formula

$$T_x^{max} = \frac{Res_x}{nRegions_x}$$

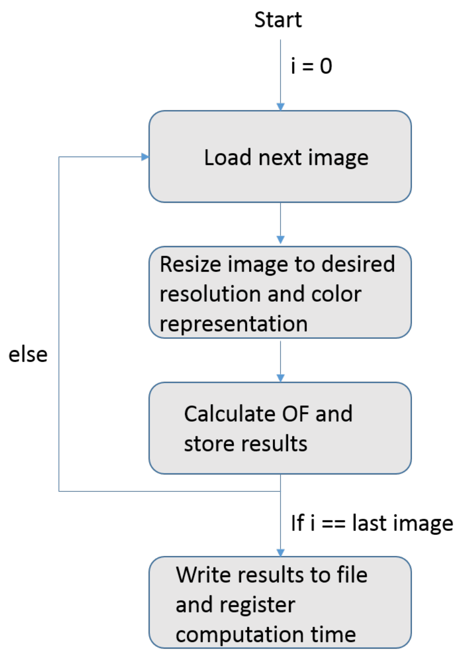

Start
i = 0
Load next image
Resize image to desired resolution and color representation
Calculate OF and store results
else
If i == last image
Write results to file and register computation time

Figure 5.3: Optical flow software structure.

where T_x^{max} is the maximal width of the template, Res_x is the horizontal resolution of the image and $nRegions_x$ is the number of horizontal regions. The same formulas are valid for the vertical direction by using the vertical parameters instead. The templates are matched with the next image to find the displacement (OF vector). A OF vector is only calculated if the match for the template is above a threshold of 99%. This value might seem high, but there is still a chance for mismatches, even with this value.

The number of regions will naturally affect the computation time since no parallelism is implemented. Thus each new region will increase the computation time almost linearly. A linear increase will be the case if the templates have the same size independently of the number of regions. However the templates are normally smaller when the number of regions increases (to keep the template smaller than the size of the region).

The following parameters have been used in the experiments

- Image resolution $= 1600 \times 1200$ pixels
- Template width $= 120 \times 90$ pixels
- Number of regions $= 4 \cdot 3 = 12$ regions

- Threshold for successful match = 99%
- Template matching performed on monochrome images.

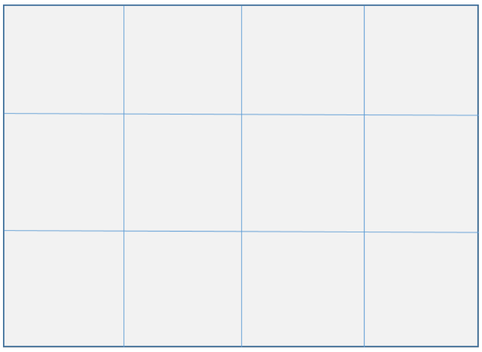


Figure 5.4: The image is divided in symmetrical regions.

SIFT

The SIFT algorithm is implemented with OpenCV. There exist a class for the SIFT algorithm. The class is used to find features and calculate descriptors for each feature. A descriptor matcher in OpenCV is used to find common features in consecutive images. The matching algorithm is called FLANN (Fast Library for Approximate Nearest Neighbours) [39]. The matcher is based on a nearest neighbour search in high dimensional spaces. The library contains a collection of algorithms for nearest neighbour matching and a system for choosing the best algorithm for the application.

Since the matching algorithms are based on a nearest neighbour search, every feature from the image with least features is provided a match. However some of these matches might be incorrect and should be removed. Each match is returned with a value that expresses the least squares distance between the descriptors matched together. It is simply a measure of the similarity between the features. Thus erroneous matches can be removed by comparing the best match with the rest of the matches. Every match that is within 2,5 times of the value of the best match is kept. Thus only the best matches are taken further for OF calculation. This decreases the number of OF vectors, but at the same time increases the robustness by avoiding insecure OF vectors. The displacement of the features between consecutive images are stored as the OF. The following list sums up the most important points for the SIFT implementation

- The number of maximal features extracted from the images is unlimited.
- A vector of 128 elements is used as the descriptor. That is the maximal size of

the descriptor and is used to maximize the possibility of unique descriptors.

- A FLANN-based matcher is used to connect features from different images.
- Only the best matches are kept by comparison with the best match.

5.4 The Nonlinear Observer

The nonlinear observer is implemented in Matlab. This is an advantage since the flight data from the experiments are converted to matrices in Matlab. The data are synchronized by the approach described in Section 5.2. OF vectors from C++ are read from a csv file and transformed to velocities through the method in Section 2.4. OF vectors from SIFT and template matching are combined to maximize the probability of having a sufficient number of OF vectors. The implementation of the nonlinear observer follows the scheme illustrated in Figure 5.5. The observer runs at a rate given by the sensor with the largest measurement rate. The different sensors are described in Chapter 6.

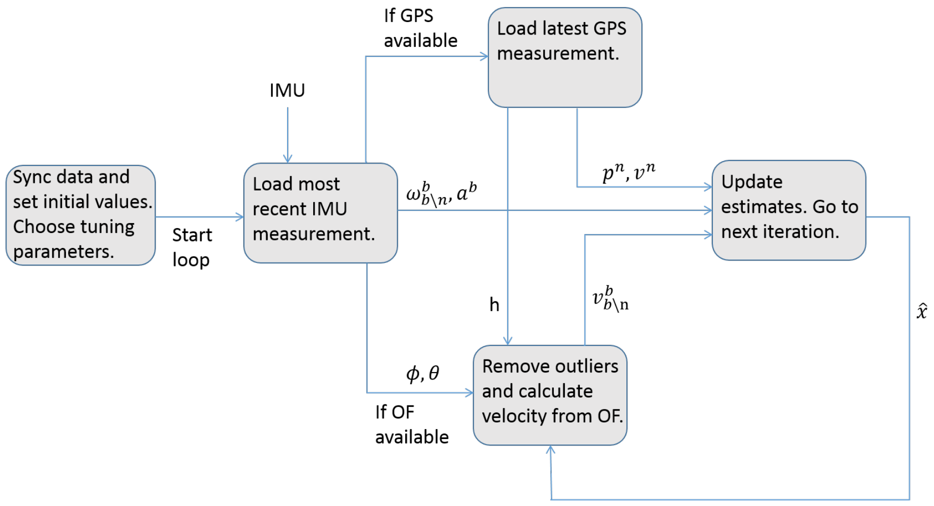

If GPS available
Load latest GPS measurement.
IMU
Sync data and set initial values. Choose tuning parameters.
Start loop
Load most recent IMU measurement.
$\omega^b_{b\backslash n}, a^b$
p^n, v^n
Update estimates. Go to next iteration.
h
$v^b_{b\backslash n}$
$\hat{x}$
ϕ, θ
If OF available
Remove outliers and calculate velocity from OF.

Figure 5.5: Observer software structure.

In the first box in Figure 5.5 the data are loaded and synchronized. Every measurement from the IMU is looped through. A large lookup-table is created in order register the indices where new GPS and IMU measurements are available. When the loop is at these indices measurements from the camera and GPS are used in the calculation of next estimate. The camera and GPS measurements are in general not available at the same indices.

Body-fixed velocity of the UAV is calculated at every index where a new OF measurement is available. The OF vectors are first filtered through an outlier detector (developed in the project report) in order to remove erroneous OF vectors. Roll and pitch angles from the inclinometers and the latest height measurement from the GPS are used to calculate the body-fixed velocity after the removal of outliers. The body-fixed velocity calculated from the camera is normalized before it is sent further to update the estimates. All available measurements at the current index are used to update the estimates in the nonlinear observer. The updates are conducted with the first order Euler method. The first order derivative of a state $\dot{x} = f(t, x(t))$ can be approximated by

$$\dot{x} = \frac{x_{i+1} - x_i}{\delta t}$$

where x_{i+1} is the current time step, x_i is the previous time step and δt is the time difference between x_{i+1} and x_i. By rearranging the equation and inserting for $\dot{x}$ the estimates can be updated by

$$x_{i+1} = x_i + \delta t \cdot f(i, x_i) \tag{5.1}$$

(5.1) is called the first order Euler method. The first order approximation of the derivative is very simple and more accurate approximations exists. However other methods have not been tested in this thesis. The large measurement rate of the IMU (300Hz) makes sure that the approximation is performed in a short time interval which increases the accuracy of the approximation.

5.5 Moving Object Detection Algorithm

The moving object detection algorithm is implemented in Matlab. The OF vectors and the position of each vector are loaded in Matlab. The search for moving objects is conducted by calculating the theoretical flow (Section 4.2). The theoretical flow depends on data from other sensors. Thus one of the most important factors of the implementation is the synchronization of data described earlier. This is crucial since the theoretical flow is calculated from the sensor data. Thus correct sensor data for each set of OF vectors are necessary to create reliable results. The implementation of the moving object detection algorithm is illustrated in Figure 5.6.

The first block in Figure 5.6 loops through every OF vector in a single image, and tries to find OF vectors belonging to moving objects. That is the algorithm described in Section 4.2.3. This block sends out the OF vectors and the position of the OF vectors. However only one detection in a local area of the image is desired. Thus every OF vector belonging to moving objects are compared in the second block. If the start position of some of the vectors are within a small pixel displacement of another OF vector, they are assumed to be fixed to the same object. All OF vectors assumed to be related to the same object are combined to find one position and OF value for each moving object. That is the mean position and OF.

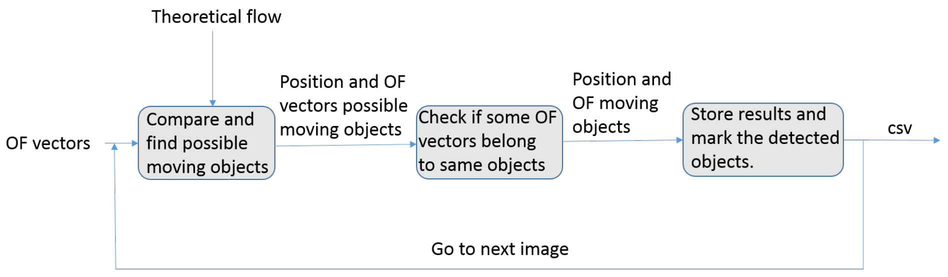

Theoretical flow
OF vectors
Compare and find possible moving objects
Position and OF vectors possible moving objects
Check if some OF vectors belong to same objects
Position and OF moving objects
Store results and mark the detected objects.
csv
Go to next image

Figure 5.6: Moving object detection software structure.

OF vectors positioned within a square of 100 pixels are assumed to belong to the same object. Thus the following assumption is necessary for the approach to be reliable.

Assumption 5.2 Only one moving object exists within a square of 100 pixels.

5.6 Tracking System

The tracking system is implemented in C++ with OpenCV. OpenCV has support for easy implementation of Kalman filters. A new instance of the Kalman filter is created for every new moving object. The segmentation algorithm in Matlab provides a single OF vector and position for moving objects. Every moving object found in a single image must be compared with previously detected objects. Every detection corresponds to an already detected object or a new object. A template centered around the position of the OF vectors is used as a classifier. Template matching between previously detected objects and the new objects are used to decide if the object have been detected before. Previously detected objects are updated with new measurements. New objects are initialized with position and OF given by the measurement. Moving objects not detected for several frames are removed from the tracking system. The implementation of the tracking system is illustrated in Figure 5.7. The first block in Figure 5.7 decides if an object has been detected before. This is a critical part of the tracking system in order to avoid objects being mixed. Every single detected object goes through the following process, which is a description every block in the figure.

1. Extract the part of the image where a moving object is detected. That is an image of 100×100 pixels centered around the position of the moving object.

2. Run template matching with every previously tracked object. Every tracked object has a template of 60×70 pixels. Find the previously tracked object with the best match and perform one of the following prioritized strategies.

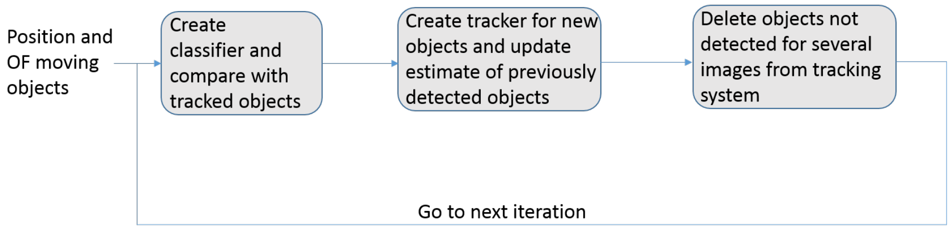

Position and OF moving objects
Create classifier and compare with tracked objects
Create tracker for new objects and update estimate of previously detected objects
Delete objects not detected for several images from tracking system
Go to next iteration

Figure 5.7: Tracking system software structure.

   (a) If the best match is above 99%, the object is tracked already and the match is very reliable. Update estimate of the best match with new measurement and update the template with the last image.

   (b) If the best match is above 95% and the position of the detected object is within a square of 200 pixels from the last estimate, the match is considered to be reliable. Update estimate for the best match with new measurement, but do not update the template.

   (c) If the best match is above 90% the result is inconclusive. The moving object might be tracked already. Therefore a new tracker is not created for the object and the best match is not updated with the new measurement.

   (d) If neither of the conditions above are satisfied, the object is considered to be new. Initialize a new instance of the Kalman filter with the new measurement and create a template of the object from the last image.

3. Update the estimates of every tracked object not detected in the last frame by prediction.

4. Delete every tracked object that satisfies one of the following conditions:

   (a) Object detected at least 40 times, but not in the last 60 images.

   (b) Object detected below 20 times and not in the last 10 images.

   (c) Object detected below 10 times and not in the last 5 images.

The following assumption is necessary for this process to be valid.

Assumption 5.3 Every moving object can be sufficiently described by a template of 60×70 pixels. In practice this means that the object should be smaller than the chosen template size.

Chapter 6

UAV Experiment and Description of Case Studies

Theory and the SW implementation have been described so far in this thesis. It is time to evaluate the theory and SW in practice. This chapter concerns a UAV experiment carried out in February, at Eggemoen Norway. Figure 6.1 is a picture of the team who conducted the experiment. The chapter is divided in the following sections:

- Section 6.1 describes the UAV and payload used to carry out the experiments.
- Section 6.2 describes a UAV test flight carried out in February 2015.
- Section 6.3 describes the case studies carried out to evaluate the nonlinear observer.
- Section 6.4 describes a case study conducted to evaluate the moving object detection algorithm.
- Section 6.5 describes two case studies conducted to evaluate the tracking system.

6.1 UAV and Payload

A fixed-wing Penguin B UAV, produced by UAV Factory, was used to carry out the flight experiment. The UAV was equipped with a custom payload module,

Figure 6.1: Picture from the experiment February 2015, Eggemoen Norway.

developed by Lorenzo Fusini, Sigurd M. Albrektsen, Jakob M. Hansen and Kasper T. Borup, in order to record the desired data. The payload is the main focus of this section. The Penguin B UAV is displayed in Figure 6.2 and the most important properties are given in Table 6.1.

Table 6.1: Specification Penguin B fixed-wing UAV.

Name	**Penguin B**
Manufacturer	UAV FACTORY
Length	2,27 m
Wing Span	3,3 m
Weight (without fuel and payload)	10 kg
Maximal payload weight	10 kg
Take-off method	Runway
Cruise speed	22 m/s
Maximal Level Speed	36 m/s
Power supply payload	Batteries
Power supply Engine	Gasoline
Autopilot	Piccolo

The penguin autopilot uses an Extended Kalman Filter (EKF) to estimate the states. The autopilot is called Piccolo and the data from Piccolo are used for comparison in the results. Piccolo stores data at a rate of 1Hz.

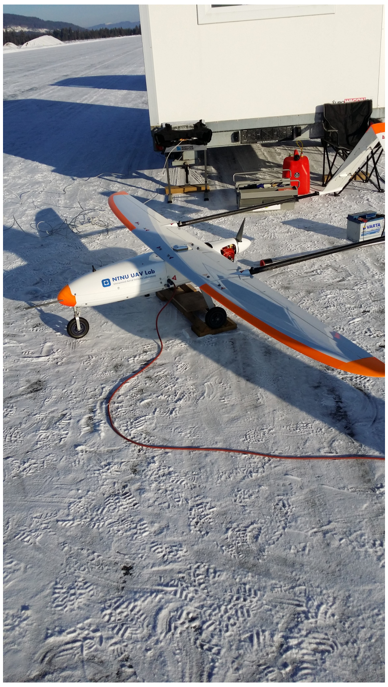


Figure 6.2: The Penguin B fixed-wing UAV on the runway at Eggemoen.

The Payload

The UAV has a payload with different sensors displayed in Figure 6.3. Accelerometers, gyroscopes, inclinometer, GPS, altimeter and a video camera are present in the payload. A complete list of the most important sensors in the payload for this thesis is given in Table 6.2.

Table 6.2: Sensors used in the flight experiment.

Sensor	**Manufacturer**	**Device name**	**Measurement Rate**
Camera	IDS	UI-5250CP-C-HQ	10 Hz
GPS	uBlox	EVK-6	5 Hz
IMU	Sensonor AS	STIM 300	300 Hz
IMU	Analog Devices	Adis 16488	410 Hz

The individual parts of the payload will be described separately.

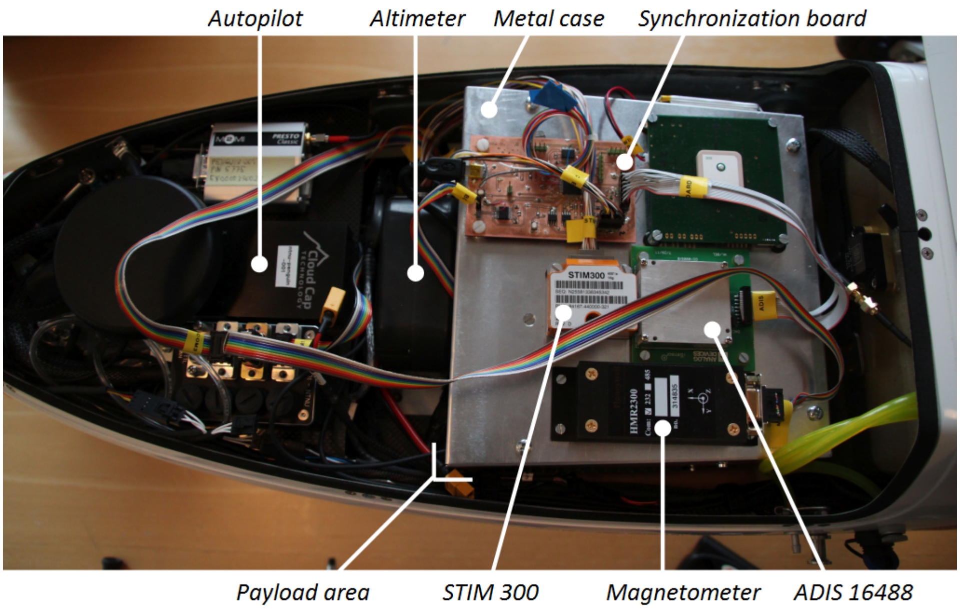


Figure 6.3: The payload.

Camera

A commercial video camera manufactured by IDS is placed in the payload to capture images of the ground. The lens is produced by Tamron. The camera and lens specification are given in Table 6.3. The lens has a constant focal length (no optical zoom), which in practice means that the field of view in both horizontal and vertical direction are constant. The camera is displayed in Figure 6.4.

Figure 6.4: The camera.

GPS

A uBlox EVK-6 GPS displayed in Figure 6.5 was used in the experiment. GPS measurements were logged at a rate of 5Hz. The GPS logs the longitude, latitude and the height of the UAV. These measurements are converted to position measurements in NED which is used in the nonlinear observer.

Table 6.3: Camera and lens specification. h is the altitude of the UAV.

Name	**UI-5250CP-C-HQ**
Resolution(h x v)	1600 x 1200
Sensor Size(h x v)	7,2 mm x 5,4 mm
Frame rate	10 Hz
Color depth	12 bit
Size of single uncompressed image	2,75 MB
Focal length	8mm
Horizontal field of view α_h	48,46 degrees
Vertical field of view α_v	37,3 degrees
Width captured	$2 \cdot h \cdot \tan(\frac{\alpha_h}{2})$
Height captured	$2 \cdot h \cdot \tan(\frac{\alpha_v}{2})$
Number of pixels per meter(max resolution)	$\frac{1600}{2 \cdot h \tan(\frac{\alpha_h}{2})}$
Data transmission	Gigabit ethernet
Input Voltage	12-24 VDC

Figure 6.5: uBlox EVK-6 GPS.

IMU STIM 300

The STIM 300 is displayed in Figure 6.3. It logs data at a rate of 300 Hz. STIM 300 measures angular velocity in body-fixed coordinates through three gyroscopes (deg/s) and acceleration in body-fixed coordinates through three accelerometers (g). It also measures inclinometer readings in g. The inclinometer measurements can be converted to roll and pitch angles [20, Chapter 11]. The STIM 300 is not placed exactly in the centre of gravity. The measurements are not converted to the centre of gravity and thus the following assumption is necessary:

Assumption 6.1 The IMU is placed close enough to the centre of gravity such that the inaccuracy related to the misalignment is negligible.

In practice the misalignment vector in body-fixed coordinates is $[0.2, 0, -0.05]^T$ given in meters.

IMU Adis 16488

The Adis 16488 IMU is displayed in Figure 6.3. It logs data at a rate of approximately 410 Hz. The sensor measures angular velocity in body-fixed coordinates through three gyroscopes (deg/s) and acceleration in body-fixed coordinates through three accelerometers (g). In addition it measures the magnetic field in body-fixed coordinates (mG) and barometer readings (bar). Assumption 7.1 is also necessary for this sensor. The misalignment vector for the Adis 16488 is $[0.15, 0, -0.05]^T$ given in meters.

6.2 Description of Flight Experiment at Eggemoen

This section provides a short description of the flight experiment at Eggemoen February 2015. Three flights were performed. The first flight lasted approximately five minutes and was used to check the camera parameters and the image quality. The images captured in the first flight were to bright. Thus the lens was adjusted to capture less light. Another short flight of approximately four minutes was conducted to check if the new settings worked better. The images from the second flight were much better and the camera settings were kept for the third flight. The third flight was the longest and lasted for approximately 30 minutes.

The data from the second and third flight were stored successfully on the hard-drive. For simplicity only the data from the second flight are used in the case studies. This is mainly because it is easier to work with a smaller amount of data.

6.3 Experiments With the Nonlinear Observer

This thesis has investigated different themes (Chapter 2-4). Therefore several case studies are performed in order to evaluate and test the theory in practice. This section concerns simulations related to the nonlinear observer. The observer is simulated offline. Two separate case studies will be carried out.

6.3.1 Case Study 1: Calculation of Body-Fixed Velocity from Optical Flow

The first case study evaluates the calculation of body-fixed velocity from OF. The body-fixed velocity are calculated from OF both by measurements of the roll, pitch and altitude and estimates of the same states. Roll, pitch and altitude are measured by an inclinometer and GPS. With estimates, roll, pitch and altitude are extracted from a state estimator (the nonlinear observer). The nonlinear observer are evaluated in case study 2.

6.3.2 Case Study 2: Simulation of the Nonlinear Observer

The second case study is simulation of the nonlinear observer. The goal of the case study is to compare the estimates from the nonlinear observer with the estimates from the Piccolo autopilot (EKF). The performance of the nonlinear observer with camera and magnetometer is going to be compared and evaluated. The nonlinear observer is initialized with attitude (Euler angles) of zero degrees. The position and velocity are initialized by the first GPS measurement. The following observer gains were chosen for the observer with camera. $\mathbf{I3}$ is the 3×3 identity matrix and $\text{diag}([x1, x2, x3])$ a diagonal matrix with $x1$, $x2$ and $x3$ on the diagonal and zeros in the rest of the matrix.

- $L_{b^b} = 2.0$ rad/s and $L_{\hat{b}^b} = 2.1$
- $\sigma = 1$, $\mathbf{K}_p = \text{diag}([0.16, 0.12, 0.16])$ and $k_I = 0.002$
- $\mathbf{K}_{pp} = 30\mathbf{I3}$, $\mathbf{K}_{pv} = 2\mathbf{I3}$
- $\mathbf{K}_{vp} = \mathbf{I3}$, $\mathbf{K}_{vv} = 100\mathbf{I3}$
- $\mathbf{K}_{\xi p} = \mathbf{I3}$, $\mathbf{K}_{\xi v} = 50\mathbf{I3}$

For the observer with magnetometer the following observer gains were chosen

- $L_{b^b} = 2.0$ rad/s and $L_{\hat{b}^b} = 2.1$
- $\sigma = 1$, $\mathbf{K}_p = 0.14*\text{diag}([0.55, 0.5, 0.45])$ and $k_I = 0.001$
- $\mathbf{K}_{pp} = 30\mathbf{I3}$, $\mathbf{K}_{pv} = 2\mathbf{I3}$
- $\mathbf{K}_{vp} = \mathbf{I3}$, $\mathbf{K}_{vv} = 100\mathbf{I3}$
- $\mathbf{K}_{\xi p} = \mathbf{I3}$, $\mathbf{K}_{\xi v} = 50\mathbf{I3}$

Measurements from the accelerometer, gyroscope and inclinometer are extracted from the logged data by the STIM300 IMU. Magnetometer measurements are extracted from the data logged by Adis 16488. The magnetometer was not calibrated before the experiment. However the data have been aligned with the magnetometer measurements in the Piccolo autopilot after the experiment. The piccolo magnetometer was calibrated before take-off. This is conducted by identifying the mean difference between the Adis and Piccolo for the all of the measurements. This calibration is likely to increase the accuracy, but the magnetometer measurements are still not very reliable. The magnetic field at Eggemoen in NED is

$[150.06, 6.71, 488.59]$ mG.

6.4 Experiment With the Moving Object Detection Algorithm

In order to test the moving object detection algorithm images with a moving object present is necessary. However moving objects did not exist in the experiment conducted at Eggemoen. Therefore several image sequences extracted from the flight at Eggemoen have been edited to include a moving object. In this manner a large test set of images have been created with moving objects. The test set have also been expanded with several of the original images to include images both with and without moving objects. Some images from the test set are displayed in Figure 6.6. Information about the test set is given in Table 6.4.

Table 6.4: Specification of moving object detection test set.

Number of images	366
Number of images with a moving object	238
Number of images with object partly visible	0
Number of images without a moving object	128

A red rectangle and a black car are the moving objects and edited into some of the images. The red rectangle is created manually and the black car is an image of a real car displayed in Figure 4.5. Figure 6.6 contains examples of images with and without the moving object. There are never two moving objects in the same image which is in accordance with the problem formulation. Four different paths in the image plane have been used to insert the moving object to make the test set reliable. Common for all paths are constant velocity in the image plane. In practice this means that the paths have constant displacements in pixels between subsequent images. The displacement is different for the image paths in order to create a reliable test set. Furthermore the paths are directed in different directions in the image plane. The minimum speed of the object is 30 km/h. Thus slowly moving objects are not present in the test set. The moving objects are chosen to be clearly visible in the image. The white diagonal lines in the rectangle are edited in to increase the possibility of the feature detector to find features on the object.

In addition to the images with moving objects, several of the original images without moving objects have been added to the test set. This is to evaluate the algorithms ability to resist false detections.

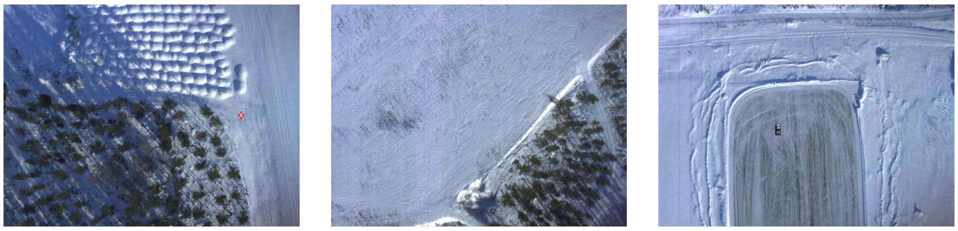


Figure 6.6: Example images from the test set for moving object detection algorithm. The red rectangle in a) and black car in c) are moving objects. b) is an image without a moving object.

6.4.1 Case Study 3: Moving Object Detection

The performance of the moving object detection algorithm with different tuning values will be evaluated in this case study. The algorithm is evaluated on the test set described in Table 6.4. Several features can be detected on a single moving object which means that several OF vectors belonging to the object might be measured. The detection algorithm should be able to find all vectors belonging to a single moving object and also keep track of the total number of moving objects in an image.

Three different sets of tuning values have been chosen. The first set is considered to default tuning values and most appropriate in general. The second set contains tuning values that lead to softer constraints before a OF vector is classified as moving. Thus it could be more sensitive to false detections, but at the same time easier find the moving objects. Thus if the object is moving slowly with respect to the background these tuning values might be appropriate. The third set contains tuning values that lead to stronger constraints. It is the least sensitive to false detections, but also increases the chance of missing moving OF vectors. The tuning set is given in Table 6.5.

Table 6.5: Specification of the tuning set for the moving object detection algorithm.

Test Set	K_σ	L_σ	K_μ	L_μ
Default	2,5	15	2,5	7
Soft Constraints	2	12	2	5
Hard Constraints	3	20	3	11

6.5 Experiments With the Object Tracking System

It is also necessary to create images with a moving object to test the tracking system. Two case studies have been created to evaluate the tracking system.

6.5.1 Case Study 4: Tracking of Single Object

A path for a moving object in NED has been designed. The path is created such that the object is visible in several images. The pinhole camera model and the position of the UAV have been used to find the correct image coordinates for the object. A car have then been inserted into the images with center at the calculated image coordinate. The car is tracked in the image plane. The tracking path is then converted back to the NED to compare it with the true path. Noise will be present since detected objects will have position given by the OF vectors and these are not necessarily given by the center of the object. Furthermore the calculation of the OF vectors are also noisy which means that all measured states have noise to make the simulation realistic.

The true position and velocity of the object are described in Figure 6.8. The position in the image plane is used to find out if the object is visible or not. The image resolution is 1600×1200 ($r \times s$). Therefore the r-coordinate must be in the closed pixel interval of $[1, 1600]$ and the s-coordinate in the closed interval $[1, 1200]$ in order to be visible in the image. Figure 6.7 displays two images with the car. It is clearly visible on the runway and the size of the car is assessed to be fairly realistic. Two real other cars at rest are visible in the left image, but are quite difficult to see because they are white. The shadows are visible though. The size of the car edited in is slightly larger than the real cars, but not significantly larger. It is important to notice that other objects might be tracked in the case of misdetections. There is only one moving object in the scene, but there are no upper limit on the number of tracked objects. Thus the segmentation algorithm might start tracking of other objects as well (if they are classified as moving).

The Kalman filters for each newly detected moving object is initialized with the following covariance matrices (given in pixels)

$$\mathbf{Q}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_k = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

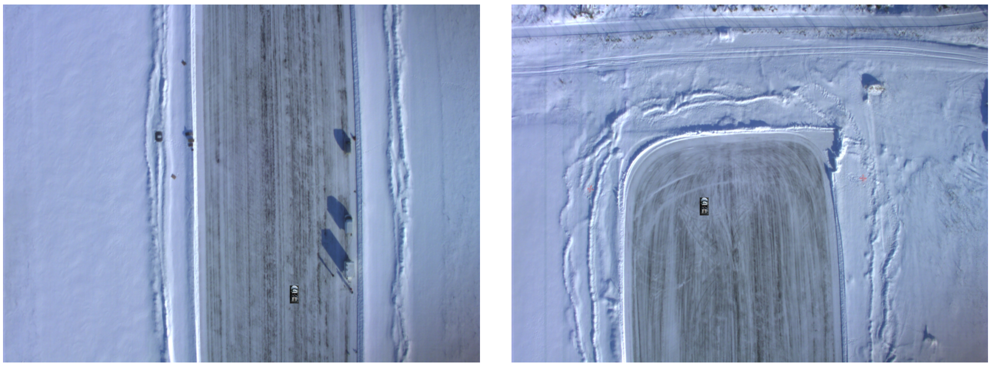


Figure 6.7: Example images for tracking of car.

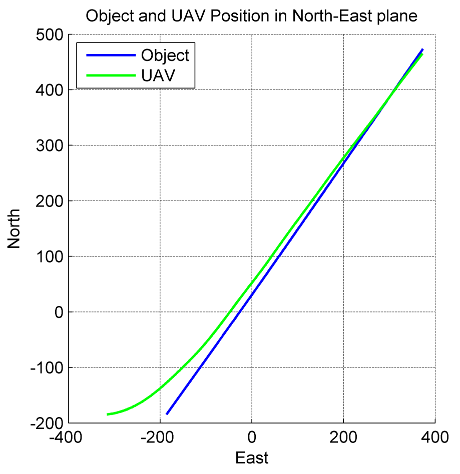
6.5.2 Case Study 5: Tracking of Multiple Objects

Multiple object tracking is a much harder problem than tracking of single objects. This case study considers tracking of two different cars. Two different paths in NED have been designed and objects edited into the images at the correct pixel position. The purpose of this case study is to evaluate the performance of the tracking system when multiple objects are moving at the same time. Furthermore the objects are visible at the same time in the images and that might be a challenge. The true path of the objects are illustrated in Figure 6.10. The first car has the same path and velocity as in Case Study 4. The second car has a constant velocity and a slightly different path. The paths have been designed such that the the first car eventually drives past the other car as illustrated in Figure 6.9.

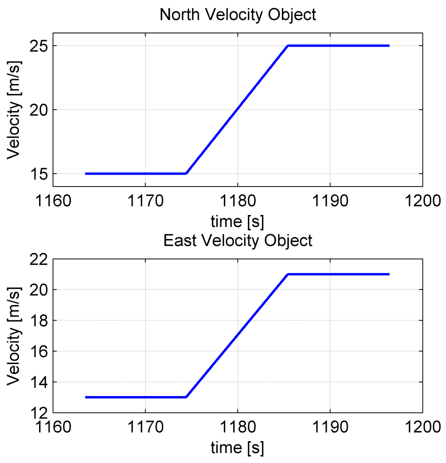
The second car has a constant velocity in NED. However the velocity in the image plane is not constant since the UAV has a time-varying attitude. Nevertheless since the second car has a constant velocity in NED, the velocity in the image plane is not varying as much as for the first car. Furthermore the second car is visible in the image from the first detection until it eventually disappears.

The Kalman filter for each newly detected moving object is initialized with the following covariance matrices (given in pixels) in this case study

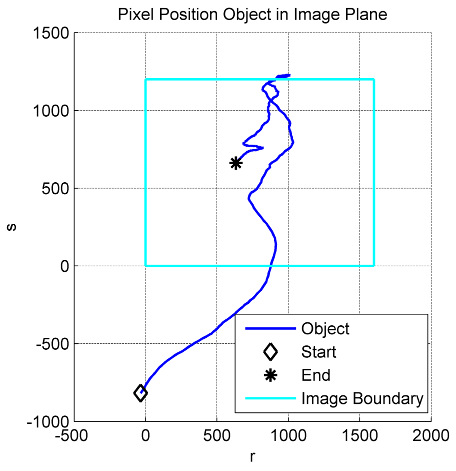
$$\mathbf{Q}_k = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad \mathbf{R}_k = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$



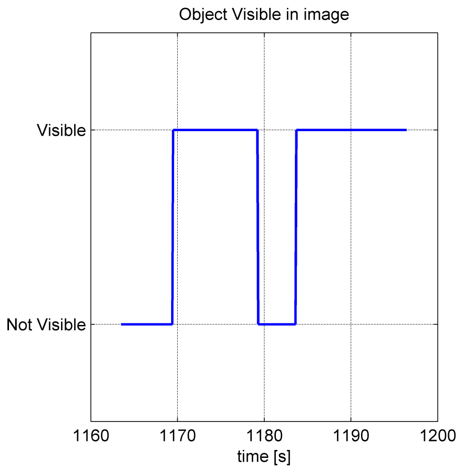
a)



b)



c)



d)

Figure 6.8: Simulated object path Case Study 4. a) Object and UAV path in North-East plane. b) Object and UAV velocity in North and East. c) Pixel position object in the image plane. d) Visibility of object.

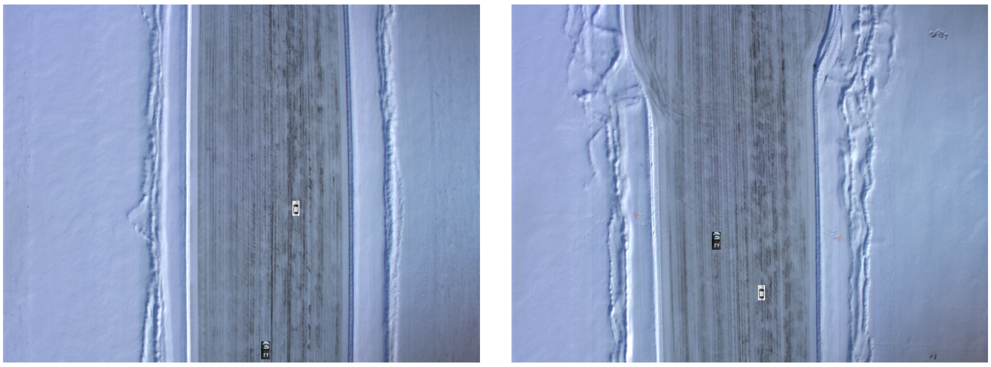


Figure 6.9: Example images for tracking of two different cars.

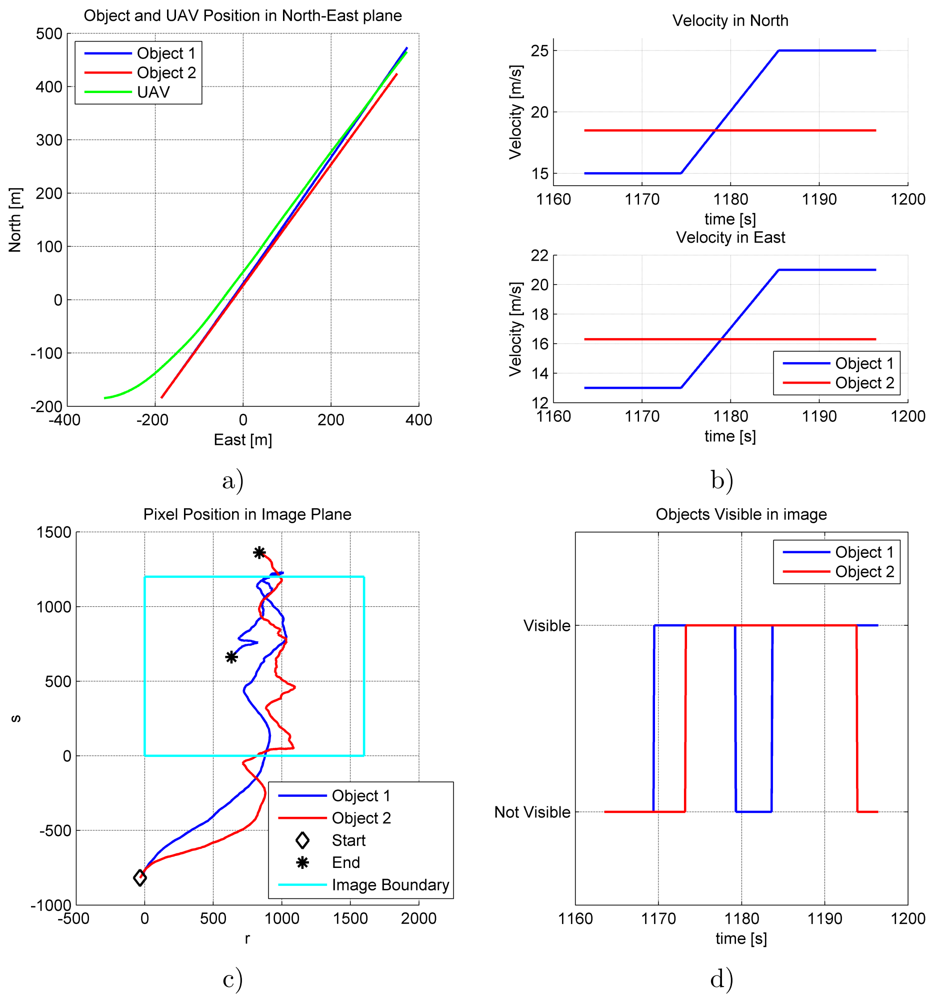


Figure 6.10: Simulated object paths Case Study 5. a) Object and UAV paths in North-East plane. b) Object and UAV velocities in North and East. c) Pixel position objects in the image plane. d) Visibility of objects.

Part IV

Results and Discussion

Chapter 7

Results and Discussion

This chapter presents the results and discussion for the case studies described in Chapter 6. The results for each case study will be presented and discussed independently. The chapter contains the following topics:

- Section 7.1 defines common performance measures for every case study.
- Section 7.2 presents the results of case study 1 described in Section 6.3.1.
- Section 7.3 presents the results of case study 2 described in Section 6.3.2.
- Section 7.4 presents the results of case study 3 described in Section 6.4.1.
- Section 7.5 presents the results of case study 4 described in Section 6.5.1.
- Section 7.6 presents the results of case study 5 described in Section 6.5.2.

7.1 Notation

Some common performance measures are defined before the results of the different case studies are presented. The estimation error is defined as

$$e = \hat{x} - x_{ref}$$

where $\hat{x}$ is the estimate and x_{ref} is the reference used for comparison. The mean error is defined as the mean of the error for all time steps

$$\bar{e} = \frac{1}{n}\sum_{i=0}^{n} e(i)$$

where n is the total number of time steps. The square root of the mean squared error (RMSE) is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n} e(i)^2}$$

RMSE represents the accuracy of the estimates and the noise level.

7.2 Case Study 1: Calculation of Body-Fixed Velocity from Optical Flow

This case study evaluates the calculation of body-fixed velocity from OF described in Section 2.4 and 6.3.1. The calculated velocities with measurements and estimates of roll, pitch and altitude are compared with the body-fixed velocity from the EKF in Piccolo. Figures displaying the calculation of body-fixed velocities will be presented. The results will be presented first and then a discussion ends the case study. This case study is solely based on real data.

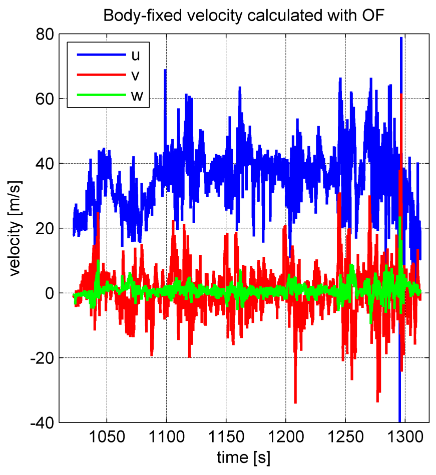
Results

Figure 7.1 displays the calculated body-fixed velocity with measurements and estimates (from nonlinear observer) of roll, pitch and altitude. Table 7.1 and Table 7.2 presents the mean error and RMSE of the calculated body-fixed velocity with measurements and estimates, respectively. The values are calculated with the EKF in Piccolo as reference.

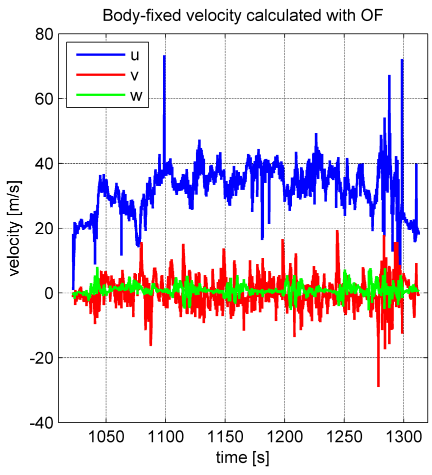
Table 7.1: RMSE and mean error in calculated body-fixed velocity with measurements of roll, pitch and altitude.

Velocity	**Mean Error $\bar{e}$**	**RMSE**
u	7,19	10,8
v	-0,62	7,57
w	-0.46	1,75

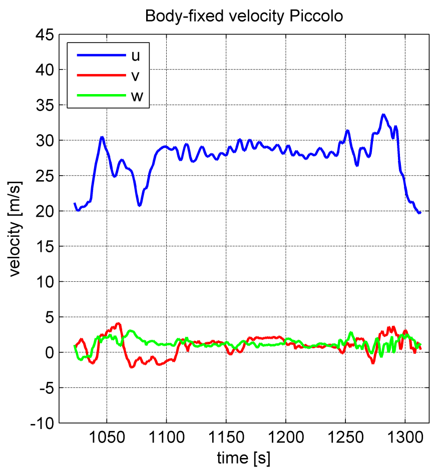
The mean of the body-fixed velocities calculated from OF are not very far from the true values, but the longitudinal velocity (u) has a mean significantly above



a) With Measurements



b) With Estimates



c) Body-fixed velocity piccolo

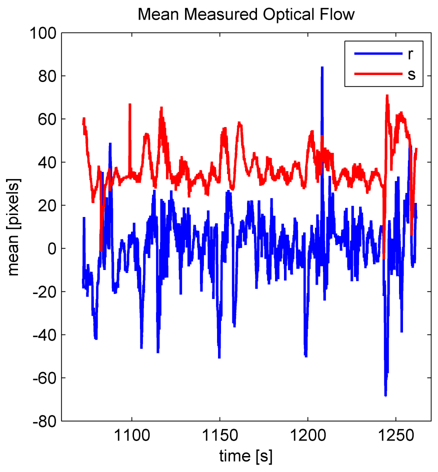
Figure 7.1: Body-fixed velocity calculated from optical flow

Table 7.2: RMSE and mean error in calculated body-fixed velocity with estimates of roll, pitch and altitude.

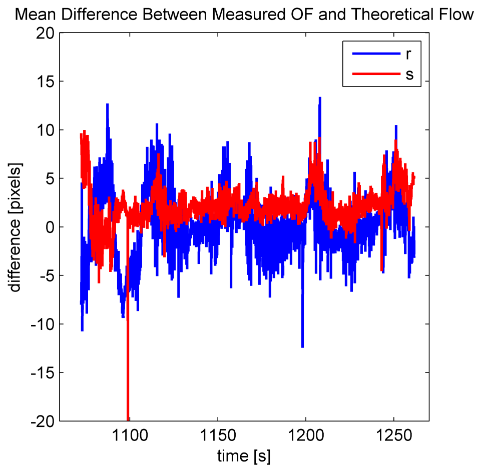
Velocity	Mean Error $\bar{e}$	RMSE
u	5,71	8,02
v	-1,27	4,20
w	-0,20	0,94

the reference. However the RMSE is large and this is easy to see in Figure 7.1 by the noise level. The noise level is especially large for the body-fixed velocity calculated with measurements of roll, pitch and altitude. The noise level is reduced significantly with estimates, but the RMSE is still quite high. Furthermore there are some oscillations in the longitudinal velocity. The local maxima of these oscillations are located at times where the UAV turns. Thus roll motion is probably not recognized properly, and instead assumed to be an increase in longitudinal velocity.

The accuracy of the transformation from OF to velocity is going to be examined further. Theoretical flow can be calculated with the body-fixed velocity, roll, pitch and altitude given by Piccolo. Thus it is possible to compare the theoretical flow with the measured OF. The difference between the theoretical and measured OF is displayed in Figure 7.2 together with the mean measured OF. The mean measured OF is the mean of every OF vector in an image which means that every image gets a single value for the OF. The mean error and RMSE between measured and theoretical flow are listed in Table 7.3.



a)



b)

Figure 7.2: a) Mean measured optical flow. b) Difference between theoretical and measured optical flow.

Table 7.3: Root of mean squared error and mean error for the optical flow measurements.

Optical Flow	**Mean Error $\bar{e}$**	**RMSE**
$\dot{r}$	0,17	3,32
$\dot{s}$	2,18	2,99

A difference of zero indicates that the measured OF and correct measurements of roll, pitch and altitude give the correct body-fixed velocity. The mean error is small compared to the measured flow. The RMSE is slightly larger. It is a surprise that the RMSE in r is larger than s, since the measured flow in s is much larger than measured flow in r. It might be a coincidence, but the results indicate that the measured OF is least accurate in r. Figure 7.2 b) supports this statement since the error in r has a higher noise level. Nevertheless Figure 7.2 and Table 7.3 indicate that the measured OF is fairly good. The noise level is quite large in r, but the mean error is small. The mean error in s is above zero. Larger OF in s directly increases the body-fixed longitudinal velocity. Thus the error in longitudinal velocity is related to the error in measured OF.

Discussion

The large RMSE with measurements are likely to be caused by the inclinometer and the GPS. The roll and pitch angle from the inclinometers are very noisy and the altitude from the GPS is not completely reliable. Since the noise level is reduced with estimates, the estimates seem to increase the performance. Thus the results indicate that the inclinometer and GPS reduces the accuracy and that estimates should be used. Nevertheless it is important to use estimates in situations where they have converged.

The assumption of flat terrain in the calculation of velocity from OF is probably an error source. In practice the points in the image plane are not located in the same horizontal plane in the terrain. This is related to the displayed depth which is assumed to be constant in the image. Varying depth in the image leads to OF vectors with different magnitude. Therefore this will directly affect the scale and direction of the calculated velocity. The magnitude of the velocities are larger than the real values and this could be related to depth variations. A more accurate mapping of points in the image plane would be possible with a known terrain profile and might increase the accuracy of the velocity calculation. A completely flat area would most likely increase the performance of the results significantly.

The altitude also directly influences the magnitude of the velocities as illustrated in Example 2.1. The velocity is scaled up if the assumed altitude is larger than the actual altitude. Since the altitude is measured by the GPS, trees and elevations in the terrain give depth variations. Therefore the mean error in u is probably caused by wrong altitude and depth variations. In the observer only the normalized body-fixed velocity is of interest. Therefore the scaling errors disappear and only the direction is of importance. This decreases the dependence on the altitude. The direction of the velocity is not necessarily affected by the noise level since the noise level seems to be related to the magnitude of the velocity components. However the RMSE in v is almost equally large as the RMSE in u. Thus the direction might be slightly wrong since the noise level in v is larger than the noise level in u with respect to the magnitude (signal-noise level).

When the measured and theoretical flow are compared in Figure 7.2 discrete and continuous OF are compared. This is because the theoretical flow is derived in continuous time and the measurement of the OF is given between two consecutive images which is a discrete time difference. Furthermore the mean measured (or estimated) roll, pitch and altitude, in the time between the images, are used in the calculation. As pointed out above, the difference between theoretical and measured flow are larger in r than s. This might be because the UAV has more roll motion than pitch motion and roll motion affect the OF in r. Thus different results might have been the case for maneuvers with a larger amount of pitching motion.

This case study has illustrated that it is possible to calculate body-fixed velocity of a UAV with OF. The accuracy is quite good, even though the transformation is very sensitive to errors in altitude, roll and pitch. Furthermore the flat horizontal terrain assumption seems to be a weakness since flat terrain is hard to find in practice. Thus a transformation without this assumption is desired. This can be achieved by epipolar geometry [34] and is utilized in the paper submitted to AIAA SciTech 2016 (Appendix B). Estimates of roll, pitch and altitude seem to increase the accuracy of the transformation. For the nonlinear observer only normalized velocity is of interest. Therefore the magnitude is of less importance and altitude related problems are not important. This is because the altitude dependency disappears with normalization because every velocity component is directly proportional to the altitude. It is also important to emphasize that the accuracy of the reference not necessarily are perfect. Thus the results must be interpreted with this in mind. Nevertheless EKF is a reliable state estimator and the reference is likely close to the true values.

7.3 Case Study 2: Simulation of the Nonlinear Observer

This case study evaluates the performance of the three different versions of the nonlinear observer presented in Chapter 3. The case study is described in Section 6.3.2. The estimates from the nonlinear observer will be compared with the estimates from the EKF in piccolo. Piccolo is assumed to have estimates very close to the real values. However it is important to remember that the EKF might not be correct at all time instants and the tuning of the EKF (which is unknown) affects the performance. In this case study the results and discussion for the attitude and translational part of the observer will be presented independently. The first version of the nonlinear observer is with camera where the body-fixed velocity is calculated with measurements of roll, pitch and altitude. This version is called the nonlinear observer without feedback. The second version is the nonlinear observer with camera where the body-fixed velocity is calculated with feedback of the estimates of roll, pitch and altitude. The third version is the nonlinear observer with magnetometer instead of the camera.

A reference for the gyro bias is not available. Thus the estimates of the gyro bias are compared with the gyro measurements before take-off when the UAV was at rest. This reference is highly insecure since the bias can be different when the UAV is airborne. The RMSE value and the mean error is calculated from 20 seconds after the start-up until the end. The first time interval is not considered in order to give the observer some time to converge. In the figures this interval corresponds to a time between 1060 and 1310 seconds.

Attitude and Gyro Bias Estimates

This section presents the results and the discussion for the attitude and gyro bias estimates for the nonlinear observer with camera (with and without feedback) and with magnetometer.

Results

Figure 7.3 displays the estimates of the Euler angles for the nonlinear observer without feedback. Table 7.4 describes the mean error and RMSE. Figure 7.4 and Table 7.5 displays the results for the nonlinear observer with feedback. Figure 7.5 and Table 7.6 displays the results for the nonlinear observer with magnetometer.

Table 7.4: Mean error and RMSE in attitude for nonlinear observer without feedback.

Attitude	**Mean Error $\bar{e}$**	**RMSE**
ϕ	1,45	2,29
θ	0,76	1,54
ψ	-1,23	3,73

Table 7.5: Mean error and RMSE in attitude for nonlinear observer with feedback.

Attitude	**Mean Error $\bar{e}$**	**RMSE**
ϕ	1,16	1,85
θ	0,93	1,51
ψ	-0,61	2,83

Table 7.6: Mean error and RMSE value in attitude for nonlinear observer with magnetometer.

Attitude	**Mean Error $\bar{e}$**	**RMSE**
ϕ	1,36	6,82
θ	2,07	8,18
ψ	-10,8	15,1

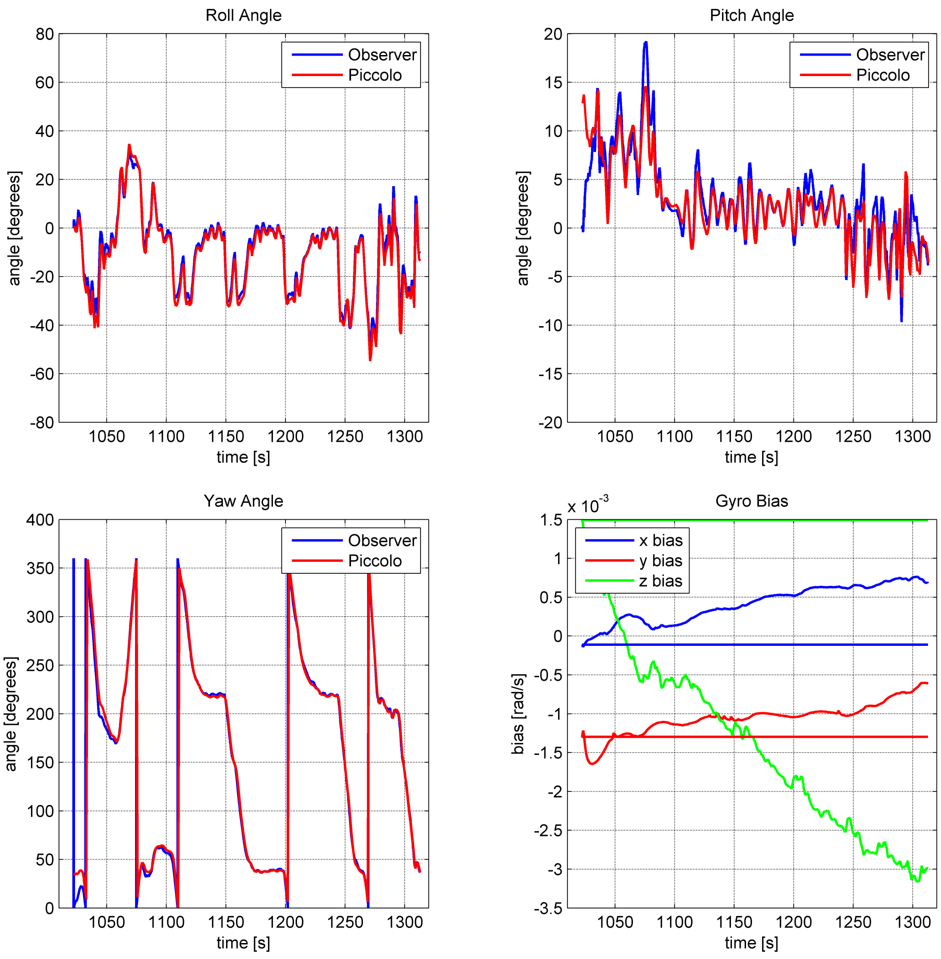


Figure 7.3: Attitude and gyro bias nonlinear observer without feedback.

The estimates of the roll and yaw angle are quite accurate for every version of the nonlinear observer. However the yaw estimate from the nonlinear observer with magnetometer converges significantly later than the other versions. Furthermore the roll angle is less accurate with magnetometer. The pitch estimate converges to the reference with camera, both with and without feedback. The pitch estimate is very close to the reference, even though the magnitude is slightly above the reference at some time instants. The pitch estimate from the nonlinear observer with magnetometer is on the other hand not correct. The magnitude is much greater than the reference, but the estimate seems to follow the trend of the reference sometimes. The estimate seems to be correlated with the yaw estimate and it looks like it is affected by the yaw motion. This is for example visible in Figure 7.5 at the time interval between 1150 and 1170. The yaw changes rapidly in this time

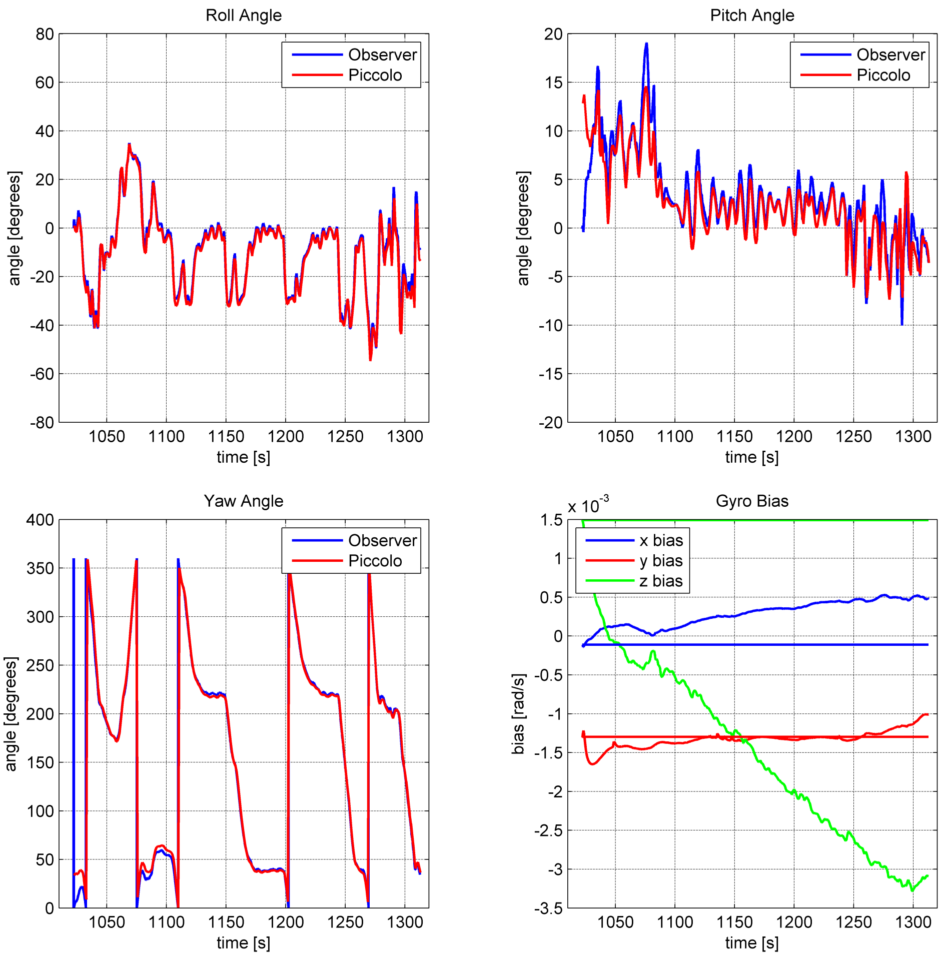


Figure 7.4: Attitude and gyro bias nonlinear observer with feedback.

interval and the pitch estimate does the same. The same behaviour is also visible in the time between 1240 and 1260.

The RMSE for the observer with feedback is smaller than corresponding values without feedback and with magnetometer. The observer with magnetometer has the largest RMSE values for all Euler angles and thus the largest amount of noise. This is especially visible for the yaw estimate which has a RMSE value of 15,1 degrees, which means that the estimate in yaw in general deviates 15,1 degrees from the reference. However the RMSE is calculated in the time interval between 1060 and 1310. Therefore the RMSE in yaw for the nonlinear observer with magnetometer is larger than it is after convergence. The observer without feedback has a limited amount of noise, but feedback increases the accuracy of the esti-

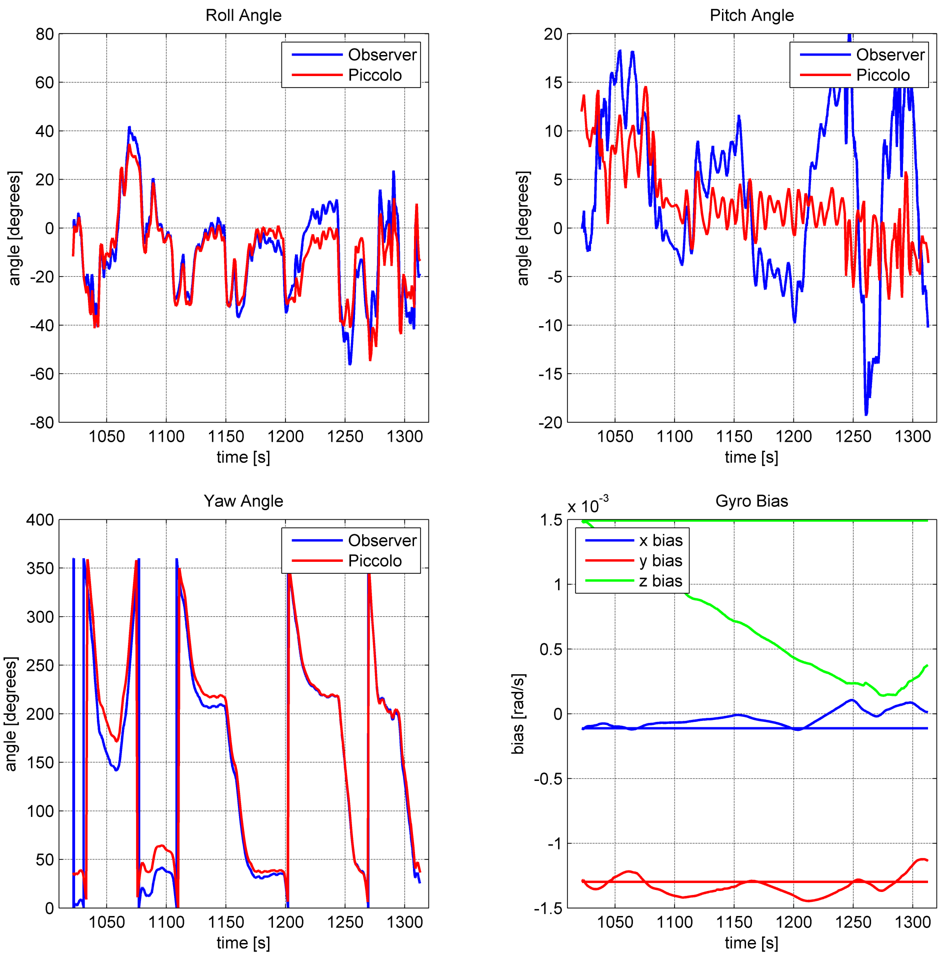


Figure 7.5: Attitude and gyro bias case study 2. Nonlinear observer with magnetometer.

mates.

The estimates of the gyro bias does not converge to a stationary value for any version of the nonlinear observer. Thus it is hard to believe that they are correct. The gyro bias with and without feedback have the same behaviour. The gyro bias estimate from the observer with magnetometer has the least amount of dynamic motion. However this is likely because of different gains in the design of the observer. The true gyro bias is unknown. The reference in the figures is calculated before take-off with the engine off. Thus noise from the engine might affect the bias, and it is hard to say something about the true bias when the UAV is airborne. Nevertheless the results indicate that the estimates of the gyro bias are inaccurate for every version of the observer since they do not converge.

Discussion

The attitude estimator, with and without feedback, is guided by the body-fixed velocity measurement from the camera and the accelerometer measurements from the IMU. The attitude estimator in the nonlinear observer with magnetometer is guided by the magnetometer and accelerometer. The direction of these measurements are used with reference vectors in NED to find the attitude. Each set of vectors provides information about two angles. The direction of the accelerometer is dominated by gravity and thus directed mainly along the body z-axis. Therefore it is a small amount of information about the yaw angle in the accelerometer measurements. The body-fixed velocity is mainly directed forward along the x-axis. Therefore the camera measurement provides little information about the roll angle. Information about the pitch angle should be available in both the accelerometers and the camera measurements. However if these measurements disagree it might decrease the accuracy of the pitch estimate. The magnetometer measures the magnetic field and at the location of the experiment that was directed mainly along the NED z-axis. Therefore the direction of the body-fixed magnetometer measurement is close to the direction of the accelerometer measurement. In practice this means that it can be hard to find three different angles with the magnetometer. That might the case in the results where the nonlinear observer with magnetometer fails to estimate the pitch accurately. Several different observer gains have been tried out, but it seems difficult to estimate both the pitch and yaw accurately with magnetometer.

An important thing to notice is that the order of the reference vectors in the matrix $\mathbf{A}_b$ and $\hat{\mathbf{A}}_n$ in (3.4) and (3.8) can be exchanged. In practice this means that the order of $\mathbf{a}^b$ and $\mathbf{v}^b_{b/n}$ ($\mathbf{m}^b$) can be switched. The most trustworthy measurement should be placed first because this measurement will be weighted the most in the injection term. In this case that is considered to be the accelerometer. Both orders have been tried out and the results were most accurate with the accelerometer as the first vector.

Case study 1 shows a large noise level in the measured body-fixed velocity from the camera, and this might decrease the accuracy of the attitude estimates. This is likely since the estimates are more accurate with feedback and the body-fixed velocity from the camera was also more accurate with feedback. Nevertheless in situations where the estimates are far off the true value the feedback will reduce the performance. Another error source might be accelerometer bias. Accelerometer measurements without bias are assumed, but a small bias in the accelerometer is likely. Thus this will also decrease the accuracy of the estimates. Vibrations caused by the engine are also something that could affect the estimates.

In the nonlinear observer with magnetometer the roll angle is the most accurate estimate in the beginning and the yaw converges much less rapid. This indicates that the accelerometer is more accurate than the magnetometer measurements. The pitch is far off, and this might be because of wrong calibration of the magnetometer. However it is still important to remember that the direction of the magnetometer is close to the accelerometer direction compared with the camera, and this can also be the reason for the inaccurate pitch estimate. The attitude estimates from the nonlinear observer with magnetometer are the least accurate. In addition to the direction issue, the magnetometer might by exposed to noise from other electrical components in the payload. The camera trigger might be such an error source because it is triggered by the GPS. Thus for reliable magnetometer measurements in a UAV (with limited space) proper shielding of the magnetometer is necessary. The reference vector for the magnetometer measurement is the magnetic field at the location of the experiment in NED. The reference vector at the location of the experiment is calculated after the experiment with a magnetic field calculator. A manual measurement of the magnetic field in NED at the location of the experiment might be better than offline calculation. That is something that could be considered for later experiments.

Position and Velocity Estimates

This section presents the results and the discussion for the translational part of the nonlinear observer with camera (with and without feedback) and with magnetometer.

Results

Figure 7.6 displays the estimates of the position in NED for the nonlinear observer without feedback. Table 7.7 describes the mean error and RMSE. Figure 7.7 and Table 7.8 displays the results for the nonlinear observer with feedback. Figure 7.8 and Table 7.9 displays the results for the nonlinear observer with magnetometer.

Table 7.7: Mean error and RMSE in position for nonlinear observer without feedback.

Position	**Mean Error $\bar{e}$**	**RMSE**
N	-1,14	8,16
E	-0,73	7,34
Altitude	0,26	1,13

Table 7.8: Mean error and RMSE in position for nonlinear observer with feedback.

Position	**Mean Error $\bar{e}$**	**RMSE**
N	-1,15	8,15
E	-0,72	7,33
Altitude	0,26	1,13

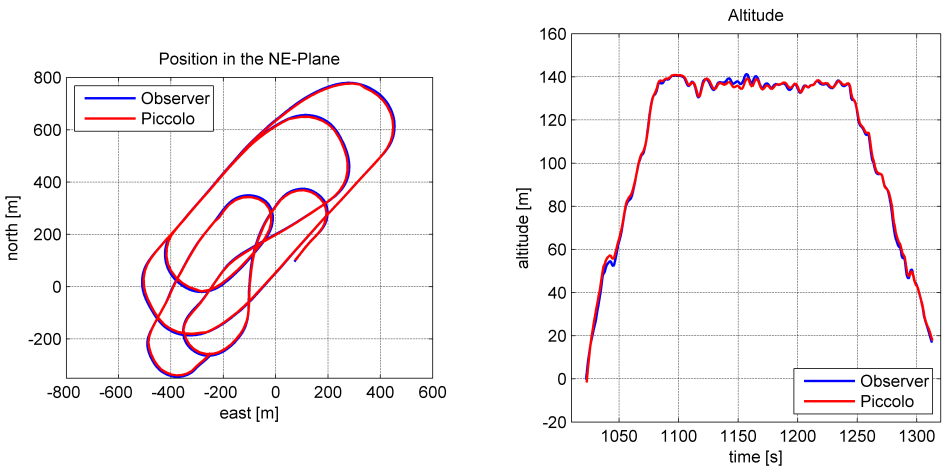


Figure 7.6: Position estimates nonlinear observer without feedback.

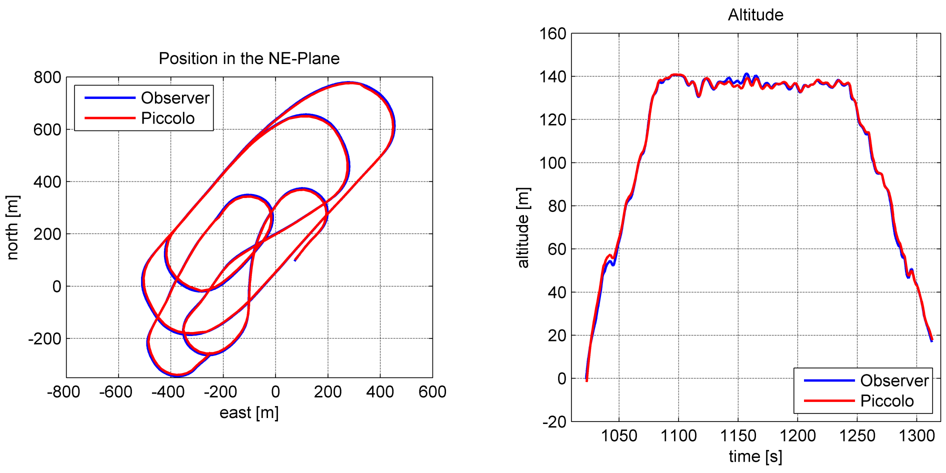


Figure 7.7: Position estimates nonlinear observer with feedback.

Table 7.9: Mean error and RMSE in position for nonlinear observer with magnetometer.

Position	**Mean Error** $\bar{e}$	**RMSE**
N	-1,08	8,14
E	-0,58	7,26
Altitude	0,27	1,16

The path in the NE-plane and the altitude are estimated accurately for every

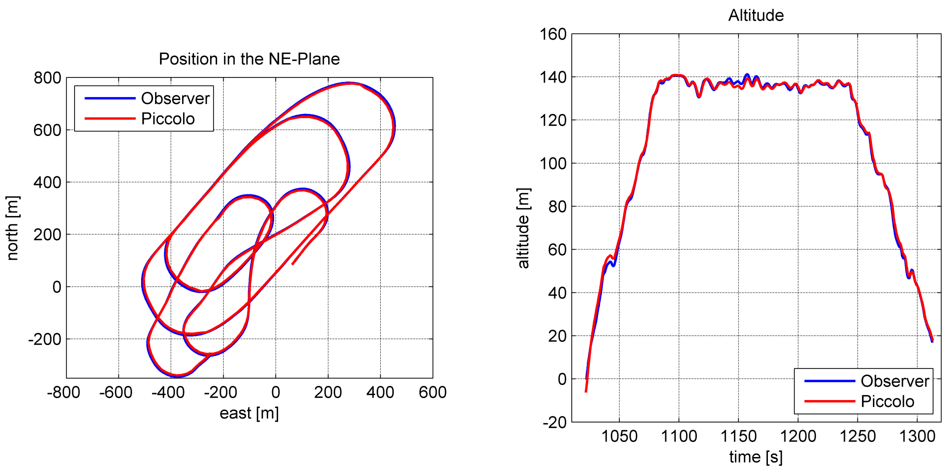


Figure 7.8: Position estimates nonlinear observer with magnetometer.

version of the observer. The estimated path in the NE-plane follows the reference closely. The altitude is also estimated accurately. The RMSE values are almost equal for every part of the observer. The error are approximately eight meters in north and east position and one meter in altitude. The estimates are considered to be reliable for every version of the observer.

Figure 7.9 displays the estimates of the velocity in NED for the nonlinear observer without feedback. Table 7.10 describes the mean error and RMS. Figure 7.10 and Table 7.11 displays the results for the nonlinear observer with feedback. Figure 7.11 and Table 7.12 displays the results for the nonlinear observer with magnetometer.

Table 7.10: Mean error and RMS value in velocity for nonlinear observer without feedback.

Velocity	**Mean Error $\bar{e}$**	**RMS**
N velocity	-0,15	1,19
E velocity	-0,16	0,98
D velocity	-0,01	0,35

Table 7.11: Mean error and RMS value in velocity for nonlinear observer with feedback.

Velocity	**Mean Error $\bar{e}$**	**RMS**
N velocity	-0,15	1,19
E velocity	-0,15	0,98
D velocity	-0,01	0,34

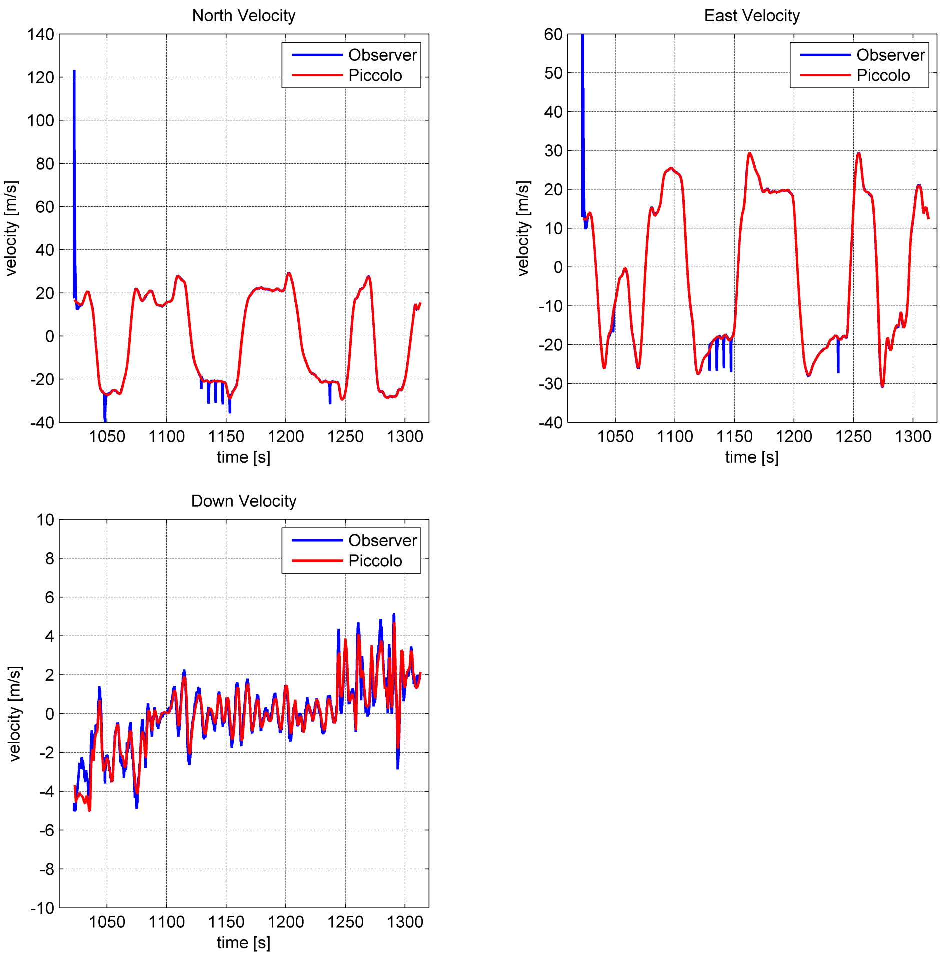
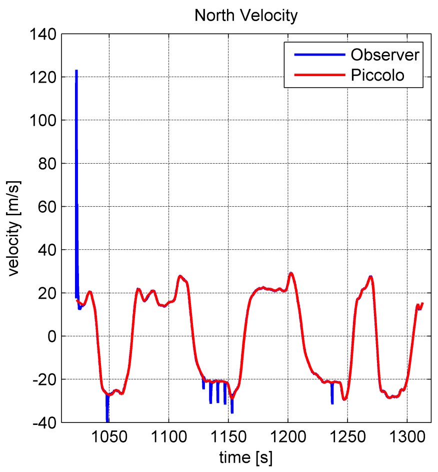


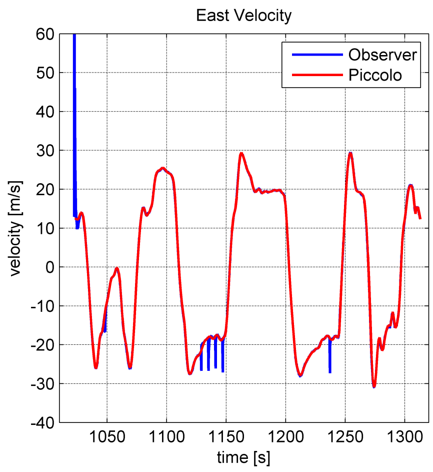
Figure 7.9: Velocity estimates case study 2. Nonlinear observer without feedback.

Table 7.12: Mean error and RMS value in velocity for nonlinear observer with magnetometer.

Velocity	**Mean Error** $\bar{e}$	**RMS**
N velocity	-0,13	1,18
E velocity	-0,07	0,97
D velocity	0	0,35

The velocity in NED is also estimated accurately. The estimates follow the reference closely and the RMS values are low compared to the actual velocity. The RMS values in north and east are close to 1 m/s and the velocity in these directions are typically 20 m/s. The RMS values in down are approximately 0,4 m/s. Therefore





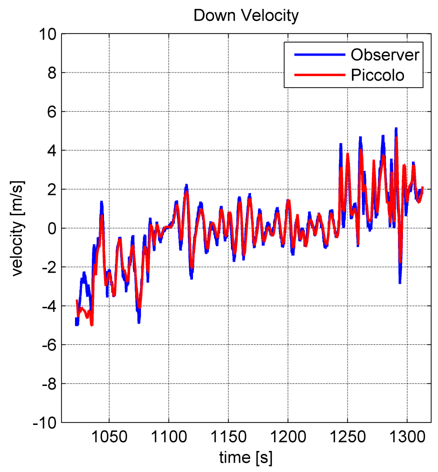


Figure 7.10: Velocity estimates case study 2. Nonlinear observer with feedback.

the errors are small compared to the actual velocity.

Discussion

The position estimates are equally good for every version of the observer. That is not a surprise since the observer use the same measurements of position and velocity. Furthermore the translational part of the observer is tuned with the same gains. Thus only marginal differences should be visible and that is the case. An error of eight meters in north and east directions is quite accurate. However the data sheet of the GPS give an error in North and East direction within 2,5 meters.

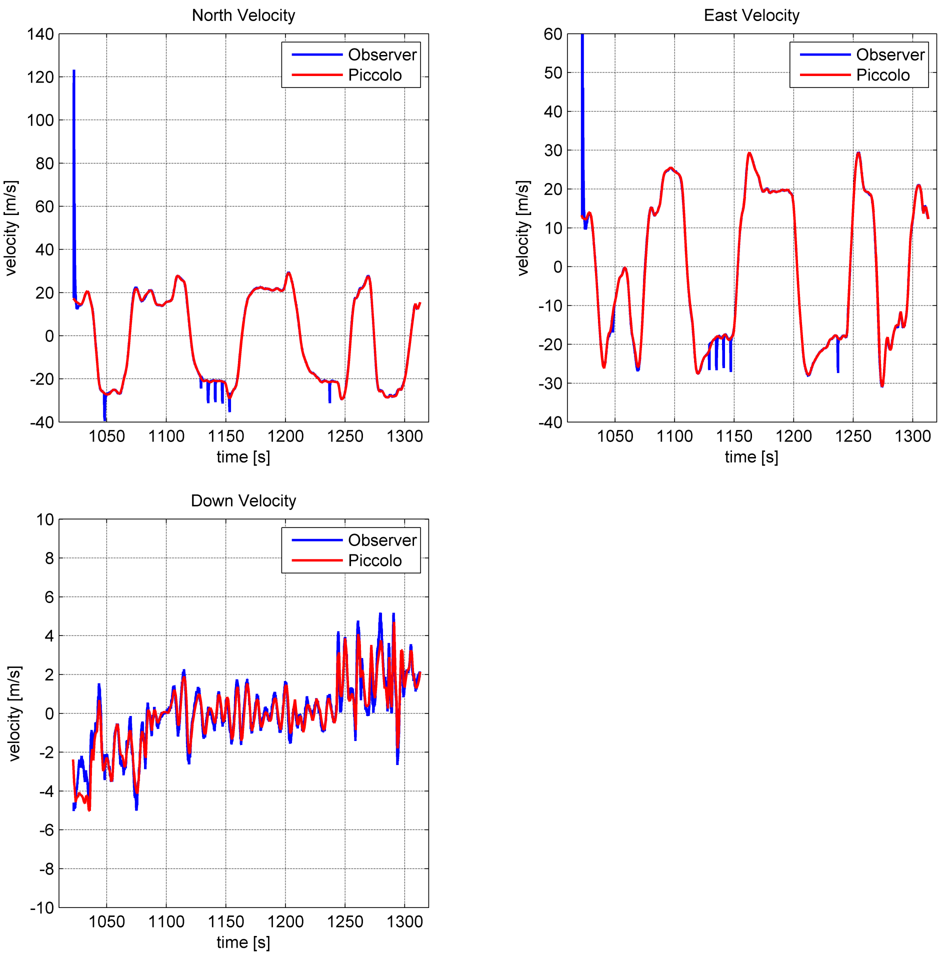


Figure 7.11: Velocity estimates case study 2. Nonlinear observer with magnetometer.

This might imply that the GPS measurement of position is more accurate than the estimate. It is not possible to discuss this further since the reference is highly insecure when errors of an magnitude of eight meters are considered. Therefore the RMSE in position and velocity are just an indication of the noise level and can not be interpreted as true values. Piccolo uses the same GPS antenna, but has its own receiver and thus different measurements. The observer gains are chosen such that the GPS is heavily trusted. Thus errors in the GPS might be a problem. However in areas where GPS are less reliable the gains could have been changed. Therefore it is not a weakness of the observer, but rather something to notice.

Some spikes in the north and east velocity are visible for every version of the observer. Four of them occur in the time interval between 1130 and 1160 seconds.

These spikes are caused by erroneous GPS measurements. The velocity measurement is calculated by differentiating the position measurements from the GPS. Therefore errors in the position measurements will heavily affect the velocity measurement. That is the case at the spikes. Since the translational part of the observer is tuned to trust the GPS very much, the spikes are larger than they would be with smaller gains in the translational part of the observer ($\mathbf{K}_{vp}$ and $\mathbf{K}_{vv}$). A solution to reduce the problem might be to compare the velocity measurements with the previous reliable measurements and discard erroneous measurements. However this would require a filter that decides what a reliable measurement is. Furthermore it is necessary to know that the first measurements are reliable which is hard online. It could increase the performance if just single non-consecutive measurements are erroneous. Such a solution is not implemented in this thesis, but could possibly remove the spikes.

Overall Performance for the Nonlinear Observer

This case study has evaluated three different configurations for the nonlinear observer presented in Chapter 3. The attitude estimates in the observer with a camera, both with and without feedback, resulted in reliable and accurate estimates. The noise level was reduced with feedback which indicates that feedback should be used, at least after convergence of the estimates. Therefore an experimental validation for this configuration of the observer have been conducted. A stability proof for the feedback of roll, pitch and altitude should be derived in the future. The feedback of roll and pitch is considered to be the most important factor and thus feedback from the attitude estimator should be first priority. This was pointed out in Case Study 1 where it was argued that the altitude was less important because the velocities are normalized and every component of the velocity are proportional to the altitude. The nonlinear observer with magnetometer did not manage to provide accurate estimates of the attitude. Tuning parameters that resulted in convergence for both yaw and pitch where not identified, even with a large amount of time dedicated to find appropriate values. This might be because the magnetometer measurement is directed mainly in the same direction as the accelerometer measurement and because the magnetometer is weak for interference from other electrical components. It is hard to shield the magnetometer in the payload and thus is assessed to be a viable alternative to the magnetometer.

The position and velocity estimates were almost equal for the different nonlinear observers. This was expected since the translational part of the observer consists of the same equations and gains. Thus only minor differences related to deviations in estimated attitude were expected. In addition the translational part of the observer was tuned to trust the GPS heavily for every configuration. An important consideration is that the the chosen tuning is weak for erroneous measurements from the GPS. Thus it might be necessary to filter the GPS measurements or change the tuning if a less accurate GPS receiver is utilized. The velocity was measured by differentiation of the position. GPS receivers based on the Doppler shift are much more accurate and can be used to increase the reliability.

This case study has also shown the weakness of the magnetometer. Thus the camera is a viable alternative for attitude estimation even though the calculation of body-fixed velocity with a camera was quite noisy in case study 1. An interesting possibility is to combine the measurements from the camera and the magnetometer to estimate the attitude more accurately. A combination of these measurements can be based on identifying the most reliable measurement at each time instant or creating a new measurement that depends on both the magnetometer and the camera. If the accuracy of the body-fixed velocity calculation from the camera can be increased, the camera can be very useful for dead reckoning. Even with the demonstrated accuracy in this case study, the camera should be useful for dead reckoning.

7.4 Case Study 3: Moving Object Detection

This case study evaluates the moving object detection algorithm described in Section 4.2. The case study and the applied test set of images are described in Section 6.4.1. The performance of the algorithm will be evaluated with the following criteria

- Detection Rate - The number of images where the moving object is detected compared to the total number of images with the moving object.
- False Positive Detections - The number of false detections.
- SIFTs ability to find a feature on the object.
- The number of OF vectors classified as belonging to a moving object compared to the true number.
- The number of OF vectors wrongly classified as belonging to a moving object.

Results

The results of the case study with the different tuning parameters are described in Table 7.13. Figure 7.12 illustrates two images where the object detection algorithm successfully detected the moving object. One of the images also includes a false detection, which is the detection of a non-existing moving object. Figure 7.13 shows an example of detected OF vectors for two images in the image plane.

Tuning 1 is later referred to as the default tuning, while tuning 2 is the soft tuning

Table 7.13: Results Case Study 3.

Evaluation criteria	**Tuning 1**	**Tuning 2**	**Tuning 3**
Number of images where SIFT found OF vector on moving object	82,4 %	82,4 %	82,4 %
Number of OF vectors on moving object	3,6%	3,6%	3,6%
Number of OF vectors correctly classified as belonging to moving object	98,1%	98,8%	95,1%
Number of OF vectors not classified as belonging to moving object (wrongly)	1,9%	1,2%	4,9%
Number of OF vectors wrongly classified as belonging to a moving object	0,06%	0,17%	0,05 %
Number of images moving object detected with OF vectors on object	97,5%	99,5%	92,3 %
Number of images moving object not detected with OF vectors on object	2,5%	0,5%	7,7%
False Positive Detections	19	33	18
Detection Rate	80,3%	81,9 %	76,1 %

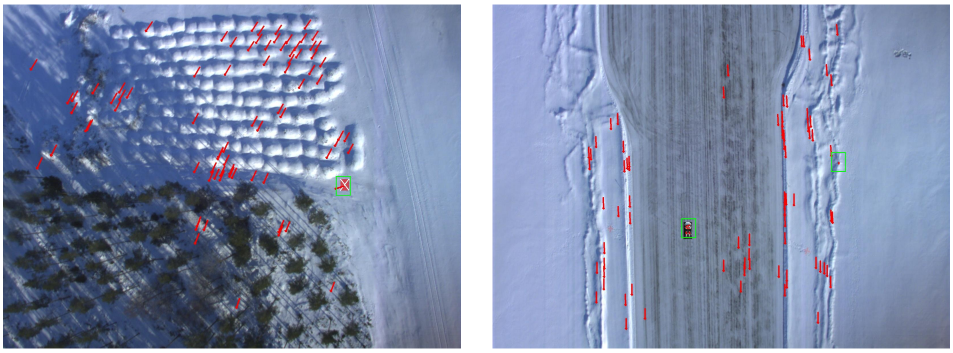


Figure 7.12: Example of segmentation case study 3. The green rectangle marks an area where a moving object is detected. The image to the right contains a rectangle where a moving object wrongly is detected.

and tuning 3 is the hard tuning. SIFT is able to find OF vectors on the moving object in 82,4% of the images with the object. The mean number of OF vectors on the object is 7,7 when SIFT is able to find a feature on the object. The moving object detection algorithm is very reliable when OF vectors exist on the object. It correctly locates 95,1% - 98,8% of the OF vectors on the moving object. The number of OF vectors wrongly classified as belonging to a moving object is negligible (maximum 0,17%) compared to the total number of OF vectors. Thus the false detection rate is very low. Since only one wrongly classified vector is necessary before a false detection, the number of false positive detections are ranging from 18

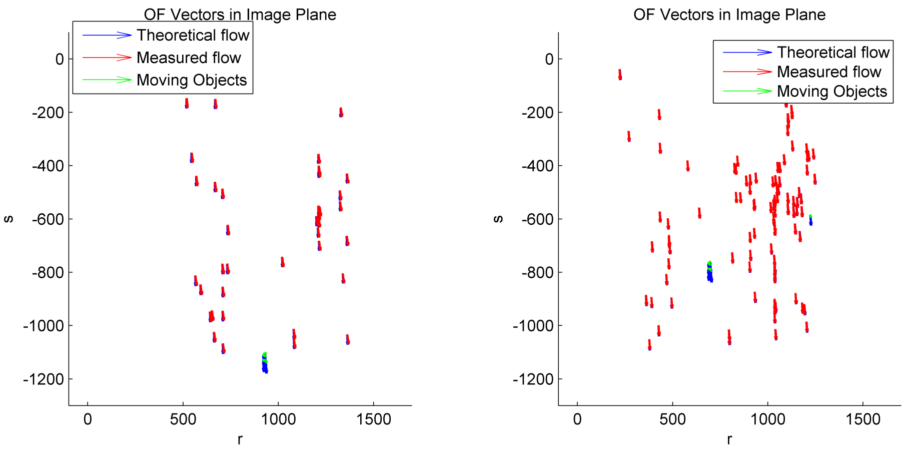


Figure 7.13: Example of OF vectors in the image plane. The green vectors are classified as belonging to a moving object. The red vectors are the OF vectors measured by SIFT and the blue vectors are the theoretical OF. The second image (to the right) has one vector wrongly classified as belonging to a moving object on the right hand side.

to 33. The algorithm is able to find the moving object in almost every image with the default tuning (97,5%) when a OF vector is present on the object. However since SIFT only finds features on the object in 82,5% of the images in the test set, the total detection rate is 80,3% with the default tuning. Thus the main challenge with the algorithm is for SIFT to find the object.

Discussion

SIFT has the poorest performance when the object is homogeneous and if it hard to separate it from the background. Large amount of sun and shadows can be a challenge since it is so easy to find features in the image on other places than the object. However large amount of sun might also increase the contrast on the image which is an advantage. Features are found when the intensity gradient in the image is large. SIFT had much larger problems of finding features on the red rectangle than the black car. That is mainly because the car has a larger amount of contrast. The red rectangle is too homogeneous, especially since the boundary has a constant color intensity. Every image where SIFT was unable to find a OF vector on the object, contained the red rectangle and not the black car. Thus it was much easier to find features on the black car, which is an image. This implies that it could be easier to detect real objects than the manually created red rectangle. However small homogeneous objects, such as small cars with the same color as the environment, would increase the difficulty of finding features on the object.

The results are fairly similar with the different tuning parameters. Larger differences could possibly be detected with a larger and more realistic test set, which would be beneficial in the future. The hard tuning is created to achieve the least amount of false detections. However the number of false detections compared to the default tuning is almost equal and the detection rate is lower for the hard tuning. Thus the default tuning works better than the hard tuning on the test set. The soft tuning has the largest number of false positive detections as expected. It also has the highest detection rate which is anticipated because it is created to achieve the largest detection rate. Therefore it could be beneficial to use the soft tuning if it is more important to find the moving object than limiting the number of false positive detections.

The algorithm works very well on the test set. Since the algorithm is based on the difference between measured and theoretical OF, it is important that moving objects have a velocity sufficiently larger than zero in order to separate them from the background. Furthermore the difference needs to be greater than the noise level. In practice this means that a difference of several pixels is necessary. The algorithm is evaluated on images with a resolution of 1600×1200. Smaller resolution would lead to smaller OF vectors in magnitude, and increase the difficulty of finding features on the object and separate them from the background. Therefore smaller resolution would require objects with a larger velocity. It is beneficial to process the images on the largest resolution, even though it also increases the computational load.

The case study has shown that it is possible to detect moving objects with OF and that it works very well when the objects have a velocity greater than 30 km/h. Therefore it is possible to detect cars and other vehicles moving with large velocity. However OF is not appropriate for detection of humans and vehicles with a small velocity from a UAV. This is because the noise level is relatively large, and it would be hard to separate the object from the background because the UAV moves with a much larger velocity than slowly moving objects. The noise level of the OF vectors were shown in case study 1 (Figure 7.2). OF can on the other hand be used for detection of slowly moving objects if the camera is at rest. The main challenge with the algorithm is that SIFT needs to locate at least one feature on the object. This is only possible with sufficient contrast on the object.

7.5 Case Study 4: Tracking of Single Object

This case study evaluates the tracking system described in Section 4.5 on a single moving object, in this case study a black car. The case study is described more closely in Section 6.5.1. The tracking system tracks objects detected by the moving object segmentation algorithm evaluated in Section 7.4. The true path is known and referred to as the reference.

Results

The estimated position in the NE-plane is displayed in Figure 7.14. The calculated velocity through first order differentiation is displayed in Figure 7.15. Figure 7.16 illustrates the tracking error in position and velocity. The tracking error in pixels is displayed in Figure 7.17.

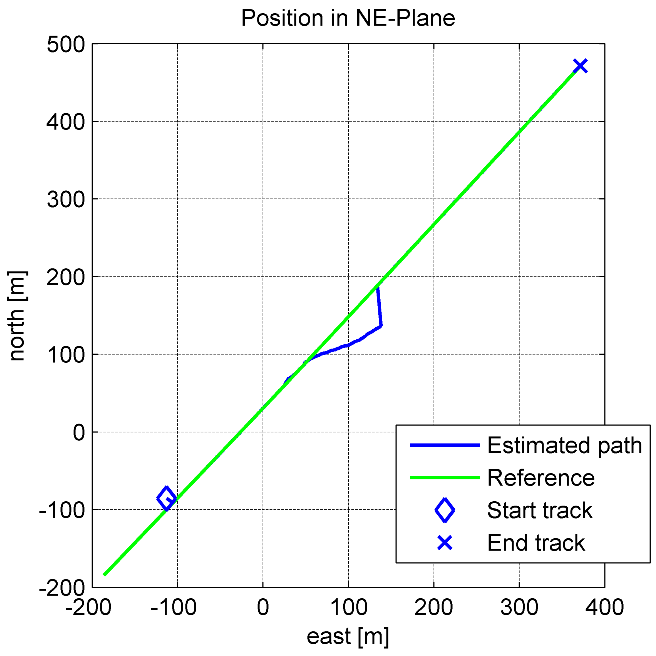


Figure 7.14: Estimated and real position.

The position estimate in Figure 7.14 follows the real path closely when the tracking system has lock on the black car. Lock is considered to be a situation where the moving object is detected in current or previous image. This also illustrated by Figure 7.16 where the error in North and East position are displayed. The error in estimated position is close to zero for most of the time. The tracking system looses lock on the car in the time interval between 1180 and 1185 approximately, and thus the estimate deviates from the true path in that time interval. The tracking system looses lock because the car is outside of the field of view of the camera (not in the image) in that time interval. This is illustrated in Figure 6.8 in the description of the case study (Section 6.5.1). The tracking error in position increases rapidly because the car accelerates in the time interval where the system looses lock. A constant velocity model is used in the Kalman filter. In practice this means that tracked objects are assumed to have constant velocity or at least a very small acceleration in the image plane. That is not the case when the system looses lock on the car. The estimated velocity in NED is calculated by first order

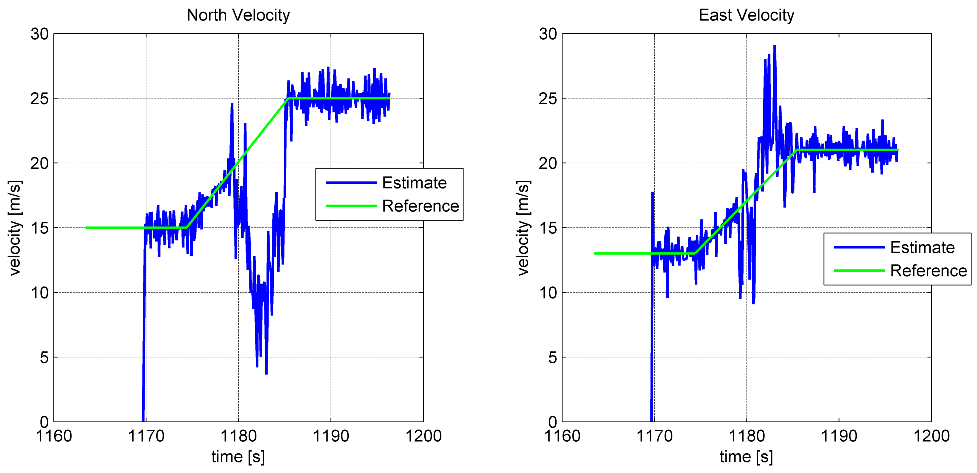


Figure 7.15: Estimated velocity. It is calculated by first order differentiation.

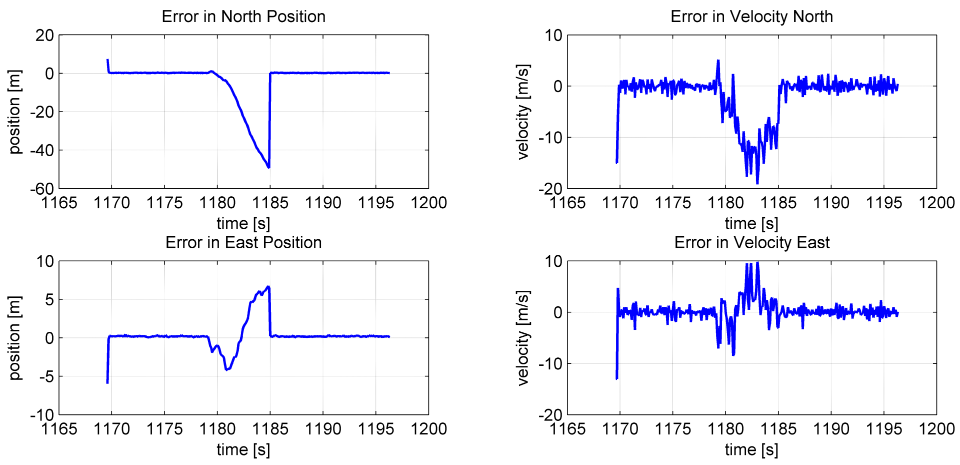


Figure 7.16: Tracking error position and velocity.

differentiation and thus the noise level is quite high in Figure 7.15. Nevertheless the estimated velocity follows the trend of the real velocity. The error in velocity is displayed in Figure 7.16 where the error increases rapidly when the tracking system looses lock. The mean error is close to zero when the tracking system has lock on the car. The down position of the car is known by the altitude of the UAV and not interesting since it (hopefully) stays on ground level.

The estimation error in pixels displayed in Figure 7.17 shows an estimation error close to zero for most of the time. The error increases rapidly when the system

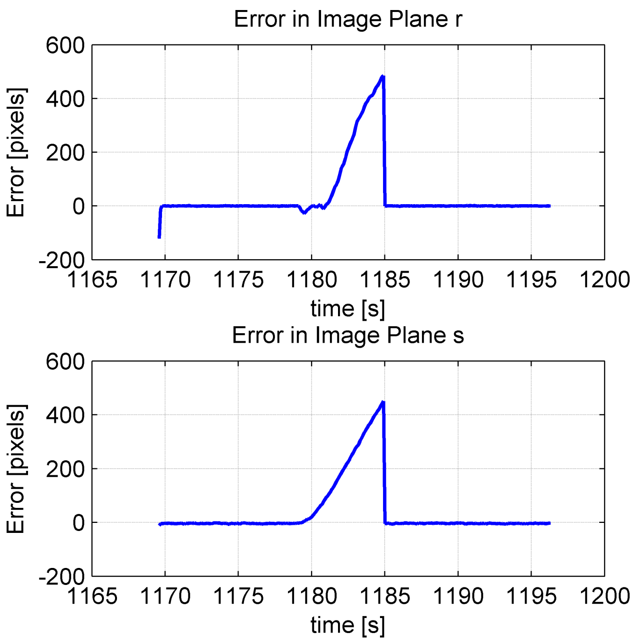


Figure 7.17: Estimation error in pixels

looses lock on the object. The mean tracking error is very close to zero in r, but slightly below zero in s. The mean tracking error in s is approximately minus five pixels when the system has lock on the car. This is due to the fact that the OF vectors are not uniformly distributed on the car. The mean position of the OF vectors on the car is used as the position measurement. Evidently the mean of the OF vectors is slightly displaced from the true center in. Furthermore the noise on the position measurement of the OF vectors are not solely white since SIFT usually finds features on the same locations on the car. Therefore the estimated pixel position will be slightly displaced from the true center when the OF vectors are not uniformly distributed around the true center. The estimation error in pixels would be closer to zero if the noise on the position measurement was white. However a single pixel captures a real distance of approximately 10 cm at an altitude of 150 meter (Height Captured in Table 6.3). Thus an error of five pixels is negligible in practice.

Discussion

The position estimate is very accurate and reliable as long as the car is detected. Another motion model could have been used to increase the accuracy of the estimate when the car accelerates. A constant acceleration model could have been used instead of the constant velocity model. This would be interesting to work with in the future and should increase the performance of the tracking system. However the main issue is the movement of the UAV which leads to changes in the image plane. A constant attitude for the UAV would increase the performance significantly since small changes in attitude can lead to large changes in the image plane, especially at lower altitudes. Nevertheless the performance is clearly good enough with lock on the object. Lock on the object is achieved in almost every image where the car is visible. Therefore the moving object segmentation algorithm works as intended in this case study where the car moves with large velocity. Tracking of a car with constant velocity will be a part of case study 5 in Section 7.6.

A possible weakness with the tracking system could be in case of misdetections. Misdetections are situations where the car wrongly is detected somewhere far of the true location. The Kalman filter is tuned with small noise on the measurements. Therefore the measurements are heavily trusted and wrong measurements would lead the estimate away from the true path. On the other side the correct path would be reached rapidly whenever accurate measurements becomes available again with the chosen tuning. The Kalman filter could be tuned differently if several objects have the same appearance or at least look quite similar. This could be the case for several cars tracked from the air because the color of the roof is the most visible attribute. Another approach is to locate erroneous measurements through the RANSAC algorithm [48, Chapter 10], but it requires storage of previous measurements. RANSAC locates inappropriate points from a set of points by finding the best model for the majority of the points. Therefore it requires that most of the measurements are valid which is a reasonable assumption.

The problem illustrated in the previous paragraph is related to the classification of objects. The tracking system uses template matching to distinguish objects. The template is updated when new measurements are reliable and the classifier is therefore time varying. Nevertheless rapid changes in the appearance of the object might cause problems because previously detected objects might be considered to be new objects. It is not an issue in this case study because the object has the same appearance in every image. In practice though the update of the template could be crucial, especially if the object rotates with respect to the camera or changes appearance because of different light conditions for example. This is the main reason for implementing a time varying classifier for objects in the tracking system. If the object changes its appearance so rapidly that the tracking system considers it to be a new object, two paths of the same object are updated. One of the paths will be updated by new measurements while the old path will be solely updated by prediction. The old path will be removed after a while if new detections are absent. In order to get the complete path of the object an algorithm for combining resembling paths could be created. This can also be achieved with the RANSAC algorithm by comparing the old path with the first points in the new path. This can be very useful if the object is exposed to occlusion and then looks slightly different after the occlusion.

The velocity estimate is quite noisy. This is, as mentioned already, because it is based on the difference between the two most recent position measurements. Such an update of the velocity was chosen for simplicity in this thesis because the position estimate was considered to be the most interesting state. However it is possible to utilize OF and the velocity of the UAV through the M-matrix to find the velocity with (2.25). It is necessary to assume that the angular velocity of the object is zero and thus only the rotation of the camera matters. One could then use the estimated OF with the angular velocity of the UAV (measured by the IMU) to find the velocity of the object in the camera-frame by equation (2.25). It is necessary to assume zero velocity in down direction to limit the system to two unknowns (u and v). By using the body-fixed velocity of the UAV they can be added together to find the velocity of the object in the camera frame with respect to the ground. It is then possible to find the velocity in NED by calculating the rotation matrix with the Euler angles. For this to work a state estimator for the UAV, such as a nonlinear observer, is necessary.

This case study has shown that the tracking system is able to track a single object very accurately as long as the object is visible in the images captured by the camera. When the object is visible in the images, the object detection algorithm is able to find the position and velocity in the image plane and track the object with high performance. The estimates are converted to position and velocity in NED. The velocity estimate is quite noisy since it is calculated by first order differentiation in the image plane. When the object is outside the field of view of the camera the tracking system predicts the motion of the object. The predicted position is quite inaccurate since the velocity in the image plane varies rapidly. This is because the car accelerates and that the attitude and velocity of the UAV vary. Thus the constant velocity model should be evaluated and most likely replaced by a model with higher order. A new model could for instance be based on constant jerk in the image plane with the same measurements. This will be discussed further in the results for Case Study 5 in section 7.6.

7.6 Case Study 5: Tracking of Multiple Objects

This case study evaluates the tracking system described in Section 4.5 on two moving objects. The objects are a black and white car. The case study is described more closely in Section 6.5.2. The tracking system tracks objects detected by the moving object segmentation algorithm evaluated in Section 7.4.

Results

The estimated position and the real position in the NE-plane is displayed in Figure 7.18. The calculated velocity for the first object is displayed in Figure 7.19 and for the second object 7.20. Figure 7.21 and 7.22 illustrates the tracking error in position and velocity for object 1 and object 2, respectively. The tracking error in pixels for both objects are displayed in Figure 7.23.

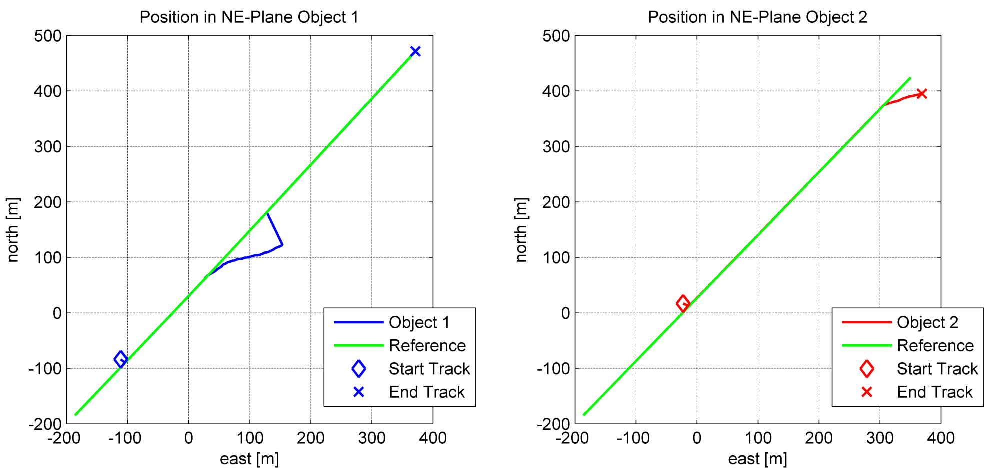


Figure 7.18: Estimated and real position for both objects.

The position estimates in Figure 7.18 show that the estimated paths follow the real path when the objects are visible in the image. It is important to note that the references for the first and second object are in fact different, but looks very similar because the distance between the cars are short. The first object has a short period where it is outside of the field of view of the camera and the position estimate drifts away from the true path. The second object stays in the image continuously from the first detection until the last detection. The time intervals where the objects are in the field of view of the camera are displayed in Figure 6.10. The second object is not visible in the end of the tracking period and the estimated path drifts away from the true path. The position estimates deviate rapidly from the true paths when position measurements for the objects are unavailable for several consecutive images. This is mainly because the velocity in the image plane varies much because of the UAV motion. A small roll motion would for example move the object significantly in the image plane. Therefore the constant velocity model is not the most appropriate model it seems. Nevertheless the estimated paths are very reliable when the objects are visible in the image and the performance is in general satisfactory.

The velocities in Figure 7.19 and 7.20 are calculated by first order differentiation. The noise level is very similar for both objects. The mean velocity is very close to the real value and the velocity calculation works pretty well. If the goal is to estimate the velocity more precisely, another method than first order differentiation should be used. Another method for calculating the velocity, utilizing the velocity in the image plane, is shortly discussed in Case Study 4 (Section 7.5).

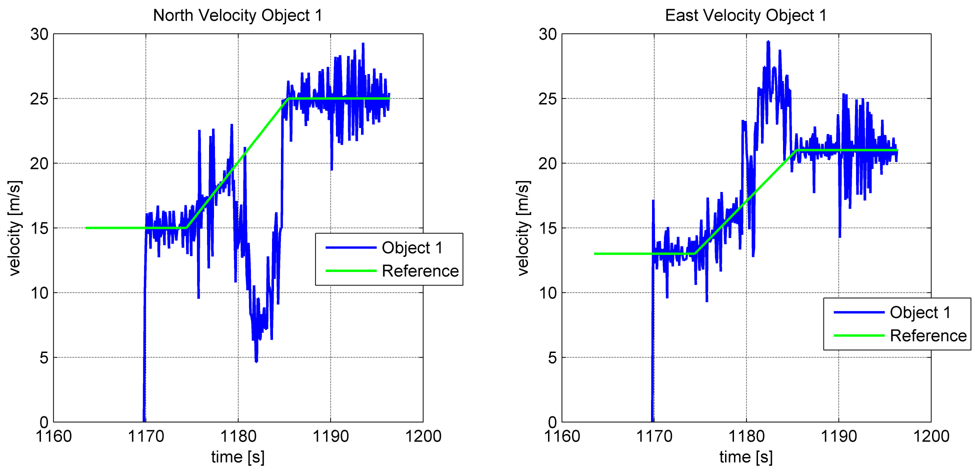


Figure 7.19: Estimated velocity object 1.

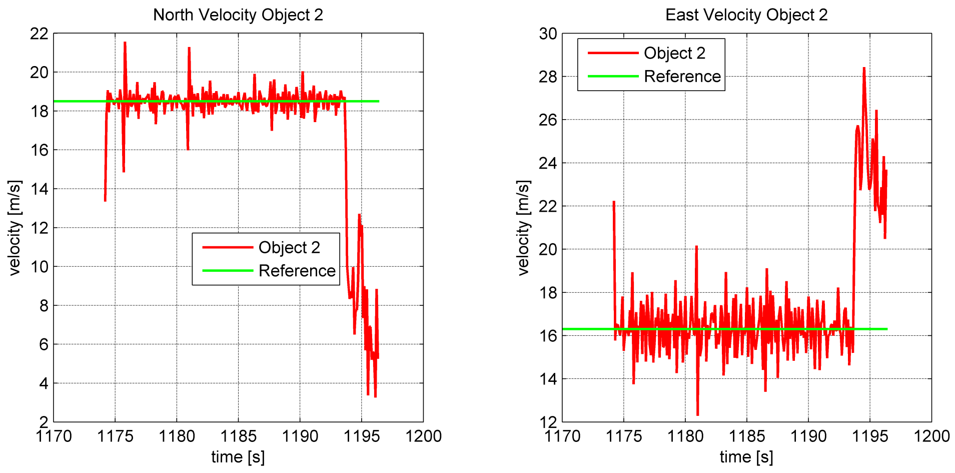


Figure 7.20: Estimated velocity object 2.

The estimation error in position and velocity for both objects, in Figure 7.21 and 7.22, show that the estimates are very accurate (as long as position measurements are available). The error in position is close to zero for both objects. The second object has a small offset in North position which leads a constant error slightly above zero. This is simply because the OF vectors on the second object are displaced from the center of the car. Thus the measured position is displaced slightly form the true position, but only a meter. The accuracy is satisfactory as long as the objects are visible. The estimation error grows rapidly when the objects are outside of the field of view of the camera. It is visible in North position for object

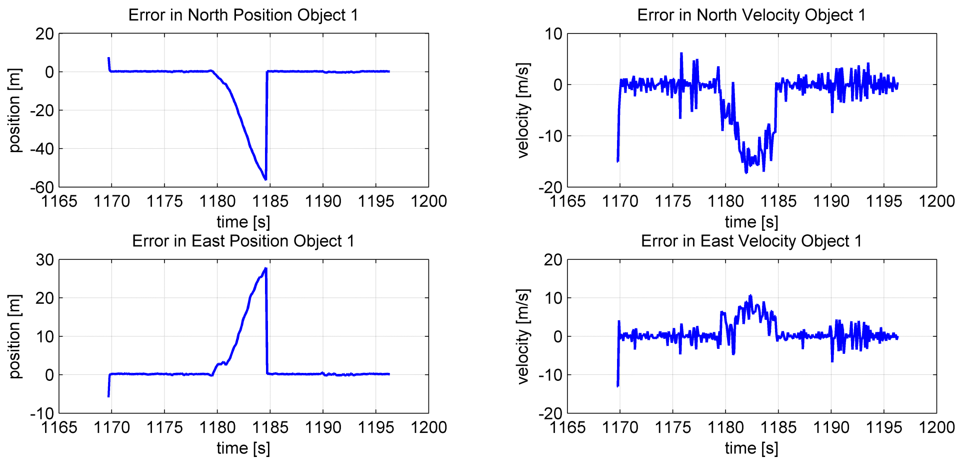


Figure 7.21: Estimation error position and velocity object 1.

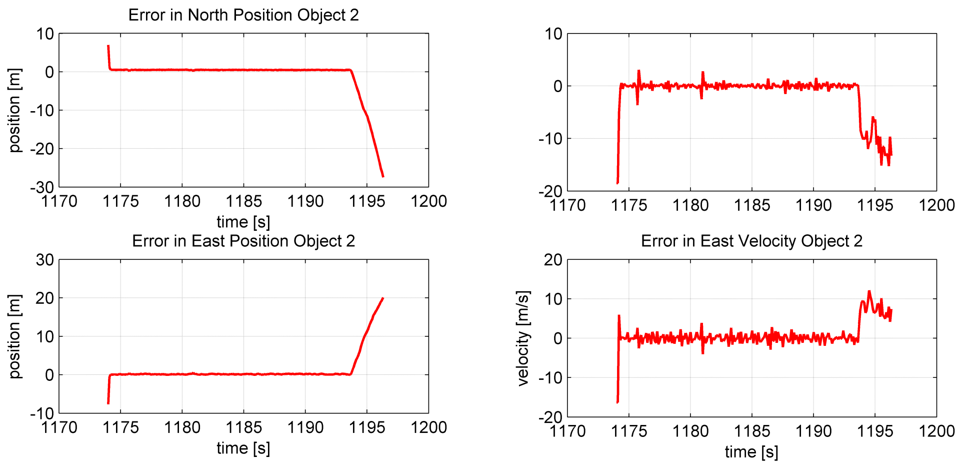


Figure 7.22: Estimation error position and velocity object 2.

1 where the error grows to 50 meters in five seconds when the object is outside the field of view of the camera. 50 images are captured in five seconds. Thus the error is approximately one meter per image. This is not that much, but the error grows rapidly without measurements of the position. The error in velocity is close to zero for both objects as long as they are visible. The noise level is approximately 2-3 m/s. The RMSE value can probably be reduced by a more accurate method for calculating the velocity.

The estimation error for the position in the image plane in Figure 7.23 shows that

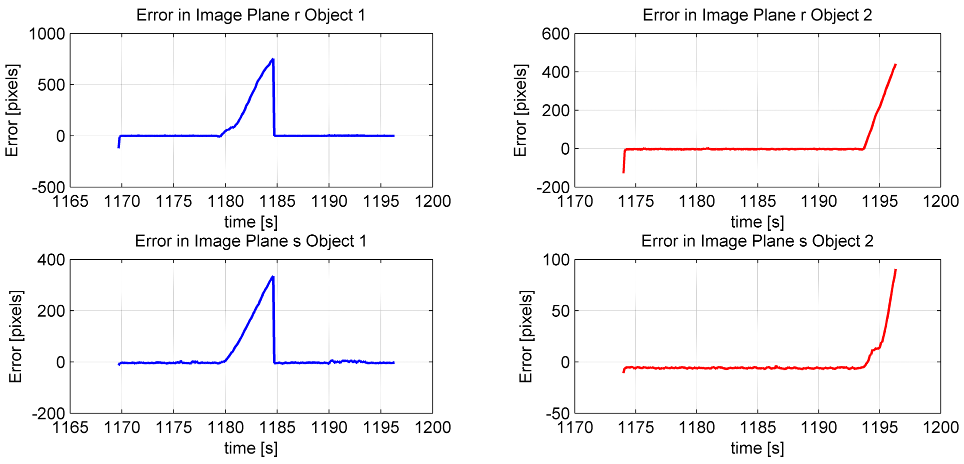


Figure 7.23: Estimation error in pixels for both objects.

both objects are tracked accurately when they are visible. The estimation error in r grows much faster than the estimation error in s when the objects are outside the field of view. The r coordinate is aligned along the body y-axis of the UAV. The UAV flies mainly straight above the objects. Thus the motion in r is very limited for the objects. Therefore it seems that the velocity in r varies too much and that the measurement noise in r velocity should be increased significantly. This is supported by the fact that the measurement error in s is much smaller for object 2, which has a constant velocity. The error does not exceed 100 pixels in 50 images, which means that the prediction error in s is below two pixels for each image.

The RMSE for the second object is displayed in Table 7.14. The values are calculated during the time span where the object is visible in the image. Therefore the values indicate the accuracy and noise level of the estimates as long as position measurements in the image plane are available. The RMSE in North and East position are very low and the accuracy is within one meter of the real position. The mean error in North is 0,52 meters which indicates a slight displacement. This is related to the mean error in the image plane. The RMSE in s is 5,98 and just above the mean error in s. This indicates that the object constantly is displaced 6 pixels in s (since the noise level is equal to the mean error). Furthermore it explains why the mean error in position is above zero. The mean error descends from the fact that the OF vectors are located on approximately the same part of the object. Thus the measurement noise is not white in reality. Nevertheless the error in position, both in the image plane and NE-plane, is very small and considered to be negligible with lock on the objects.

The RMSE value in r is 9,68 pixels. This is a significantly larger value than in

s. Therefore the position estimate in r is more insecure than the estimate in s and explains the rapid drift away from the true path in r when measurements are lost. The velocity in r seems to vary too much to have a reliable prediction of position in r. This would almost certainly be a smaller issue with a camera at rest. Nevertheless the performance is very reliable for position in the image plane and NE-plane as long as measurements are unavailable for just a few consecutive images. In fact, it is likely to believe that the performance would be great even if just one measurement is available in five images. The main issue appears when measurements are unavailable for more than ten consecutive images. An error of 50 pixels in the image plane is relatively small in NE since one pixel covers approximately 10 cm in the NE-plane.

Table 7.14: Mean error and $RMSE$ value for the second object during the period when measurements are available.

Error	**Mean Error $\bar{e}$**	**RMSE**
Position North	0,52	0,70
Position East	0,10	0,58
Velocity North	-0,22	2,02
Velocity East	-0,13	2,04
Position Image Plane r	-3,33	9,68
Position Image Plane s	-5,95	5,98

Discussion

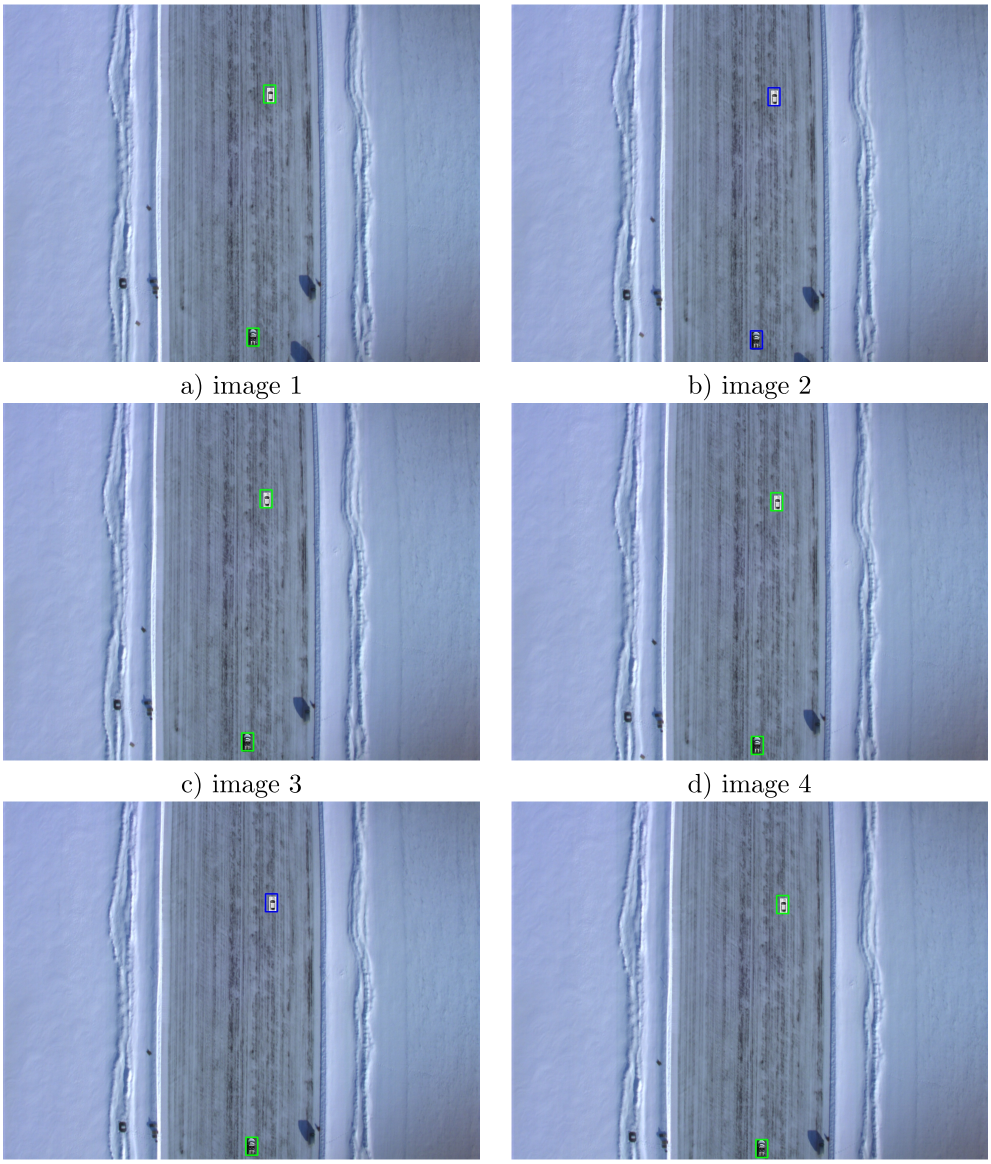
The observant reader might notice that the position estimate of object 1 drifts away more rapid in this case study than in Case Study 4 (even though the path is equal). This is kind of a surprise since the measurements are lost and reacquired at the same time. In both case studies the car is lost for the same amount of images and one would suspect the same error dynamics. However the problem can be explained by different values for the noise matrices in the Kalman filter. The process noise was increased by a factor of three in this case study and the measurements noise was increased by a factor of 5 in position and 2 in velocity (in the image plane). The increase in process noise is likely the main reason for the larger drift away from the true path. Therefore the process noise should maybe be decreased slightly, at least in position. The same noise level was assumed in the position and velocity measurements. If another case study had been carried out, the measurement noise in velocity would have been increased and the noise in position kept or decreased. However it would still be quite challenging with accurate prediction because small changes in roll, pitch or yaw has a large influence on the object position and velocity in the image plane.

The main issue with the estimates in the image plane is that the motion of the UAV varies too much to know the velocity in the image plane. Even small vibrations might displace the object several pixels in the image plane and appear as a sudden change in velocity. Thus the results indicate that frequent measurements of the position in the image plane are necessary, but that one measurement per second might be sufficient. This is based on the tracking error in the image plane which is limited for ten images without a measurement. Another case study in the future should investigate a situation where one measurement is available each second (one measurement in ten images) instead of up to ten each second.

The similar noise level in NED velocity for both objects is somewhat surprising since object 2 moves with a constant velocity in NED. Thus one might expect that the constant velocity model would be more accurate for the second object. However the constant velocity model refers to constant velocity in the image plane. Constant velocity in NED leads to constant velocity in the image plane, but only if the UAV moves with a constant velocity and more importantly has the same attitude. Therefore the issues discussed in the previous paragraph lead to a varying velocity in the image plane for both objects, and it is understandable that the accuracy are similar for both objects.

Overall the performance of the tracking system is satisfactory, even in the case of multiple objects in the image plane at the same time. The tracking system successfully managed to distinguish the objects. Furthermore the moving object detection algorithm worked equally well with another object in the image plane. An interesting point is that approximately ten OF vectors were calculated on the first object, but just two to three OF vectors on the second object. Therefore it seems that a single OF vector is enough for reliable measurements. The important factor is that at least one OF vector is located on the object frequently, and of course that the moving object segmentation algorithm locates that OF vector. The main challenges with the tracking system in its current state are that the objects needs to move with a significant velocity with respect to the background and that it might be hard to separate objects that are very similar. In Case Study 4 and 5 the objects moves with a velocity almost equal to the UAV in order to keep them in the image plane for a sufficient amount of images (time). The classification of objects are, as mentioned before, based on template matching. Therefore objects with approximately the same size and color are a problem. Tracking of a large amount of objects might be challenging for the tracking system in its current state. A solution could be to add another layer to the classification algorithm to increase the reliability. The second layer could for instance be based on a nearest neighbour approach to confirm the the results from the template matching (first layer).

An example of six consecutive images from the tracking system finishes this case study. They illustrate the performance of the tracking system. The rectangle are centered around the estimated position. A green color indicates that measurements of position and velocity were available in that image (the moving object detection algorithm located the object). A blue color indicates that measurements are unavailable and that the estimates are based on prediction. The images are displayed in Figure 7.24.



a) image 1 b) image 2

c) image 3 d) image 4

e) image 5 f) image 6

Figure 7.24: The figure shows six consecutive images. A green rectangle around the objects indicates that measurements are available for the objects in the image. A blue rectangle indicates that the estimates are based on prediction.

Part V

Closing Remarks

Chapter 8

Conclusion and Future Work

In the present thesis, several different topics have been investigated. The first part focused on computer vision and the camera system. The second part focused on the design of a camera-aided nonlinear observer for the estimation of attitude, position, velocity and gyro bias for a fixed-wing UAV. The latter part focused on detection and tracking of moving objects based on the camera system and the nonlinear observer. The last part of this thesis have been dedicated to the implementation of necessary SW and experiments carried out to evaluate how the navigation system and the tracking system work in practice. This chapter will conclude the work and summarize the most important findings. It contains the following topics:

- Section 8.1 gives a brief overview of the work carried out in this thesis.
- Section 8.2 summarizes and underlines the most important findings
- Section 8.3 describes future work that might be worth investigating based on the findings in this thesis.

8.1 Overview

The main objectives of this thesis have been to calculate the body-fixed velocity of a fixed-wing UAV with optical flow, evaluate a vision-aided navigation system for fixed-wing UAVs, develop a moving object detection algorithm and develop a tracking a system for moving objects. The navigation system is based on estimation of attitude, position, velocity and gyro bias with a nonlinear observer. The attitude part of the nonlinear observer is aided by calculation of the body-fixed linear velocity of the UAV with images captured by a video camera. The tracking system is based on an algorithm for detection of moving objects, a classifier used to describe each object and a Kalman filter for motion estimation. The moving object detection algorithm uses the attitude, velocity and position of the UAV to find moving objects. Therefore it uses the estimates from the nonlinear observer to find moving objects. This is done by calculating the theoretical optical flow and compare it with the measured optical flow. If the measured optical flow deviates significantly from the theoretical flow at some pixels, a moving object is assumed to be located at that position. The detected objects are tracked by creating a unique instance of the Kalman filter for each object. A template of the area with the detected object is used as a classifier and describes the object. The classifiers are used to distinguish different objects in order to associate new measurements with already tracked objects.

Several case studies have been conducted in order to evaluate the navigation and tracking system in practice. Therefore the navigation and tracking system have been implemented in SW in order to evaluate the performance through computer simulations. Furthermore several optical flow algorithms have been implemented. A UAV experiment have been carried out to gather images captured from a UAV and collect measurements from inertial sensors and GPS. In addition two papers have been written in cooperation with fellow MSc. student Jesper Hosen, PhD. student Lorenzo Fusini, Professor Thor I. Fossen and Professor Tor A. Johansen. The first paper has been accepted for ICUAS'15 and the second paper is submitted to AIAA SciTech 2016 with acceptance or rejection in august.

8.2 Findings

Case study 1 focused on the calculation of body-fixed velocity for the UAV through optical flow and was based solely on real data. The results showed that the calculation was exposed to noise when measurements of roll, pitch and altitude were used, but that estimates of the same states decreased the noise level significantly. The longitudinal velocity was calculated to be slightly greater than the longitudinal velocity from the autopilot and the lateral velocity oscillated more than excepted. However the case study showed that it is possible to calculate reasonable values for the body-fixed velocity through optical flow. More accurate calculation of the body-fixed velocity requires a flat horizontal terrain in the images and an altimeter to get a more precise altitude measurement. If the normalized velocity (direction) is more important than the up-to scale velocity, epipolar geometry can be used without the dependency on the Euler angles and the altitude. Epipolar geometry and the approach to calculate normalized body-fixed velocity is described in the second paper (Appendix B).

The second case study compared three different configurations for the nonlinear observer presented in Chapter 3 on real data gathered at a UAV experiment. The results showed that the attitude could be estimated accurately with the vision-aided navigation system. Normalized body-fixed velocity (from Case Study 1), calculated both with and without feedback of the estimated states, was used to aid the attitude estimator. The attitude was not estimated correctly with magnetometer aiding. This might be because of wrong calibration, but the case study illustrated some of the issues with a magnetometer in a UAV. The limited amount of space increases the difficulty in proper shielding of the magnetometer. Furthermore the magnetometer measurement had a direction not very far from the accelerometer measurement. It is harder to find three different angles (Euler angles) when the reference vectors are more similar. The normalized body-fixed velocity from the camera has a direction very different from the accelerometer and thus it is easier to find three separate angles. The case study showed that the nonlinear observer managed to estimate the states with comparable performance to the EKF in the autopilot of the UAV. In addition to evaluating the different configurations of the nonlinear observer, one of the main contributions from this work is the work related to the publication of the two papers attached in Appendix A and B.

The third case study evaluated the moving object detection algorithm proposed in Section 4.2.3. The algorithm is one of the contributions in this thesis and the case study showed that the algorithm, which is based on optical flow, could be used to detect moving objects. The simulated results illustrate that objects with a certain velocity can be detected with high reliability as long as the objects are not too homogeneous. The algorithm detects fast moving objects, such as cars, but slowly moving objects, such as walking humans, can not be detected because of the large UAV velocity. Slowly moving objects could probably be detected with a camera at rest, but the noise level is too high to detect slowly moving objects from an airborne fixed-wing UAV. The detection rate was shown to be quite reliable with very few false detections. The main challenge with the algorithm is its dependency on finding features on the objects of interest.

The fourth case study evaluated the proposed tracking system on a single moving object. The tracking system consists of the moving object detection algorithm, a classifier to distinguish different objects and a Kalman filter for each tracked object. The results illustrated that the tracking system worked satisfactory on a single object and that the estimation errors were negligible as long as measurements were available. The object was tracked in the image plane with high accuracy. Furthermore the estimates were transformed to NED and the estimated path in NED converged to the true path when measurements were available. The velocity in NED was calculated by first order differentiation of the estimated position in NED. The calculated velocity had a pretty large noise level, but the mean was close to the true value. For applications with larger requirements to the accuracy another way to estimate the velocity should be investigated.

The fifth case study evaluated the tracking system with multiple objects in the images. Two different moving objects were visible in the images. The tracking system was able to distinguish the two objects and track both objects accurately, as long as measurements of position were available. Furthermore the estimates were transformed to NED and compared with the true paths. The root mean squared error showed that accuracy was within one meter in NED as long as measurements were available. However one of the main challenges with the tracking system is to predict the motion of the objects correctly. This is very difficult since small UAV vibrations, UAV velocity variations and small changes in attitude for the UAV can displace the objects significantly in the image plane. This is especially challenging with a fixed-wing UAV moving with large velocity. Therefore the performance of the tracking system depends on measuring the position regularly, and preferably at least once each second. Even though the estimates are inaccurate if measurements are unavailable the estimates converges quickly if measurements become available again. The overall performance of the tracking system was assessed to be satisfactory.

8.3 Future Work

This section is going to discuss the most important factors for future work with respect to the findings in this report. The most exciting and promising improvements will receive extra attention.

The nonlinear observer should be investigated further, both theoretically and experimentally. A stability proof for the feedback of roll, pitch and altitude in the calculation of body-fixed velocity should be derived. This is because case study 1 and 2 indicated that feedback increases the accuracy in practice. The flat terrain assumption in [21] should be relaxed by epipolar geometry as suggested in the paper submitted to AIAA SciTech 2016. However for dead reckoning applications the body-fixed velocity up-to scale is of importance. Therefore it is also important to evaluate the velocity calculation presented in Section 2.4 further. If the assumed altitude is larger than the actual altitude, the body-fixed velocity is calculated to be larger than in reality. This is the case when flying over rugged terrain such as trees. Therefore the altitude measurements from an altimeter should replace the altitude measurement from the GPS in order to increase the accuracy. An experimental evaluation of the performance with altimeter should be conducted in order to check if the accuracy surpasses the accuracy in case study 1.

Another possibility for increasing the performance of the nonlinear observer is to use a combination of a magnetometer and video camera to aid the attitude estimator. This can be achieved by creating a filter that extracts the measurement from the most accurate sensor at each time step, or ideally combines the measurements to create a more accurate and reliable measurement. The misalignment of the inertial sensors might be an error source. It can be measured and accounted for in the equations to check if it increases the accuracy of the estimates. It would also be adequate to investigate if the inclinometer measurements can be utilized in the attitude estimator. In addition it would be interesting to investigate if it is possible to find some directives in how the observer gains can be chosen. For the moment they are mostly based on trial and error. Therefore it would be beneficial to identify a way to tune the observer more efficiently.

A future flight experiment should be carried out and include more difficult maneuvers. Turns with smaller radius and an increased amount of pitching motion can be used to evaluate the performance of the nonlinear observer in more challenging conditions. A flight experiment at lower altitudes would also be interesting since features in the images are larger at smaller altitude. The accuracy of the calculated body-fixed velocity can be different, and most likely better at lower altitude. This is because the calculated body-fixed velocity is proportional to the altitude and the optical flow measurement. Thus an error in measured optical flow will be multiplied by the altitude. Therefore the same noise level (as in Case Study 1) in measured optical flow leads to more accurate body-fixed velocity measurements at smaller altitudes.

A factor that can increase the reliability of the gathered data is to develop a more accurate way to synchronize the images from the camera with the rest of the sensors. The current method is described in Section 5.2 and depends on performing the synchronization manually. It would be beneficial to store the images with a time stamp directly related to the rest of the sensors instead of conducting a manual synchronization after the UAV experiment.

Future work should improve and evaluate the moving object detection algorithm on real data. Most importantly, images with moving objects should be gathered at a flight experiment. This can be achieved by driving a car below the UAV for example. The car can be equipped with a GPS receiver in order to collect velocity and position measurements of the car, and use those measurements to test the tracking system on real data. The captured images can be used to validate if the moving object detection algorithm is able to locate the car. Furthermore if the algorithm can be tested on real data the weaknesses of the algorithm will be more prominent and the algorithm can be developed further. More advanced statistical methods would probably be more reliable and should be investigated. A viable option would be to use the RANSAC algorithm when looking for moving objects. RANSAC can be used to identify optical flow vectors that do not coincide with the majority of the optical flow vectors.

The tracking system should also be evaluated experimentally in the future. In addition more case studies based on computer simulation can be carried out to investigate the tracking system further. It would be particularly interesting to check if another motion model in the Kalman filter can increase the accuracy of the predicted paths. Furthermore different noise models should be tested in order to increase the performance. A possibility is to change the process noise when measurements are unavailable. Another interesting topic would be to investigate if it is possible to account for changes in the attitude of the UAV. If this is possible, the prediction step in the Kalman filter can be conducted more accurately. The main issue today is the fact that the velocity in the image plane varies rapidly because even small changes in attitude (between images) can displace the object several pixels in the image plane. Therefore a way to limit the dynamics of the predicted velocity in the image plane should be investigated.

Part VI

Appendices

Appendix A

Paper Accepted for ICUAS'15

This appendix includes the paper submitted and accepted for the International Conference on Unmanned Aerial Vehicles 2015 (ICUAS 15') in Denver in June.

Experimental Validation of a Uniformly Semi-globally Exponentially Stable Non-linear Observer for GNSS- and Camera-aided Inertial Navigation for Fixed-wing UAVs

Lorenzo Fusini, Jesper Hosen, Håkon H. Helgesen, Tor A. Johansen and Thor I. Fossen
Norwegian University of Science and Technology
Centre for Autonomous Operations and Systems
Department of Engineering Cybernetics
Trondheim, Norway
E-mail: {lorenzo.fusini, tor.arne.johansen}@itk.ntnu.no, thor.fossen@ntnu.no, {jesperho, hakonhhe}@stud.ntnu.no

***Abstract*—This paper provides experimental validation of a uniformly semi-globally exponentially stable (USGES) non-linear observer for estimation of attitude, gyro bias, position, velocity and acceleration of a fixed-wing Unmanned Aerial Vehicle (UAV). The available sensors are an Inertial Measurement Unit (IMU), a Global Positioning System (GPS) receiver, a video camera, and an inclinometer. The UAV is flown with the sensor payload and all data is stored locally on a hard drive, which is recovered at the end of the flight. The non-linear observer is then tested offline with the recorded sensor data. An optical flow algorithm is used to calculate the UAV velocity based on the camera images, which is used as a reference vector of the body-fixed velocity in the attitude observer. The results are compared with an Extended Kalman Filter (EKF) and illustrate that the estimates of the unmeasured states converge accurately to the correct values, and that the estimates of the measured states have less noise than the measurements.**

I. Introduction

The estimation of position, velocity, and attitude of a vehicle at any time is commonly referred to as "navigation". The most used tool for this purpose has been the EKF, but in the last decades researchers have started to investigate new solutions, alternative to the Kalman filter, to the navigation problem, namely by developing non-linear observers with complete stability proofs and experimental validation. Non-linear observers have the advantage, over the EKF, of featuring a smaller computational footprint and often being globally exponentially stable (GES), a result that renders the observers robust to disturbances and initialization uncertainties. The problem of attitude estimation has received significant attention as a stand-alone problem [1]–[11]. In addition to this, other researchers have integrated Inertial Navigation System (INS), magnetometer/compass and GNSS to estimate the navigation states of a vehicle.

In [12] the authors expanded the vector-based observer proposed by [6] and [7] to include GNSS velocity measurements. [1] and [2] built globally exponentially stable (GES) attitude estimators based on multiple time-varying reference vectors or a single persistently exiting vector. A similar observer was developed in [13] and [14] to include also gyro bias and GNSS integration. An extension of this [15] replaced the rotation matrix with the unit quaternion for representing attitude, considered Earth rotation and curvature, a non-constant gravity vector, and included accelerometer bias estimation.

Another sensor commonly used in navigation is the camera. Low weight, low power consumption, and a wide range of machine vision software make it a viable choice for navigation purposes. Some drawbacks are its dependence on lighting and weather conditions, which directly affect the availability of features in the scene, and the difficulty in separating camera motion from moving objects in complex non-stationary environments.

Optical flow (OF) is how features in an image plane move between two consecutive images, caused by relative motion between the camera and the object being depicted. In the simplest case it could be understood as the pixel displacement of a single feature between two successive images. The OF can be represented as multiple vectors describing the change in the image plane in time. Several methods exists for determining the OF of a series of images, e.g. [16]–[19].

Machine vision and OF have been used for different applications in UAV navigation including indoor manoeuvring [20], [21], linear and angular velocity estimation [22]–[24], and obstacle avoidance [20], [25]–[29] as well as height above the ground estimation in [30]. [31], [32] use OF in assisting a landing of a UAV independent of external sensor inputs. OF from a single camera is used in [33], [34] to estimate body axes angular rates of an aircraft as well as wind-axes angles. [24], [35], [36] have used OF as input in Kalman filter-based navigation systems, fusing OF measurements with acceleration and angular velocity measurements. [37], [38] have used camera as sensor for navigating in GPS-denied environments.

A comparison of the performance of different methods of estimating the attitude of UAV based on machine vision is presented in [39], and different OF algorithms are evaluated in [23], [40] by estimating UAV velocity.

In [41] OF vectors are used to calculate the normalized body-fixed velocity of the UAV, and fed into the non-linear observer as a reference vector.

A. Contribution of this Paper

This paper provides experimental tests of a USGES non-linear observer for estimation of attitude, gyro bias, position, velocity, and acceleration of a fixed-wing UAV [41]. Exponential stability guarantees strong convergence and robustness properties, hence it is an important property to have in systems that are exposed to disturbances and uncertain initialization. The camera can sometimes replace the magnetometers: in small UAVs the magnetometers are heavily affected by disturbances and noise generated by the engine, while the camera is not conditioned by this.

The sensor data are logged during the UAV flight and used offline on a PC to test the observer. An OF algorithm is used to calculate the body-fixed velocity of the UAV based on the camera images, altitude and inclinometer data. To demonstrate their validity, the estimated states are compared with those evaluated via the EKF.

II. Notation and Preliminaries

Vectors and matrices are represented by lowercase and uppercase letters respectively. X^{-1}, X^{+}, and $\mathrm{tr}(X)$ denote the inverse, pseudoinverse, and trace of a matrix respectively, X^T the transpose of a matrix or vector, $\hat{X}$ the estimated value of X, and $\tilde{X} = X - \hat{X}$ the estimation error. $\|\cdot\|$ denotes the Euclidean norm, I_n the identity matrix of order n, and $0_{m\times n}$ the $m \times n$ matrix of zeros. A vector $x = [x_1, x_2, x_3]^T$ is represented in homogeneous coordinates as $\underline{x} = [x_1, x_2, x_3, 1]^T$. The function $\mathrm{sat}(\cdot)$ performs a component-wise saturation of its vector or matrix argument to the interval $[-1, 1]$. The operator $S(x)$ transforms the vector x into the skew-symmetric matrix

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

The inverse operation is denoted as $\mathrm{vex}(\cdot)$, such that $\mathrm{vex}(S(x)) = x$. For a square matrix A, its skew-symmetric part is represented by $\mathbb{P}_a(A) = \frac{1}{2}(A - A^T)$.

Two reference frames are considered in the paper: the body-fixed frame {B} and the North-East-Down (NED) frame {N} (Earth-fixed, considered inertial). The rotation from frame {B} to {N} is represented by the matrix $R_b^n \equiv R \in SO(3)$, with $SO(3)$ representing the Special Orthogonal group.

A vector decomposed in {B} and {N} has superscript ^{b} and ^{n} respectively. The camera location w.r.t. {N} is described by $c^n = [c_x^n, c_y^n, c_z^n]^T$. A point in the environment expressed w.r.t. {N} is $t^n = [x^n, y^n, z^n]^T$: note that a point located on the ground corresponds to $z^n = 0$ and such it will be throughout the paper. The same point expressed w.r.t. {B} is $t^b = [x^b, y^b, z^b]^T$. It will also be assumed that every point is fixed w.r.t. {N}. The greek letters ϕ, θ, and ψ represent the roll, pitch, and yaw angles respectively, defined according to the zyx convention for principal rotations [42], and they are collected in the vector $\Theta = [\phi, \theta, \psi]^T$. A 2-D camera image has coordinates $[r, s]^T$, aligned with the y^b- and x^b-axis respectively (see Fig. 2). The derivative $[\dot{r}, \dot{s}]^T$ of the image coordinates is the OF. The subscript $_F$ indicates a quantity evaluated by means of the OF.

A. Measurements and Sensors

The sensor payload consists of an IMU, a GPS receiver, a video camera, and an inclinometer, providing the following information:

- *GPS*: NED position p^n and, by differentiation, NED velocity v^n;
- *IMU*: biased angular velocity $\omega_m^b = \omega^b + b^b$, where b^b represent the gyro bias, and acceleration a^b;
- *camera*: 2-D projections $[r, s]^T$ onto the image plane of points $[x^n, y^n, z^n]^T$ from the 3-D world;
- *inclinometer*: roll ϕ and pitch θ angles.

Further information on the actual sensors employed in the experiment is presented in Section V.

III. Optical Flow

The observer presented in Section IV depends on velocity measurements from the on-board camera decomposed in the body-fixed frame. These measurements are generated from OF, therefore it is necessary to compute the OF vectors from consecutive images before these vectors are transformed to velocity measurements. The OF calculation and the transformation is presented in the forthcoming section.

A. Optical flow computation

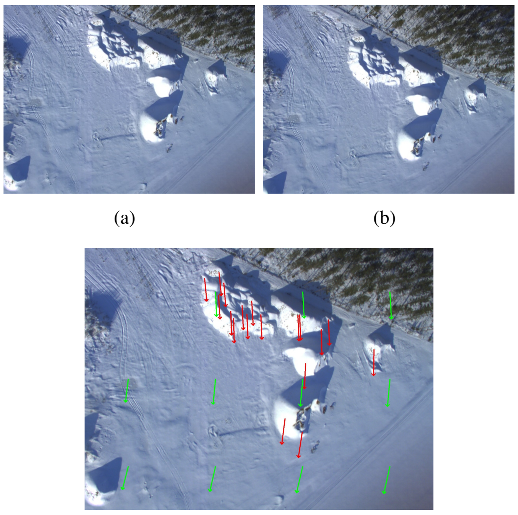
There exist several methods for computing OF. For the experiment presented in Section V two specific methods are combined. The first one is SIFT [18] which provided the overall best performance in [23]. SIFT uses a feature-based approach to compute OF. The total number of OF vectors in each image depends on the number of features detected and matched together. Since the transformation in Section III-B requires at least three OF vectors [41], it is necessary to make sure that this is handled. It is not possible to guarantee three OF vectors with SIFT since homogeneous environments, like snow or the ocean, increase the difficulty of finding distinct features. Therefore the OF vectors created by SIFT are combined with a second method, which is based on region matching [43].

The region matching method used here is a template matching approach based on normalized cross-correlation [44]. The displacements of twelve templates, created symmetrically across the images, are used to find twelve OF vectors. Template matches below a given threshold are discarded and the corresponding OF vector removed. Unreliable matches can occur in case of uniform terrain, changes in brightness or simply when the area covered by the template has disappeared from the image in the time between the capture of images. An example of OF vectors computed with SIFT and template matching is displayed in Fig. 4.

In case of mismatches, both methods will create erroneous OF vectors. It is desired to locate and remove these vectors. Therefore an outlier detector is implemented before the vectors are used to calculate body-fixed velocities. The outlier detector utilizes a histogram to find the vectors that deviate from the mean with respect to direction and magnitude.

B. Transformation from optical flow to velocity

For the OF computations to be useful in the observer, a transformation to body-referenced veloc-



(a) (b)

(c)

Fig. 1. a) Image captured at time t_0. b) Image captured at time $t_0+\Delta t$. c) Optical flow vectors between image a) and b), generated by SIFT (red) and Template Matching (green).

ity is necessary. The transformation is motivated by [41] and the pinhole camera model is used [45]. The camera-fixed coordinate system is related to the body-fixed coordinate system through Fig. 2, where the downward-looking camera is aligned with the body z-axis. The focal point of the camera is for simplicity assumed to coincide with the origin of {B}.

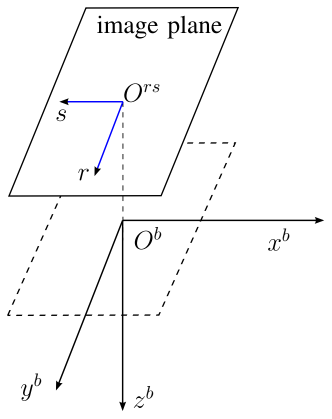


Fig. 2. Pinhole camera model. The camera is oriented downwards, while x^b is the direction of flight.

It is necessary to relate a point in the terrain expressed in {N} t^n to {B}, since points in the terrain are used to compute the body-referenced velocity. The matrix R^n_b and vector c^n represent a rotation and a translation between {N} and {B}. They can be merged to form a homogeneous 4×4-transformation T^n_b

$$T^n_b(\Theta) = \begin{bmatrix} R^n_b(\Theta) & c^n \\ 0_{1\times 3} & 1 \end{bmatrix}$$

such that $\underline{t}^n = T^n_b(\Theta)\underline{t}^b$ where $\underline{t}^n$ and $\underline{t}^b$ is the same homogeneous point represented in {N} and {B} respectively. The transformation can be inverted to find t^b from t^n

$$\underline{t}^b = T^n_b(\Theta)^{-1}\underline{t}^n = \begin{bmatrix} R^n_b(\Theta)^T & -R^n_b(\Theta)^T c^n \\ 0_{1\times 3} & 1 \end{bmatrix} \underline{t}^n \tag{1}$$

and t^b is now a function of x^n, y^n, z^n, c^n_x, c^n_y, c^n_z, ϕ, θ, ψ.

The relationship between t^b and its projection onto the image plane is expressed by the well-known pinhole camera model [45] [46].

$$\begin{bmatrix} r \\ s \end{bmatrix} = \frac{f}{z^b}\begin{bmatrix} y^b \\ -x^b \end{bmatrix}, z^b \neq 0 \tag{2}$$

where f is the focal length of the camera. As t^b in itself is not available, the relationship in (1) is used to express t^b in (2) as

$$\begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} = \begin{bmatrix} \frac{s c^n_z}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{r c^n_z}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \\ -\frac{f c^n_z}{s\sin(\theta)+\cos(\theta)(f\cos(\phi)+r\sin(\phi))} \end{bmatrix} \tag{3}$$

All features tracked by the camera are assumed to be stationary with respect to {N}. Hence the UAV linear and angular velocity relative to a feature tracked by the OF algorithm, v^b_F and ω^b_F, will be equal for every tracked feature. Furthermore it is assumed that the terrain is flat, such that every feature is located at the same altitude. For every feature j, the relationship between OF and body-fixed linear/angular velocity is given as

$$\begin{bmatrix} \dot{r}_j \\ \dot{s}_j \end{bmatrix} = -M_j(f, r_j, s_j, \phi, \theta, c^n_z)\begin{bmatrix} v^b_F \\ \omega^b_F \end{bmatrix}$$
$$M_j = \frac{f}{z^b_j}\begin{bmatrix} 0 & 1 & -\frac{y^b_j}{z^b_j} & -\frac{{y^b_j}^2}{z^b_j} - z^b_j & \frac{y^b_j x^b_j}{z^b_j} & x^b_j \\ -1 & 0 & \frac{x^b_j}{z^b_j} & \frac{x^b_j y^b_j}{z^b_j} & -\frac{{x^b_j}^2}{z^b_j} - z^b_j & y^b_j \end{bmatrix} \tag{4}$$

where $M_j \in \mathcal{R}^{2\times6}$ in (4) is motivated by [41]. If the number of features being tracked is k, then the OF vector has dimension $2k$. A matrix $M \in \mathcal{R}^{2k\times6}$ might be created by concatenating the matrices $M_j, j = 1\dots k$, and the following relationship is obtained

$$\begin{bmatrix} \dot{r_1} \\ \dot{s_1} \\ \vdots \\ \dot{r_k} \\ \dot{s_k} \end{bmatrix} = -M \begin{bmatrix} v_F^b \\ \omega_F^b \end{bmatrix}, M = \begin{bmatrix} M_1 \\ \vdots \\ M_k \end{bmatrix} \tag{5}$$

By calculating the pseudoinverse of M in (5) the angular and linear velocity can be computed as

$$\begin{bmatrix} v_F^b \\ \omega_F^b \end{bmatrix} = -M^+ \begin{bmatrix} \dot{r_1} \\ \dot{s_1} \\ \vdots \\ \dot{r_k} \\ \dot{s_k} \end{bmatrix} \tag{6}$$

M^+ exists only if M^TM has full rank. This can only happen if the number of flow vectors are greater or equal to three. This is always the case in the experiment.

IV. Observer Design

A. Dynamic System

The dynamics of attitude, position, and velocity is described by

$$\dot{R} = RS(\omega^b) \tag{7a}$$
$$\dot{p^n} = v^n \tag{7b}$$
$$\dot{v}^n = a^n + g^n \tag{7c}$$

The objective is to estimate the attitude R, the position p^n, and the velocity v^n with exponential convergence rate. In addition to this, an estimator for the gyro bias b^b is also provided.

B. Assumptions

The observer design is based on the following assumptions:

Assumption 1: the OF algorithm uses a sufficient number of image features, such that M has full rank and Eq. (6) can be used.

Assumption 2: the gyro bias b^b is constant.

Assumption 3: there exists a constant $c_{obs} > 0$ such that, $\forall t \geq 0,\ \|v_F^b \times a^b\| \geq c_{obs}$.

Assumption 3 is a condition of non-collinearity for the vectors v_F^b and a^b, i.e. the angle between them is non-zero and none of them can be identically zero (see, e.g., [12], [6]). For a fixed-wing UAV this means that the observer cannot work while the vehicle stands still on the ground, but presents no issues during flight.

C. Observer Equations

The full observer was introduced in [41] as

$$\Sigma_1 \begin{cases} \dot{\hat{R}} = \hat{R}S(\omega_m^b - \hat{b}^b) + \sigma K_P \hat{J} \\ \dot{\hat{b}}^b = \text{Proj}(\hat{b}^b, -k_I \text{vex}(\mathbb{P}_a(\hat{R}_s^T K_P \hat{J}))) \end{cases} \tag{8}$$

$$\Sigma_2 \begin{cases} \dot{\hat{p}}^n = \hat{v}^n + K_{pp}(p^n - \hat{p}^n) + K_{pv}(v^n - \hat{v}^n) \\ \dot{\hat{v}}^n = \hat{a}^n + g^n + K_{vp}(p^n - \hat{p}^n) + K_{vv}(v^n - \hat{v}^n) \\ \dot{\xi} = -\sigma K_P \hat{J} a^b + K_{\xi p}(p^n - \hat{p}^n) + K_{\xi v}(v^n - \hat{v}^n) \\ \hat{a}^n = \hat{R}a^b + \xi \end{cases} \tag{9}$$

$$\text{OF} \left\{ \begin{bmatrix} \hat{v}_F^b \\ \hat{\omega}_F^b \end{bmatrix} = -\hat{M}^+ \begin{bmatrix} \dot{r} \\ \dot{s} \end{bmatrix} \right. \tag{10}$$

The subsystem Σ_1 represents the attitude observer, whereas Σ_2 represents the translational motion observer. In addition, (10) is given by machine vision. $\sigma \geq 1$ is a scaling factor tuned to achieve stability, k_I is a positive scalar gain, Proj$(\cdot,\cdot)$ represents a parameter projection [47] that ensures that $\|\hat{b}^b\|$ does not exceed a design constant $L_{\hat{b}^b} > L_{b^b}$ (see Appendix), and $\hat{R}_s = \text{sat}(\hat{R})$. The matrix $\hat{J}$ is the output injection term, whose design is inspired by the TRIAD algorithm [48] and defined

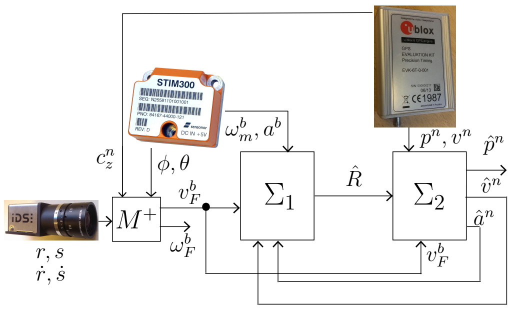


Fig. 3. Block diagram of the observer. Σ_1 represents the attitude observer, and Σ_2 the translational motion observer.

as

$$\hat{J}(v_F^b,\ \hat{v}^n, a^b, \hat{a}^n, \hat{R}) := \hat{A}_n A_b^T - \hat{R} A_b A_b^T \tag{11a}$$

$$A_b := [v_F^b,\ v_F^b \times a^b,\ v_F^b \times (v_F^b \times a^b)] \tag{11b}$$

$$\hat{A}_n := [\hat{v}^n,\ \hat{v}^n \times \hat{a}^n,\ \hat{v}^n \times (\hat{v}^n \times \hat{a}^n)] \tag{11c}$$

The subsystem Σ_2 represents the translational motion observer, where $K_{pp}, K_{pv}, K_{vp}, K_{vv}, K_{\xi p}$, and $K_{\xi v}$ are observers gains yet to be defined, and $g^n = [0, 0, 9.81]^T$ is the gravity vector in NED.

The system Σ_1–Σ_2 is a feedback interconnection, as illustrated by Fig. 3.

D. Stability Proof

The error dynamics of the non-linear observer can be written in a compact form as

$$\Sigma_1 \begin{cases} \dot{\tilde{R}} = RS(\omega^b) - \hat{R}S(\omega_m^b - \hat{b}^b) - \sigma K_P \hat{J} \\ \dot{\tilde{b}}^b = -\text{Proj}(\hat{b}^b, \tau(\hat{J})) \end{cases} \tag{12a}$$

$$\Sigma_2 \left\{ \dot{\tilde{w}} = (A - KC)\tilde{w} + B\tilde{d} \right. \tag{12b}$$

where $\tilde{w} = [(\tilde{p}^n)^T, (\tilde{v}^n)^T, (\tilde{a}^n)^T]^T$ collects the estimated position, velocity and acceleration vectors, $\tilde{d} = (RS(\omega^b) - \hat{R}S(\omega_m^b - \hat{b}^b))a^b + (R - \hat{R})\dot{a}^b$, and the four matrices in (12b) are defined as

$$A = \begin{bmatrix} 0_{6\times3} & I_6 \\ 0_{3\times3} & 0_{3\times6} \end{bmatrix}, \qquad B = \begin{bmatrix} 0_{6\times3} \\ I_3 \end{bmatrix},$$

$$C = \begin{bmatrix} I_6 & 0_{6\times3} \end{bmatrix}, \qquad K = \begin{bmatrix} K_{pp} & K_{pv} \\ K_{vp} & K_{vv} \\ K_{\xi p} & K_{\xi v} \end{bmatrix}.$$

Theorem 1 provides conditions that ensure USGES of the origin of the error dynamics (12).

Theorem 1: Let σ be chosen to ensure stability according to Lemma 1 in [13] and define $H_K(s) = (Is - A + KC)^{-1}B$. There exists a set $(0, c)$ such that, if K is chosen such that $A - KC$ is Hurwitz and $\|H_K(s)\|_\infty < \gamma$, for $\gamma \in (0, c)$, then the origin of the error dynamics (12) is USGES as defined by [49] when the initial conditions satisfy $\|\hat{b}^b(0)\| \leq L_{\hat{b}^b}$.

Proof: For the proof, see [41]. ■

V. Experimental Results

This section describes the experiment carried out to gather the necessary data and the results obtained with the non-linear observer and OF.

A. Setup

The UAV employed is a UAV Factory Penguin-B, equipped with a custom-made payload that includes all the necessary sensors. The IMU is a Sensonor STIM300, a low-weight, tactical grade, high-performance sensor that includes gyroscopes, accelerometers, and inclinometers, all recorded at a frequency of 300 Hz. The chosen GPS receiver is a uBlox LEA-6T, which gives measurements at 5 Hz. The video camera is an IDS GigE uEye 5250CP provided with an 8mm lens. The camera is configured for a hardware-triggered capture at 10 Hz: the uBlox sends a digital pulse-per-second signal whose rising edge is accurately synchronized with the time of validity of the recorded GPS position, which guarantees that the image capture is synchronized with the position measurements. The experiment has been carried out on 6 February 2015 at the Eggemoen Aviation and Technology Park, Norway, in a sunny day with good visibility, very little wind, an air temperature of about -8°C. The terrain is covered with snow and flat enough to let all features be considered as lying at zero altitude.

The observer is implemented using forward Euler discretization with a time-varying step depending on the interval of data acquisition of the fastest sensor, namely the STIM300, and it is typically around 0.003 seconds. The various parameters and gains are chosen as $L_{b^b} = 2°/\text{s}$, $L_{\hat{b}^b} = 2.1°/\text{s}$, $\sigma = 1$, $K_P = \text{diag}[0.1, 0.1, 0.5]$, $k_I = 0.006$, $K_{pp} = 30I_3$, $K_{pv} = 2I_3$, $K_{vp} = I_3$, $K_{vv} = 100I_3$, $K_{\xi p} = I_3$, and $K_{\xi v} = 50I_3$. All the gains are obtained by running the observer several times and correcting the gains until a satisfactory performance was achieved. Concerning the gains of the translational motion observer, it is also possible to tune them with the help of a linear matrix inequality formulation that allows $\|H_K(s)\|_\infty$ to satisfy the conditions of Theorem 1 (see [13], [50] for details).

The reference provided for the attitude, position, and velocity is the output of the EKF of the autopilot mounted on the Penguin-B. An exact reference for the gyro bias is not available, but an approximation of the real value is calculated by averaging the gyro measurements at standstill before and after the flight. The accelerometer bias is not estimated, but it is computed the same way as the gyro bias and subtracted from the accelerometers measurements before being used in the observer.

All the images are processed with a resolution of 1600×1200 (width×height) pixels and in their original state, without any filtering. The lens distortion of the camera is not accounted for, and no correction is applied to the images. SIFT is implemented with the open source computer vision library (OpenCV) [51] with default settings. Each match is tagged with a value indicating the accuracy of the match, and the smallest of these values is considered to be the best match. To increase the reliability of the OF vectors, each match is compared to the best one. Every match with an uncertainty more than double the uncertainty of the best match is removed. Also the template matching algorithm is implemented with OpenCV. The size of the templates is chosen to be 120×90 pixels and a correlation of 99% is required in order for a template match to be considered reliable and not removed.

B. Results

The results here presented refer to a complete flight of the Penguin-B, from take-off to landing,

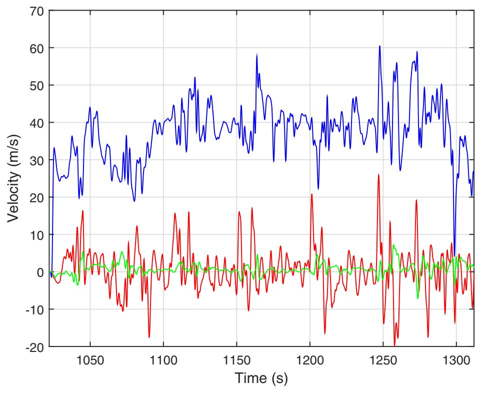


Fig. 4. Body-fixed velocity in the x, y, and z axis (blue, red, and green, respectively) calculated via machine vision.

which correspond to a travelled distance of approximately 9 km in around 5 min. The time on the x-axes is the elapsed time since the data logging begins, and only the significant part involving the flight is represented. The manoeuvres performed include flights on a straight line and turns with a large and small radius of curvature, namely approximately 200 m and 100 m, as it can be noticed from Fig. 6. The body velocity calculated via the OF is represented in Fig. 4 and is the result of (6). The estimated attitude, position, and velocity are illustrated in Fig. (5)–(7): it is clear that the observer (blue, solid line) performs well when compared to the EKF (red, dashed line). The pitch estimate presents some deviation, probably due to a misalignment of the sensor on the payload. The estimated North and East velocities show some small peaks at around 1130–1150 s and 1240 s, which are due to the presence of outliers in the GPS data. The estimated gyro bias is presented in Fig. 8: the estimates do not converge as well as the other states, but they remain within $0.2°$ of their initial estimate.

VI. Conclusions

In this paper a USGES non-linear observer has been tested on experimental data obtained by flying a fixed-wing UAV with a custom-made payload of sensors. An OF algorithm has been employed to calculate the body-referenced velocity of the vehicle by means of IMU, camera

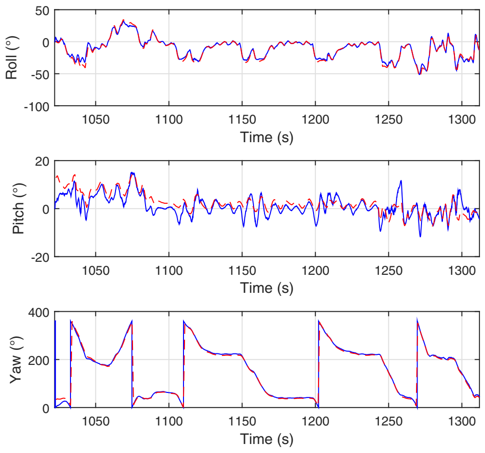


Fig. 5. Estimated (blue, solid) and EKF (red, dashed) attitude.

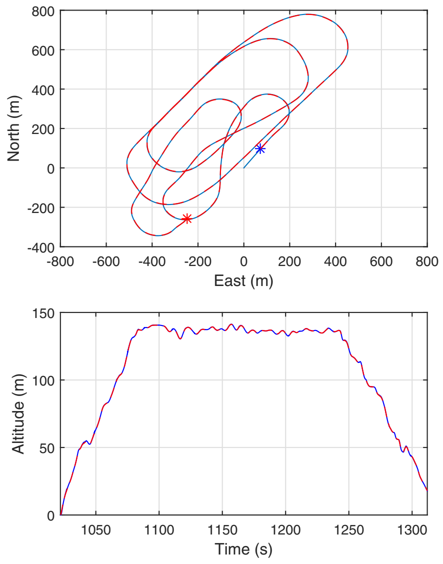


Fig. 6. Estimated (blue, solid) and EKF (red, dashed) position on the N-E plane and altitude. The blue and red star indicate the start and end of the data set, respectively.

images and GPS measurements. Such velocity, accelerometers measurements, estimated NED velocity, and estimated NED acceleration have been used as reference vectors in the injection term of the observer in order to provide the desired estimates. The results presented for different types

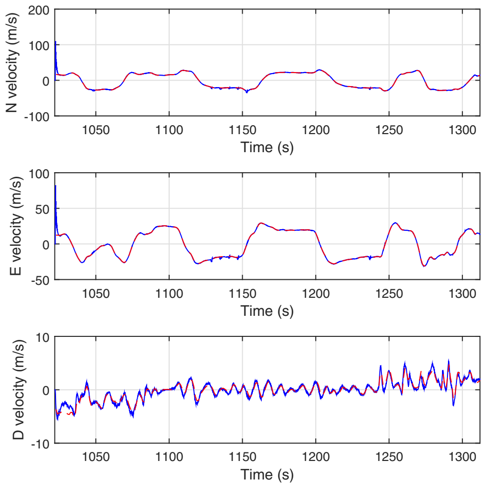


Fig. 7. Estimated (blue, solid) and EKF (red, dashed) NED velocities.

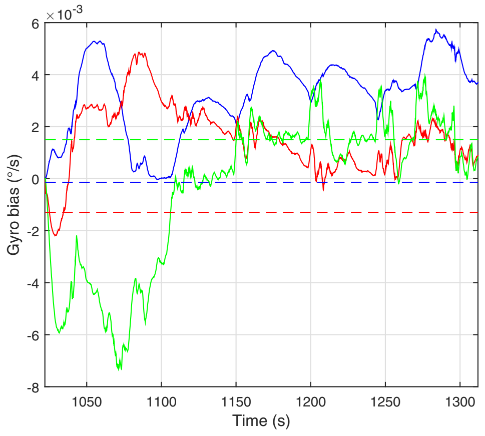


Fig. 8. Gyro bias calculated at standstill (dashed) and estimated by the observer (solid). The x, y, and z components are blue, red, and green, respectively.

of manoeuvres confirm the validity of the analysis and design.

Acknowledgements

This work has been carried out at the Centre for Autonomous Marine Operations and Systems (AMOS), supported by the Research Council of Norway through the Centres of Excellence funding scheme, grant no. 223254, and the FRINATEK grant 221666. The Norwegian Research Council is acknowledged as the main sponsor of AMOS. The authors are grateful for the assistance provided by the UAV engineers at NTNU and Maritime Robotics AS, in particular Lars Semb and Carl Erik Stephansen. Significant contributions to the construction of the UAV payload was made by the rest of the navigation team at NTNU, in particular Sigurd M. Albrektsen, Jakob M. Hansen and Kasper T. Borup.

REFERENCES

[1] P. Batista, C. Silvestre, and P. Oliveira, "GES attitude observers - Part I: Multiple general vector observations," *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 2985–2990, 2011.

[2] ——, "GES attitude observers - Part II: Single vector observations," *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 2991–2996, 2011.

[3] H. Grip, T. Fossen, T. Johansen, and A. Saberi, "Attitude estimation using biased gyro and vector measurements with time-varying reference vectors," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1332–1338, 2012.

[4] J. Guerrero-Castellanos, H. Madrigal-Sastre, S. Durand, L. Torres, and G. Muñoz-Hernández, "A robust nonlinear observer for real-time attitude estimation using low-cost MEMS inertial sensors," *Sensors*, vol. 13, no. 11, pp. 15 138–15 158, 2013.

[5] M. Hua, G. Ducard, T. Hamel, R. Mahony, and K. Rudin, "Implementation of a nonlinear attitude estimator for aerial robotic vehicles," *IEEE Transactions on Control System Technology*, vol. 22, no. 1, pp. 201–213, 2014.

[6] R. Mahony, T. Hamel, and J. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[7] R. Mahony, T. Hamel, J. Trumpf, and C. Lageman, "Nonlinear attitude observers on SO(3) for complementary and compatible measurements: A theoretical study," *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 6407–6412, 2009.

[8] R. Mahony, M. Euston, J. Kim, P. Coote1, and T. Hamel, "A non-linear observer for attitude estimation of a fixed-wing unmanned aerial vehicle without GPS measurements," *Transactions of the Institute of Measurement and Control*, vol. 33, no. 6, pp. 699–717, 2011.

[9] J. Trumpf, R. Mahony, T. Hamel, and C. Lageman, "Analysis of non-linear attitude observers for time-varying reference measurements," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2789–2800, 2012.

[10] S. Salcudean, "A globally convergent angular velocity observer for rigid body motion," *IEEE Transactions on Automatic Control*, vol. 36, no. 12, pp. 1493–1497, 1991.

[11] J. Thienel and R. Sanner, "A coupled nonlinear spacecraft attitude controller and observer with an unknown constant gyro bias and gyro noise," *IEEE Transactions on Automatic Control*, vol. 48, no. 11, pp. 2011–2015, 2003.

[12] M. Hua, "Attitude estimation for accelerated vehicles using GPS/INS measurements," *Control Engineering Practice*, vol. 18, no. 7, pp. 723–732, 2010.

[13] H. Grip, T. Fossen, T. Johansen, and A. Saberi, "A nonlinear observer for integration of GNSS and IMU measurements with gyro bias estimation," *American Control Conference (ACC)*, pp. 4607–4612, 2012.

[14] ——, "Globally exponentially stable attitude and gyro bias estimation with application to GNSS/INS integration," *Automatica*, vol. 51, pp. 158–166, 2015.

[15] ——, "Nonlinear observer for GNSS-aided inertial navigation with quaternion-based attitude estimation," *American Control Conference (ACC)*, pp. 272–279, 2013.

[16] B.Lucas and T.Kanade, "An iterative image restoration technique with an application to stereo vision," *Proc. DARPA Image Underst, Workshop*, pp. 121–130, 1981.

[17] B.K.P.Horn and B.G.Schunk, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–204, 1981.

[18] D. Lowe, "Object recognition from local scale-invariant features," *Proc. Int. Conf. Computer Vision*, pp. 1150–1157, 1999.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[20] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav navigation through indoor corridors using optical flow," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3361–3368, 2010.

[21] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 20–25, 2011.

[22] D. Dusha, W. Boles, and R. Walker, "Attitude estimation for a fixed-wing aircraft using horizon detection and optical flow," *Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*, pp. 485–492, 2007.

[23] M. Mammarella, G. Campa, M. Fravolini, and M. Napolitano, "Comparing optical flow algorithms using 6-dof motion of real-world rigid objects," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1752 – 1762, 2012.

[24] S. Weiss, R. Brockers, and L. Matthies, "4DoF drift free navigation using inertial cues and optical flow," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4180–4186, 2013.

[25] J. C. Zufferey and D. Floreano, "Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, pp. 2594–2599, 2005.

[26] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a UAV," *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 302–309, 2005.

[27] P. C. Merrell, D.-J. Lee, and R. W. Beard, "Obstacle avoidance for unmanned air vehicles using optical flow probability distributions," *Society of Photo-Optical Instrumentation*

Engineers (SPIE) Conference Series, vol. 5609, no. 1, pp. 13–22, 2004.

[28] J. Conroy, G. Gremillion, B. Ranganathan, and J. Humbert, “Implementation of wide-field integration of optic flow for autonomous quadrotor navigation,” *Autonomous Robots*, vol. 27, no. 3, pp. 189–198, 2009.

[29] F. Ruffier and N. Franceschini, “Visually guided micro-aerial vehicle: Automatic take off, terrain following, landing and wind reaction,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2004, no. 3, pp. 2339–2346, 2004.

[30] P. C. Merrell, D.-J. Lee, and R. W. Beard, “Statistical analysis of multiple optical flow values for estimation of unmanned aerial vehicle height above ground,” *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 5608, pp. 298–305, 2004.

[31] R. Brockers, S. Susca, D. Zhu, and L. Matthies, “Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs,” *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 8387, 2012.

[32] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, “Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow,” *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 801–806, 2008.

[33] J. Kehoe, A. Watkins, R. Causey, and R. Lind, “State estimation using optical flow from parallax-weighted feature tracking,” *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, vol. 8, pp. 5030–5045, 2006.

[34] R. J. D. Moore, S. Thurrowgood, and M. V. Srinivasan, “Vision-only estimation of wind field strength and direction from an aerial platform,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 4544–4549, 2012.

[35] “Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 957–964, 2012.

[36] D. Mercado, G. Flores, P. Castillo, J. Escareno, and R. Lozano, “Gps/ins/optic flow data fusion for position and velocity estimation,” *International Conference on Unmanned Aircraft Systems, ICUAS - Conference Proceedings*, pp. 486–491, 2013.

[37] M. Bibuli, M. Caccia, and L. Lapierre, “Path-following algorithms and experiments for an autonomous surface vehicle,” *Control Applications in Marine Systems*, vol. 7, no. 1, pp. 81–86, 2007.

[38] S. Ahrens, D. Levine, G. Andrews, and J. P. How, “Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2643–2648, 2009.

[39] A. Shabayek, C. Demonceaux, O. Morel, and D. Fofi, “Vision based uav attitude estimation: Progress and insights,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, no. 1-4, pp. 295–308, 2012.

[40] H. Chao, Y. Gu, and M. Napolitano, “A survey of optical flow techniques for robotics navigation applications,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 73, no. 1-4, pp. 361–372, 2014.

[41] L. Fusini, T. Fossen, and T. Johansen, “A uniformly semiglobally exponentially stable nonlinear observer for GNSS- and camera-aided inertial navigation,” *22nd Mediterranean Conference of Control and Automation (MED)*, pp. 1031–1036, 2014.

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

[43] C. S. Fuh and P. Maragos, “Region-based optical flow estimation,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings CVPR.*, pp. 130–135, 1989.

[44] J. Sarvaiya, S. Patnaik, and S. Bombaywala, “Image registration by template matching using normalized cross-correlation,” *International Conference on Advances in Computing, Control and Telecommunication Technologies*, pp. 819–822, 2009.

[45] S. Hutchinson, G. Hager, and P. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[46] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*. Stamford, CT: Cengage Learning, 2008.

[47] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*. New York: Wiley, 1995.

[48] M. Shuster and S. Oh, “Three-axis attitude determination from vector observations,” *Journal of Guidance, Control and Dynamics*, vol. 4, no. 1, pp. 70–77, 1981.

[49] A. Loria and E. Panteley, “Cascaded nonlinear time-varying systems: analysis and design,” in *Advanced Topics in Control Systems Theory*. London: Springer-Verlag (F. Lamnabhi-Lagarrigue, A. Loria and E. Panteley Eds.), 2004, ch. 2, pp. 23–64.

[50] H. Grip, A. Saberi, and T. Johansen, “Observers for interconnected nonlinear and linear systems,” *Automatica*, vol. 48, no. 7, pp. 1339–1346, 2012.

[51] G. Bradski, “The opencv library,” *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

Appendix

The parameter projection $\mathrm{Proj}(\cdot,\cdot)$ is defined as:

$$\mathrm{Proj}(\hat{b}^b,\tau)=\begin{cases}\left(I-\frac{c(\hat{b}^b)}{\|\hat{b}^b\|^2}\hat{b}^b\hat{b}^{bT}\right)\tau, & \|\hat{b}^b\|\geq L_b,\ \hat{b}^{bT}\tau>0\\ \tau, & \text{otherwise}\end{cases}$$

where $c(\hat{b}^b) = \min\{1, (\|\hat{b}^b\|^2 - L_b^2)/(L_{\hat{b}^b}^2 - L_{b^b}^2)\}$. This operator is a special case of that from Appendix E of [47].

Appendix B

Paper Submitted to AIAA SciTech 2016

This appendix includes the paper submitted to the 2016 American Institute of Aeronautics and Astronautics Science and Technology Forum and Exposition (AIAA SciTech 2016).

A Vision-aided Nonlinear Observer for Fixed-wing UAV Navigation

Jesper Hosen*, Håkon H. Helgesen*, Lorenzo Fusini†, Thor I. Fossen† and Tor A. Johansen†
*Norwegian University of Science and Technology
Department of Engineering Cybernetics
Trondheim, Norway
E-mail: {jesperhosen, haakon.helgesen}@gmail.com
†Norwegian University of Science and Technology
Centre for Autonomous Operations and Systems
Trondheim, Norway
E-mail: {lorenzo.fusini, thor.fossen, tor.arne.johansen}@itk.ntnu.no

Abstract—**This paper presents a vision-aided uniformly semi-globally exponentially stable (USGES) nonlinear observer for estimation of attitude, gyro bias, position, velocity and specific force of a fixed-wing Unmanned Aerial Vehicle (UAV). The nonlinear observer uses measurements from an Inertial Measurement Unit (IMU), a Global Navigation Satellite System (GNSS) receiver, and a video camera. This paper present a nonlinear observer representation with a computer vision (CV) system without any assumptions related to the depth in the images and the structure of the terrain being recorded. The CV utilizes a monocular camera and the continuous epipolar constraint to calculate body-fixed linear velocity. The observer is named a Continuous Epipolar Optical Flow (CEOF) nonlinear observer. Experimental data from a UAV test flight and simulated data are presented showing that the CEOF nonlinear observer has robust performance. Experimental results are compared with an Extended Kalman Filter (EKF) and illustrate that the estimates of the states converges accurately to the correct values. Results show that using the proposed CV in addition to IMU and GNSS improves the accuracy of the estimates. The CV provides accurate information about the direction of travel of the UAV, which improves the attitude and gyro bias estimate.**

I. Introduction

The use of Unmanned Aerial Vehicles (UAV) has in the last decade gained an increasingly interest, and already plays a major role in military use. The field of applications for UAVs will grow even more in the future, and the demands for robustness, safety and reliability are considered to be crucial. Robust navigation is one of the most important parts when working with UAVs. A challenge in navigation systems is to maintain accurate estimates of the states with low-cost measurement units. The output of such low-cost sensors are typically contaminated by noise and bias. As it is desirable to have low energy consumption on UAVs, it is necessary to find light weight navigation systems with good performance. The Kalman filter has been the preferred filter algorithm, but in recent years nonlinear observers, like the nonlinear complementary filter, have gained increased attention [1]–[6].

The use of cameras for navigational purposes is expected to grow quickly since video cameras are lightweight, energy efficient and the prices are constantly decreasing. As magnetometers are sensitive to disturbances, such as electromagnetic fields [7], cameras might be a good alternative or complementary to the magnetometer. The camera images can be used to output the body-fixed velocity of a UAV [8], but depend on favourable atmospheric conditions, light and detection of visual stationary features.

Computer Vision and Optical flow (OF) have been used for different applications in UAV navigation including indoor maneuvering [9], [10], linear and angular velocity estimation [8], [11], [12] and obstacle avoidance [9], [13]–[17], as well as height above the ground estimation in [18]. [19], [20] uses OF in landing assistance for UAVs without external sensor inputs. OF from a single camera is used in [21], [22] to estimate body axes angular rates of an aircraft as well as wind-axes angles. [12], [23], [24] have used OF as input in Kalman filter-based navigation systems, fusing OF measurements with acceleration and angular velocity measurements. [25], [26] have used camera as sensor for navigating in GPS-denied environments.

Attitude estimation has received significant attention as a stand-alone problem [1], [27]–[35]. In addition, other researchers have integrated Inertial Navigation System (INS), magnetometer/compass and GNSS to estimate the navigation states of a vehicle. [4] expanded the vector-based observer proposed by [1] and [32] to include GNSS velocity measurements. [27] and [28] built globally exponentially stable (GES) attitude estimators based on multiple time-varying reference vectors or a single persistently exiting vector. A similar observer was developed in [5], [36] to include also gyro bias and GNSS integration. [3] extended [36] to use linear velocity and specific force as reference vectors. [3] proved that feedback of estimated NED velocity and specific force in NED from the translational motion observer to the attitude observer, yield USGES in the origin of the error dynamics.

In this paper the observer presented in [3] is denoted as Ground Truth Optical Flow (GTOF) nonlinear observer. By assuming known distance to every feature in the camera image, the body-fixed velocity was recovered from the relationship between ego-motion and theoretical optical flow. This relationship is called the GTOF relationship between velocity and OF. The distance to every point was recovered by assuming flat horizontal terrain coinciding with NED, measured distance to the terrain by a laser altimeter and measured attitude of the UAV relative to NED by an inclinometer. The assumption of flat and horizontal terrain will cause the CV in the GTOF nonlinear observer to produce erroneous velocity measurements in the case of flying over rugged terrain. Therefore it is desirable to exchange the CV of the GTOF nonlinear observer with a CV system with no requirement of flat horizontal terrain.

Optical flow (OF) describes how objects in an image plane moves between two consecutive images. The motion in the image plane is caused by relative motion between the camera and the visual features being detected. In the simplest case it could be understood as the pixel displacement of a single feature between two successive images. The OF can be represented as multiple vectors describing the change in the image plane in time. Several methods exists for determining the OF of a series of images [37]–[40].

A camera fixed to a UAV can be used to recover the motion of the vehicle relative to the scene. An effective principle for recovering ego-motion of a camera is epipolar geometry. Epipolar geometry has been applied in e.g. navigation, landing and collision avoidance [12], [23], [41]–[45]. [46] presented the epipolar constraint in the continuous case. [47] and [48] have used the continuous epipolar constraint to recover the velocity of a UAV.

In this paper OF vectors together with the continuous epipolar constraint [46] are used to calculate the normalized body-fixed velocity of the UAV, and fed into the nonlinear observer as a reference vector. The use of the continuous epipolar constraint eliminates the dependency on the depth in the image. In practice this means that prior information about the distance and structure of the terrain are not required any more. Thus the observer is applicable when flying over any terrain.

A. Contribution of this Paper

This paper presents a more robust CV subsystem for the nonlinear observer from [3]. In [3] the ground truth optical flow (GTOF) relationship between motion and OF were used to recover the ego-motion of the UAV. A fundamental restriction from [3] was that the distance to every feature corresponding to an OF vector must be known in order to calculate the body-fixed linear velocity. The CV in this paper utilizes epipolar geometry [49] and only depends on the angular velocity of the UAV. Furthermore it works without knowing the distance to the features in the image. To the authors knowledge, this is the first time the continuous epipolar constraint has been employed in a USGES nonlinear observer.

Experimental and simulated results show that the proposed CEOF observer has comparable performance with the GTOF observer from [3] when flying over flat horizontal terrain. Simulations show that the proposed CEOF observer is structure independent, and that it outperforms the GTOF observer when flying above rugged and elevated terrain. Moreover, results show that using CV increases the accuracy of the estimates, compared to using only IMU and GNSS measurements. This is particularly clear in the attitude, as CV provides information about the direction of the body-fixed velocity. A pure IMU and GNSS approach assumes zero crab and flight path angle, and thus looses important information about the attitude. The experimental results are compared to an EKF, while the simulated results are compared to the known reference. The results imply that the CEOF observer is a robust option to the GTOF nonlinear observer.

The last contribution is a stability proof showing that the CEOF observer has the same stability properties as the GTOF observer, namely a USGES equilibrium point at the origin of the error dynamics.

II. Notation and Preliminaries

Matrices and vectors are represented by uppercase and lowercase letters respectively. X^{-1} and X^{+} denote the inverse and the pseudoinverse of a matrix respectively, X^T the transpose of a matrix or vector, $\hat{X}$ the estimated value of X, and $\tilde{X} = X - \hat{X}$ the estimation error. $\|\cdot\|$ denotes the Euclidean norm, $I_{n\times n}$ the identity matrix of order n, and $0_{m\times n}$ the $m \times n$ matrix of zeros. A vector $x = [x_1, x_2, x_3]^T \in \mathcal{R}^3$ is represented in homogeneous coordinates as $\underline{x} = [x_1, x_2, x_3, 1]^T$. The function $\text{sat}(\cdot)$ performs a component-wise saturation of its vector or matrix argument to the interval $[-1, 1]$. The operator $[x]_\times$ transforms the vector x into the skew-symmetric matrix

$$[x]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

The inverse operation is denoted as $\text{vex}(\cdot)$, such that $\text{vex}([x]_\times) = x$. The determinant of a matrix A is denoted $\det(A)$. The skew symmetric part of a square matrix A is obtained by the operator $\mathbb{P}_a(A) = \frac{1}{2}(A - A^T)$.

The North-East-Down, camera- and the body-fixed reference frames are used in this paper as shown in Fig. 1: the body-fixed frame are denoted {B} and the North-East-Down (NED) frame denoted {N} (Earth-fixed, considered inertial), while the camera frame is denoted {C}. The rotation from {B} to {N} is represented by the matrix $R_b^n \in SO(3)$, with $SO(3)$ representing the Special Orthogonal group. The image plane is denoted {M}. {B} and {C} are assumed to be aligned, ie. the camera is strapped to the body.

A vector decomposed in {B} and {N} has superscript ^{b} and ^{n} respectively. The subscript of a vector indicates which frame is measured relative to what. For instance $p^n_{b/n}$ is the position of {B} relative to {N} expressed in {N}. The camera location w.r.t. {N} is described by $c^n = [c^n_x, c^n_y, c^n_z]^T$. A point in the environment expressed w.r.t. {N} is $p^n = [x^n, y^n, z^n]^T$. The same point expressed in {C} is $p^c = [x^c, y^c, z^c]^T$. It will also be assumed that every point is fixed w.r.t. {N}. The Greek letters ϕ, θ, and ψ represent the roll, pitch, and yaw angles respectively, defined according to the zyx convention for principal rotations [6], and they are collected in the vector $\Theta_{b/n} = [\phi, \theta, \psi]^T$. A 2-D camera image has coordinates $x^m = [r, s]^T$, aligned with the y^b- and x^b-axis respectively (see Fig. 3). The corresponding homogeneous image coordinate is denoted $\underline{x}^m = [r, s, 1]^T$. The derivative $[\dot{r}, \dot{s}]^T$ of the image coordinates is the OF. The subscript $_{cv}$ indicates a quantity evaluated by means of the computer vision, $_{\text{imu}}$ indicates a quantity measured by the IMU, while $_{\text{GPS}}$ indicates that the quantity is measured by the GNSS.

A. Measurements and Sensors

The observer is designed to take use of a IMU, a GPS receiver and a video camera, providing the following measurements:

- *GPS*: NED position p^n and NED velocity v^n.
- *IMU*: biased angular velocity $\omega^b_{\text{imu}} = \omega^b_{b/n} + b^b_{\text{gyro}}$, where b^b_{gyro} represents the gyro bias, and specific force $f^b_{\text{imu}} = f^b_{b/n}$.
- *Camera*: 2-D projections $x^m = [r, s]^T$ onto the image plane {M} of points $[x^n, y^n, z^n]^T$ in {N}.

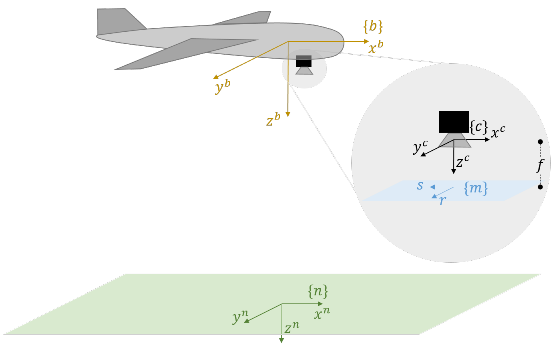


Fig. 1. Body frame is denoted {B}, camera frame is denoted {C}, and NED frame is denote {N}. Points in the terrain are projected by the pinhole camera model onto the image plane {M}, as illustrated by the blue plane.

Detailed information on the actual sensors employed in the experiment is presented in Section V.

III. Computer Vision

The observer presented in Section IV depends on body-fixed velocity measurements from the onboard camera. These measurements are generated through OF, therefore it is necessary to compute the OF vectors for consecutive images before these vectors are transformed to velocity measurements. The OF calculation and the transformation are presented in the forthcoming section.

A. *Optical flow computation*

There exist several methods for computing OF. For the experiment presented in Section V two specific methods are chosen. The first one is SIFT [39] which provided the overall best performance in [8]. The second method is a region matching-based method [8], namely template matching utilising cross-correlation [50]. SIFT uses a feature-based approach to compute OF. A set of features are extracted from two consecutive images with a feature detector. The detected features are then matched together to find common features in successive images. An OF vector is created from the displacement of each feature. The total number of such vectors in each image depends on the number of features detected and successfully matched.

It is desired to make sure that the OF algorithm produces at least two OF vectors to calculate the body-fixed velocity. It is not possible to guarantee a given number of vectors with SIFT since homogeneous environments, like snow or the ocean, increase the difficulty of finding distinct features. Therefore the OF vectors created by SIFT are combined with OF vectors from template matching [51]. The displacement of twelve templates, created symmetrically across the images, are used to find twelve OF vectors. Template matches below a given threshold are discarded and the corresponding OF vectors removed. Unreliable matches can occur in case of homogeneous terrain, changes in brightness or simply when the area covered by the template has disappeared from the image in the time between the capture of images.

The combination of two individual OF methods increases the probability of having OF vectors distributed across the whole image, as well as maintaining a high number of OF vectors. An example of OF vectors computed with SIFT and template matching from UAV test flights is displayed in Fig. 2.

In case of mismatches, both methods create erroneous OF vectors. It is desired to locate and remove these vectors. Therefore a simple outlier detector is implemented before the vectors are used to calculate body-fixed velocities. The outlier detector utilizes a histogram to find the vectors that deviates from the mean with respect to direction and magnitude.

B. *Transformation from optical flow to velocity*

For the OF computations to be useful in the observer a transformation to body-fixed velocity is necessary. The transformation is motivated by the continuous epipolar constraint and the pinhole camera model [52]. The camera-fixed coordinate system, {C}, is related to {N} as illustrated in Fig. 3. The focal point of the camera is for simplicity assumed to coincide with the origin of {B}. A point p in the terrain is projected from {C} to {M} by the pinhole camera model by

$$\underline{x}^m = \frac{1}{z^c} K p^c \tag{1}$$

where $\underline{x}^m$ is the homogeneous image coordinate and K is a projection matrix mapping points in the

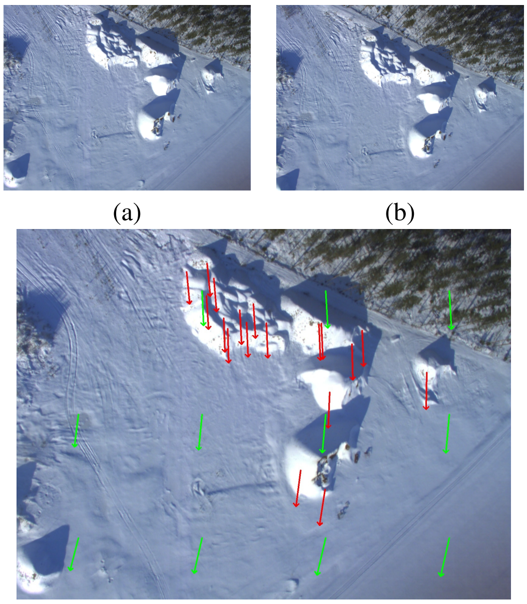


Fig. 2. a) Image captured at time t_0. b) Image captured at time $t_0 + \Delta t$. c) Optical flow vectors between image a) and b), generated by SIFT (red) and Template Matching (green).

camera frame to the image plane. It is defined as

$$K = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where f is the focal length of the camera. The focal length of a camera can be verified by the computer vision toolbox in Matlab. The same toolbox can be used to estimate coefficients describing the distortion of the camera. These coefficients can be used to generate undistorted images. For the rest of this paper, it is assumed that the distortion is insignificant.

u^c is defined as the back-projected point lying on the projection ray between the origin of $\{C\}$ and p^c with unity z-component

$$u^c = K^{-1}\underline{x}^m \tag{3}$$

Epipolar geometry [49] relates the motion of the camera frame with the motion in the image plane independent of the distance to the scene and the structure being recorded. By assuming that all

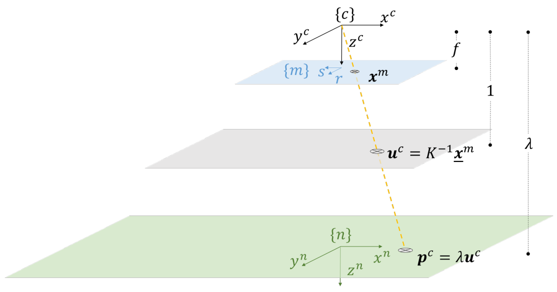

$\{c\}$
x^c
y^c
z^c
f
$\{m\}$ s r
$\boldsymbol{x}^m$
1
$\boldsymbol{u}^c = K^{-1}\underline{\boldsymbol{x}}^m$
λ
$\{n\}$
y^n
x^n
z^n
$\boldsymbol{p}^c = \lambda \boldsymbol{u}^c$

Fig. 3. Pinhole camera model. The camera frame is denoted $\{C\}$, image plane is illustrated in blue and denoted $\{M\}$ and NED frame is illustrated in green and denoted $\{N\}$. The gray plane is called the back projected plane. The back projected plane is located at unit length away from the camera frame in camera z-direction.

matched features are at rest w.r.t $\{N\}$, the continuous epipolar constraint [46] can be expressed as

$$\left(\dot{u}^{cT} + u^{cT}\left[\omega^c_{c/n}\right]_\times^T\right)\left(v^c_{c/n} \times u^c\right) = 0 \tag{4}$$

where $\omega^c_{c/n}$ and $v^c_{c/n} = [v_x, v_y, v_z]^T$ are the angular and linear velocity of the camera relative to $\{N\}$ expressed in $\{C\}$, respectively. Note that the epipolar geometry has an inherited sign ambiguity due to the fact that the scale is not preserved. This means that it is only possible to determine the body-fixed velocity up to scale.

Using now the properties of a triple product [53], (4) can be rewritten as

$$\begin{aligned} &{v^c_{c/n}}^T\left(u^c \times \left(\dot{u}^{cT} + u^{cT}\left[\omega^c_{c/n}\right]_\times^T\right)^T\right) \\ &= {v^c_{c/n}}^T\left(u^c \times \left(\dot{u}^{cT} + \left[\omega^c_{c/n}\right]_\times u^c\right)\right) = 0 \end{aligned} \tag{5}$$

(5) might be rewritten as a linear equation in $v^c_{c/n}$. The crossproduct term is defined as:

$$c := u^c \times \left(\dot{u}^{cT} + \left[\omega^c_{c/n}\right]_\times u^c\right) = [c_x, c_y, c_z]^T$$

If the angular velocity is measured, then all quantities in the crossproduct term c are known. Using the definition of c, (5) is rewritten as

$$ {v^c_{c/n}}^T c = c^T v^c_{c/n} = 0 \tag{6}$$

Assuming that a fixed-wing UAV will never have zero forward velocity, then since $\{C\}$ and $\{B\}$ are aligned, one can divide (6) by the forward velocity component $v_x \neq 0$

$$\frac{1}{v_x} c^T \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = c^T \begin{bmatrix} 1 \\ \frac{v_y}{v_x} \\ \frac{v_z}{v_x} \end{bmatrix} = c_x + [c_y, c_z] \begin{bmatrix} \frac{v_y}{v_x} \\ \frac{v_z}{v_x} \end{bmatrix} = 0$$

$$[c_y, c_z] \begin{bmatrix} \frac{v_y}{v_x} \\ \frac{v_z}{v_x} \end{bmatrix} = -c_x \tag{7}$$

As can be seen from (7), one ends up with a linear equation. Assuming N features, the scaled body-fixed velocity with unity forward component can be found as:

$$v^c_{c/n} = v_x A^+ b, \quad v_x \neq 0$$

$$A = \begin{bmatrix} c_{y,1} & c_{z,1} \\ \vdots & \\ c_{y,N} & c_{z,N} \end{bmatrix}$$

$$b = -\begin{bmatrix} c_{x,1} \\ \vdots \\ c_{x,N} \end{bmatrix}$$

$$u^c_j \times \left(\dot{u}^{cT}_j + \left[\omega^c_{c/n}\right]_\times u^c_j \right) = [c_{x,j}, c_{y,j}, c_{z,j}]^T \tag{8}$$

This gives a correct solution only if A has full rank. This can only happen if the OF algorithm chooses linearly independent feature points and OF vectors as defined in Def. 1. Linearly independent OF vectors are in general obtained by not choosing all features from the same line in the image plane. $u^c_j = K^{-1}\underline{x}^m_j$ and $\dot{u}^c_j = K^{-1}[\dot{r}_j, \dot{s}_j, 0]^T$ are the back projected coordinate and OF of feature j respectively. Recall the sign ambiguity of the epipolar geometry, meaning that one must know the sign of v_x to recover the normalized linear velocity. For a fixed-wing UAV the forward velocity will always be greater than zero, $v_x > 0$.

Definition 1. ***Linearly Independent Optical Flow Vectors***
A pair of image features and their corresponding optical flow vectors x^m_1, $\dot{x}^m_1$ and x^m_2, $\dot{x}^m_2$, are said to be linearly independent if and only if the rank of A in (8) *is full, yielding $[v_y, v_z]^T = v_x A^+ b$ to be uniquely defined. The rank is full if and only if some 2×2 sub-matrix of A, $A_{2\times2}$, has det($A_{2\times2}$) $\neq 0$.*

IV. Observer Design

A. Kinematics

The kinematics of attitude, position, and velocity are described by

$$\dot{R}^n_b = R^n_b \left[\omega^b_{b/n}\right]_\times \tag{9a}$$

$$\dot{p}^n_{b/n} = v^n_{b/n} \tag{9b}$$

$$\dot{v}^n_{b/n} = f^n_{b/n} + g^n \tag{9c}$$

The objective is to estimate the attitude R^n_b, the position $p^n_{b/n}$, and the velocity $v^n_{b/n}$ with exponential convergence rate. In addition to this, an estimator for the gyro bias b^b_{gyro} is also provided.

B. Assumptions

The observer design by [3] is based on the following assumptions:

Assumption 1. *The gyro bias b^b_{gyro} is constant, and there exists a known constant $L_b > 0$ such that $\|b^b_{\text{gyro}}\| \leq L_b$.*

Assumption 2. *There exists a constant $c_{obs} > 0$ such that, $\forall t \geq 0$, $\|v^b_{cv} \times f^b_{\text{imu}}\| \geq c_{obs}$.*

Assumption 2 states that the UAV cannot have a specific force parallel to the velocity of the UAV. Furthermore neither the specific force nor the velocity can be identically equal to zero. In practice this condition restricts the types of maneuvers that ensure guaranteed performance of the proposed observer. This is however not a problem for fixed-wing UAVs as they always have forward speed to remain airborne. Moreover the observer does not converge while the vehicle is at rest without aiding from e.g. a magnetometer, but presents no issues during flight.

For the CEOF observer, two assumptions are introduced to ensure that CV can recover the body-fixed velocity.

Assumption 3. *The UAV has forward body-fixed velocity, $v_x > 0$.*

Assumption 4. *The OF algorithm provides at least two linearly independent OF vectors, as defined in Def.1.*

C. Observer Equations

Provided Assumptions 1-4 hold, the CEOF observer representation is stated as

$$\Sigma_1 \begin{cases} \dot{\hat{R}}_b^n = \hat{R}_b^n S(\omega_{\text{imu}}^b - \hat{b}_{\text{gyro}}^b) + \sigma K_P \hat{J} \\ \dot{\hat{b}}_{\text{gyro}}^b = \text{Proj}(\hat{b}_{\text{gyro}}^b, -k_I \text{vex}(\mathbb{P}_a(\hat{R}_s^T K_P \hat{J}))) \end{cases} \tag{10}$$

$$\Sigma_2 \begin{cases} \dot{\hat{p}}_{b/n}^n = \hat{v}_{b/n}^n + K_{pp}(p_{\text{GPS}}^n - \hat{p}_{b/n}^n) \\ \qquad + K_{pv}(v_{\text{GPS}}^n - \hat{v}_{b/n}^n) \\ \dot{\hat{v}}_{b/n}^n = \hat{f}_{b/n}^n + g^n + K_{vp}(p_{\text{GPS}}^n - \hat{p}_{b/n}^n) \\ \qquad + K_{vv}(v_{\text{GPS}}^n - \hat{v}_{b/n}^n) \\ \dot{\xi} = -\sigma K_P \hat{J} f_{\text{imu}}^b + K_{\xi p}(p_{\text{GPS}}^n - \hat{p}_{b/n}^n) \\ \qquad + K_{\xi v}(v_{\text{GPS}}^n - \hat{v}_{b/n}^n) \\ \hat{f}_{b/n}^n = \hat{R}_b^n f_{\text{imu}}^b + \xi \end{cases} \tag{11}$$

$$\text{CV} \begin{cases} v_{cv}^b = \text{sign}(v_x) \frac{v_e}{\|v_e\|} \\ v_e = \frac{v_{b/n}^b}{v_x} = [1, (A^+ b)^T]^T, \quad v_x \neq 0 \\ u_j^c \times \left(\dot{u}_j^{cT} + \left[\omega_{\text{imu}}^b - \hat{b}_{\text{gyro}}^b \right]_\times u_j^c \right) \\ \quad = [c_{x,j}, c_{y,j}, c_{y,j}]^T \end{cases} \tag{12}$$

The subsystem Σ_1 represents the attitude observer, whereas Σ_2 represents the translational motion observer. The CV gives (12), together with (8). $\sigma \geq 1$ is a scaling factor tuned to achieve stability, k_I is a positive scalar gain and K_P is a symmetric positive definite gain matrix. $\text{Proj}(\cdot,\cdot)$ represents a parameter projection [54] that ensures that $\|\hat{b}_{\text{gyro}}^b\|$ does not exceed a design constant $L_{\hat{b}} > L_b$ (see Appendix A), and $\hat{R}_s = \text{sat}(\hat{R}_b^n)$. K_{pp}, K_{pv}, K_{vp}, K_{vv}, $K_{\xi p}$, and $K_{\xi v}$ are observers gains, and g^n is the gravity vector in {N}. The matrix $\hat{J}$ is the output injection term, whose design is inspired by the TRIAD algorithm [55] and defined as

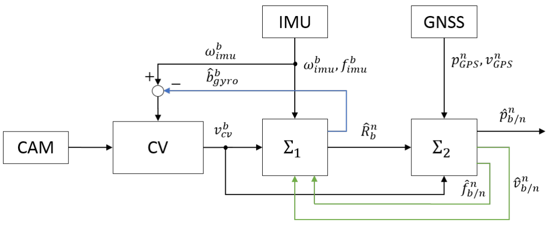


Fig. 4. Block diagram of the observer. Σ_1 represents the attitude observer, and Σ_2 the translational motion observer. The feedback illustrated in green have been proved to yield USGES stability of the nonlinear observer. The stability of the gyrob bias feedback illustrated in blue has not been analysed.

$$\hat{J}(v_{cv}^b, \ \hat{v}_{b/n}^n, f_{\text{imu}}^b, \hat{f}_{b/n}^n, \hat{R}_b^n) := \hat{A}_n A_b^T - \hat{R}_b^n A_b A_b^T \tag{13a}$$

$$A_b := [f_{\text{imu}}^b, \ f_{\text{imu}}^b \times v_{cv}^b, \ f_{\text{imu}}^b \times (f_{\text{imu}}^b \times v_{cv}^b)] \tag{13b}$$

$$\hat{A}_n := [\hat{f}_{b/n}^n, \ \hat{f}_{b/n}^n \times \hat{v}_{b/n}^n, \ \hat{f}_{b/n}^n \times (\hat{f}_{b/n}^n \times \hat{v}_{b/n}^n)] \tag{13c}$$

The system Σ_1–Σ_2 is a feedback interconnection, as illustrated by Fig. 4.

D. Stability Proof

The error dynamics of the nonlinear observer can be written in a compact form as

$$\Sigma_1 \begin{cases} \dot{\tilde{R}}_b^n = R_b^n \left[\omega_{b/n}^b \right]_\times - \hat{R}_b^n \left[\omega_{\text{imu}}^b - \hat{b}_{\text{gyro}}^b \right]_\times - \sigma K_P \hat{J} \\ \dot{\tilde{b}}_{\text{gyro}}^b = -\text{Proj}(\hat{b}_{\text{gyro}}^b, -k_I \text{vex}(\mathbb{P}_a(\hat{R}_s^T K_P \hat{J}))) \end{cases} \tag{14a}$$

$$\Sigma_2 \left\{ \dot{\tilde{w}} = (A_w - K_w C_w)\tilde{w} + B_w \tilde{d} \right. \tag{14b}$$

where $\tilde{w} = [(\tilde{p}_{b/n}^n)^T, (\tilde{v}_{b/n}^n)^T, (\tilde{f}_{b/n}^n)^T]^T$ collects the estimated position, velocity and acceleration vectors, $\tilde{d} = \left(R_b^n \left[\omega_{b/n}^b \right]_\times - \hat{R}_b^n \left[\omega_{\text{imu}}^b - \hat{b}_{\text{gyro}}^b \right]_\times \right) f_{b/n}^b + \left(R_b^n - \hat{R}_b^n \right) \dot{f}_{b/n}^b$, and the four matrices in (14b) are defined as

$$A_w = \begin{bmatrix} 0_{6\times 3} & I_6 \\ 0_{3\times 3} & 0_{3\times 6} \end{bmatrix}, \qquad B_w = \begin{bmatrix} 0_{6\times 3} \\ I_3 \end{bmatrix},$$

$$C_w = \begin{bmatrix} I_6 & 0_{6\times 3} \end{bmatrix}, \qquad K_w = \begin{bmatrix} K_{pp} & K_{pv} \\ K_{vp} & K_{vv} \\ K_{\xi p} & K_{\xi v} \end{bmatrix}.$$

The following theorem can be stated about the stability of the nonlinear observer (10)-(12), if assuming that $\hat{b}^b_{\text{gyro}}$ is kept constant in (12).

Theorem 1. ***(Stability of the CEOF observer)*** *Let σ be chosen to ensure stability according to Lemma 1 in [5] and define $H_K(s) = (Is - A_w + K_w C_w)^{-1} B_w$. There exists a set (0,c) such that, if K_w is chosen such that $A_w - K_w C_w$ is Hurwitz, and $\|H_K(s)\|_\infty < \gamma$, for $\gamma \in (0, c)$, then the origin of the error dynamics* (10)-(12)*, provided Assumptions 1-4, is USGES when the initial conditions satisfy $\|\hat{b}^b_{\text{gyro}}(0)\| \leq L_{\hat{b}}$.*

Proof: Proof is based on Theorem 1 in [3], where we have replaced M with the new computer vision subsystem from (12). We must show that v^b_{cv} is uniquely defined. Then it follows from Theorem 1 in [3] that the origin of the error dynamics (10)-(12) is USGES.

v^b_{cv} has a one to one mapping to the scaled body-fixed velocity with unit forward component v_e. Moreover if the sign of v_x is known, then $v^b_{cv} = \frac{v^b_{b/n}}{\|v^b_{b/n}\|}$. From Assumption 3 $v_x > 0$, hence the uniqueness of v^b_{cv} can be shown by the uniqueness of v_e. $v_e = [1, (A^+ b)^T]^T$ has a unique solution if and only if the rank of A is full [53]. Given that the computer vision algorithm extracts features such that Assumption 4 is not violated, then A has full rank, and v_e is uniquely determined. Hence v^b_{cv} is uniquely determined, and it follows from Theorem 1 in [3] that the system is USGES. ■

V. Experimental Results

An experiment is carried out to validate the theory in practice. The UAV employed is a UAV Factory Penguin-B, equipped with a custom-made payload that includes all the necessary sensors. The IMU is a Sensonor STIM300, a low-weight, tactical grade, high-performance sensor that includes gyroscopes, accelerometers, and inclinometers, all recorded at a frequency of 300 Hz. The chosen GPS receiver is a uBlox LEA-6T, which gives measurements at 5 Hz. The video camera is an IDS GigE uEye 5250CP provided with a 8mm lens. The camera is configured for a hardware-triggered capture at 10 Hz. The experiment has been carried out on 6 February 2015 at the Eggemoen Aviation and Technology Park, Norway, in a sunny day with good visibility, very little wind, an air temperature of about -8°C. The terrain is relatively flat and covered with snow.

The observer is evaluated offline with the flight data gathered at the experiment. It is implemented using first order forward Euler discretisation with a time-varying step depending on the interval of the data acquisition of the fastest sensor, namely the STIM300, and it is typically around 0.003 seconds. The gyro bias is initialized by averaging the gyroscope measurement at stand still before take-off. The position estimate is initialized by using the first GPS measurement, while the NED velocity is initialized by the difference between the two first consecutive GPS measurements. The various parameters and gains are chosen as $L_b = 2°/\text{s}$, $L_{\hat{b}} = 2.1°/\text{s}$, $\sigma = 1$, $K_P = \text{diag}[0.08, 0.04, 0.06]$, $k_I = 0.0001$, $K_{pp} = 30 I_{3\times3}$, $K_{pv} = 2 I_{3\times3}$, $K_{vp} = 0.01 I_{3\times3}$, $K_{vv} = 20 I_{3\times3}$, $K_{\xi p} = I_{3\times3}$, and $K_{\xi v} = 50 I_{3\times3}$.

The reference provided for the attitude, position, and velocity is the output of the EKF of the autopilot mounted on the Penguin-B. A reference for the gyro bias is not available.

All the images are processed with a resolution of 1600×1200 (width×height) pixels and in their original state, without any filtering. The lens distortion of the camera is not accounted for, and no correction is applied to the images. SIFT is implemented with the open source computer vision library (OpenCV) [56] with default settings. Each match is tagged with a value indicating the accuracy of the match, and the smallest of these values is considered to be the best match. To increase the reliability of the OF vectors, each match is compared to the best one. Every match with an uncertainty more than double the uncertainty of the best match is removed. Also the template matching algorithm is implemented with OpenCV. The size of the templates is chosen to be 120×90 pixels and a correlation of 99% is required in order for a template match to be considered reliable and not removed.

In addition to the CEOF and GTOF observer, a nonlinear observer without CV is implemented.

This is done by removing the CV subsystem in (12) from the nonlinear observer, and approximating the body-fixed linear velocity measurement by $v^b = [1, 0, 0]^T$. The nonlinear observer without CV is denoted NoCV. Although Theorem 1 does not cover feedback of the gyro bias estimate to CV in the CEOF nonlinear observer, this feedback is implemented. This is assumed to increase the accuracy without being destabilizing, as the bias estimator is tuned to have slow dynamics.

A. Results

The results presented here refer to a complete flight of the Penguin-B, from take-off to landing. The time on the x-axis is the elapsed time since the data logging began, and only the significant part involving the flight is presented. The maneuvers performed include flights on a straight line and turns with a large and small radius of curvature, approximately 200 m and 100 m.

Fig. 5 shows the measured body-fixed velocity from the GTOF CV. The measurements are contaminated by noise. The mean values are close to the reference, although the mean forward velocity (u) is slightly greater than the reference. The measured crab and flight path angle of the UAV are shown in Fig. 6. It is seen that both the GTOF and CEOF CV succeeds in measuring the correct direction, but GTOF has a larger noise level than CEOF.

Fig. 7 illustrates the estimated attitude. It can be seen that all observers need approximately 60 seconds to converge. The estimates of the roll angle are fairly similar for NoCV, GTOF and CEOF. The estimated pitch angle has a small offset for all nonlinear observers throughout the entire flight. The yaw angle estimate is almost identical for the NoCV, GTOF and CEOF. Fig. 8 and Fig. 9 illustrates the estimated velocity and position in {N}, and shows small differences for NoCV, GTOF and CEOF. The estimated gyro bias is seen in Fig. 10. No bias reference is available, but the estimated bias is close to equal for NoCV, GTOF and CEOF. The flight terrain is relatively flat and the UAV has small crab and flight path angle during the flight. Therefore the weaknesses of the GTOF and NoCV observer are not significant in the results. However the experimental results show that the nonlinear observers yield small deviations from the reference EKF, and that the CV give reasonable estimates of normalized body-fixed velocity.

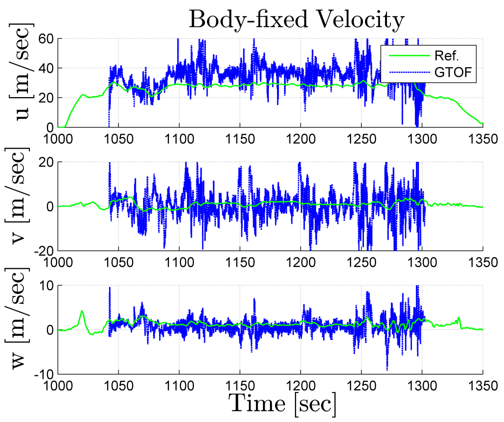


Fig. 5. Measured and estimated body-fixed velocity by GTOF and autopilot EKF respectively.

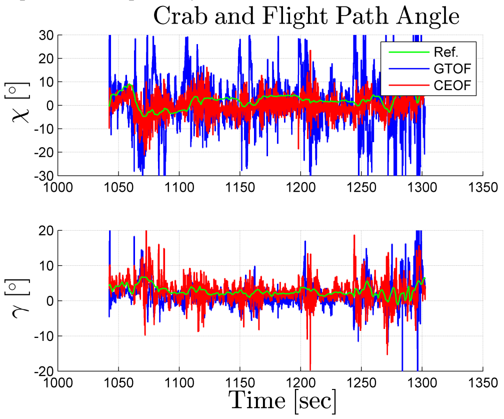


Fig. 6. Measured and estimated crab and flight path angle.

VI. Simulation Results

In order to evaluate the NoCV, GTOF and CEOF observer representations in the presence of more rugged terrain and to compare with an exactly known reference, a simulator is implemented in Matlab. An elevation profile of a coastline is generated, and a UAV flight is simulated.

The following parameters and gains are chosen identical for the NoCV, GTOF and CEOF observer: $L_b = 2°/\text{s}$, $L_{\hat{b}} = 2.1°/\text{s}$, $\sigma = 1$, $K_{pp} = \text{diag}[5, 5, 0.7]$, $K_{pv} = \text{diag}[50, 50, 50]$, $K_{vp} =$

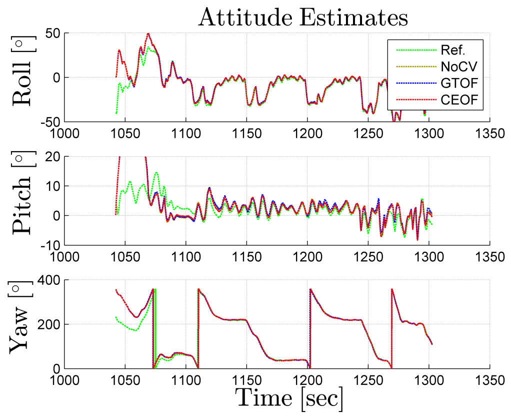


Fig. 7. Estimated attitude by the observers.

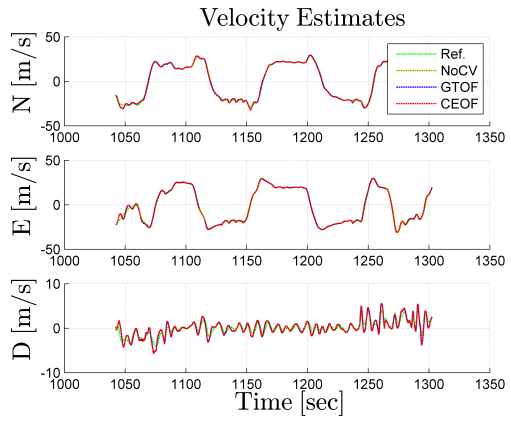


Fig. 8. Estimated velocity by the observers.

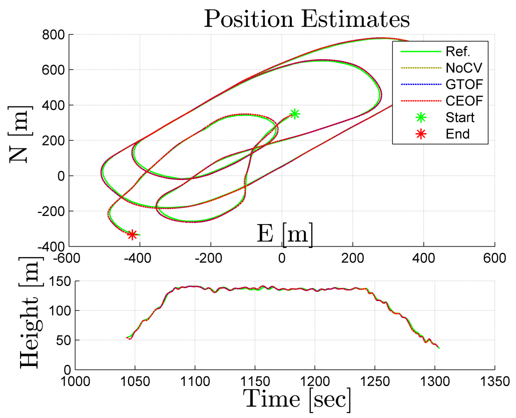


Fig. 9. Estimated position by the observers.

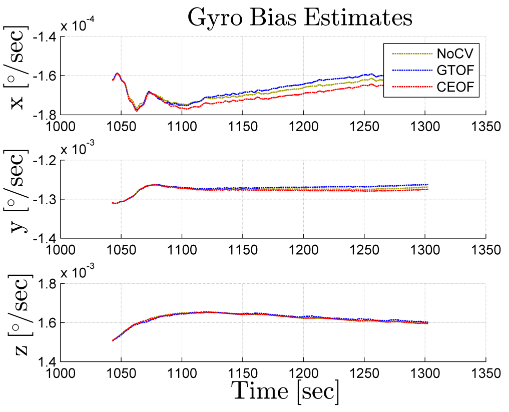


Fig. 10. Estimated gyro bias by the observers.

diag$[0.1, 0.1, 0.01]$, $K_{vv} = 10I_{3\times3}$, $K_{\xi p} = 0.1I_{3\times3}$, and $K_{\xi v} = 5I_{3\times3}$. For the GTOF and CEOF observer $K_P = I_{3\times3}$ and $k_I = 0.03$ are chosen. The NoCV is tuned with $K_P = \text{diag}[1, 0.2, 0.1]$ and $k_I = 0.01$.

All observers are initialised with $\hat{R}^n_b = I_{3\times3}$, $\hat{b}^b_{\text{gyro}} = 0_{3\times1}$. $\hat{p}^n_{b/n}$ and $\hat{v}^n_{b/n}$ are initialised as the first GNSS position and velocity measurement respectively.

A. *UAV Path*

Linear and angular velocity, $v^b_{b/n}$ and $\omega^b_{b/n}$, are generated over a time interval of $200\,\text{sec}$. Wind directed straight north with magnitude 5m/s is simulated causing the UAV to have a crab angle. As the camera measures the velocity relative to the ground, one does not have to consider the sideslip angle. Kinematic equations are used to generate positions and attitude of the UAV.

$$v^n_{b/n} = R^n_b(\Theta_{b/n})v^b_{b/n} \tag{15}$$

$$f^n_{b/n} = R^n_b(\Theta_{b/n})(\dot{v}^b_{b/n} + \omega^b_{b/n} \times v^b_{b/n}) - g^n \tag{16}$$

$$f^b_{b/n} = (R^n_b)^T(\Theta_{b/n})f^n_{b/n} \tag{17}$$

$$\dot{\Theta}_{b/n} = T_\Theta(\Theta_{b/n})\omega^b_{b/n} \tag{18}$$

$R^n_b(\Theta_{b/n})$ and $T_\Theta(\Theta_{b/n})$ being the rotation matrix between $\{\text{B}\}$ and $\{\text{N}\}$ and the angular transformation matrix respectively. The variables are integrated numerically with first order Euler integration

$$\Theta_{b/n}(k+1) = \Theta_{b/n}(k) + \delta t\dot{\Theta}_{b/n} \tag{19}$$

$$p^n_{b/n}(k+1) = p^n_{b/n}(k) + \delta t v^n_{b/n} \tag{20}$$

B. Sensor Data

Sensor data are generated before running the observer. A gyroscope, accelerometer, inclinometer, GNSS and CV are simulated. The GNSS is simulated to measure $\{N\}$ position and velocity, and CV is simulated to measure the OF. The gyroscope, accelerometer, inclinometer are configured to output measurements with a rate of 100 Hz. The GNSS is configured to output measurements at 5Hz. The noise on the position measurement from GNSS is modelled as a Gauss-Markov process by $\nu(k+1) = e^{-K_{\text{GNSS}}\Delta T}\nu(k) + \eta_{\text{GNSS}}$, with noise parameters given in Table I.

TABLE I. GAUSS-MARKOV ERROR MODEL PARAMETERS FOR GNSS POSITION MEASUREMENTS.

Direction	*Std. dev.* η_{GNSS}[m]	$1/K_{\text{GNSS}}$[s]	ΔT_{GNSS}[s]
North	0.21	360	0.2
East	0.21	360	0.2
Down	0.4	360	0.2

The camera is simulated to capture 25 frames per second. The camera extracts features and calculates OF as described in Appendix B. White noise is added to the IMU, inclinometer, camera and velocity from GNSS sensor data by the multivariate normal random noise-function, `mvnrnd`, in Matlab. Inclinometer measurements are denoted $\Theta^b_{incl} = [\phi, \theta]^T$. The following mean and covariance are used:

$$\begin{aligned}
w_{\omega^b_{\text{imu}}} &\sim \mathcal{N}(0_{3\times1}, \Sigma_{\omega^b_{\text{imu}}}), \Sigma_{\omega^b_{\text{imu}}} = (0.135\,\text{deg})^2 I_{3\times3} \\
w_{f^b_{\text{imu}}} &\sim \mathcal{N}(0_{3\times1}, \Sigma_{f^b_{\text{imu}}}), \Sigma_{f^b_{\text{imu}}} = (1.29\cdot10^{-3}\text{g})^2 I_{3\times3} \\
w_{\Theta^b_{incl}} &\sim \mathcal{N}(0_{2\times1}, \Sigma_{\Theta^b_{incl}}), \Sigma_{\Theta^b_{incl}} = (0.18\,\text{deg})^2 I_{2\times2} \\
w_{v^n_{GNSS}} &\sim \mathcal{N}(0_{3\times1}, \Sigma_{v^n_{GNSS}}), \Sigma_{v^n_{GNSS}} = (0.21\text{m/s})^2 I_{3\times3}
\end{aligned}$$

No bias on the accelerometer is assumed, and a constant bias is assumed on the gyroscope. The gyroscope is simulated with the following bias

$$b^b_{\text{gyro}} = \begin{bmatrix} 0.1\text{deg/s} \\ -0.3\text{deg/s} \\ -0.35\text{deg/s} \end{bmatrix}$$

White noise is also added to the OF data from the simulated camera. Every extracted feature is given white noise with variance, $\sigma^2_{dr} = \sigma^2_{ds} = \sigma^2_d = (4.5\cdot10^{-5}\text{mm})^2$. As two corresponding features are needed to get an OF vector, the resulting noise of the OF vector has variance $\sigma_{OF} = \sigma^2_d I_{2\times2}$. On a camera chip with 1600×1200 pixels and dimension 7.2×5.4 mm, this would yield a small variance of $(\sqrt{2}\cdot0.01\text{px})^2$ for the OF vector noise.

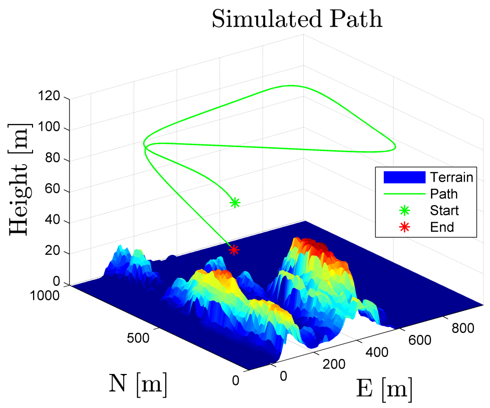


Fig. 11. The simulated UAV path and the elevation profile of the terrain model.

C. Terrain Simulation

In order to evaluate the performance of the GTOF and CEOF observer representations with a realistic environment, a terrain model is generated. The terrain model is a matrix, Z, with values corresponding to the elevation profile of the terrain. It is also called the elevation profile of the terrain, as it describes the elevation of the terrain. The terrain model is made to mimic a coastline, and has a resolution of $1\text{m}\times1\text{m}$ meter. The covered area is $1\text{km}\times1\text{km}$. At position x, y of the matrix the elevation h of the terrain at x meters North and y meters East is found. A point on the surface of the terrain will have NED coordinate $x, y, -h$. Fig. 11 displays the simulated UAV path and the terrain model.

D. Results

Fig. 12 shows the crab angle error and the flight path angle error in the measured normalized body-fixed velocity from CV. It can be seen that the GTOF fails to produce correct measurement of the body-fixed velocity when the terrain is non-planar (at time 110-220 seconds). Any crab and flight path angle of the UAV causes NoCV to fail as it assumes pure forward motion.

Fig. 13 and Fig. 14 show the attitude estimates and the error in the estimates. The NoCV observer fails to produce accurate estimates of the attitude. It is seen that the accuracy of the GTOF observer is heavily reduced when flying over the non-planar area. The CEOF observer on the other hand is not limited by the rugged terrain, and provides accurate estimates during the entire flight. The estimated and real gyro bias is displayed in Fig. 15. It is seen that the bias values from NoCV does not converge to the correct value. The shortcomings of the GTOF observer is again illustrated when the UAV flies over the non-planar area.

Fig. 16 and 17 show the real and estimated velocity and position. The estimates are close to the reference and quite similar for GTOF and CEOF. This is expected as the velocity and position measurements from GNSS have the largest influence on these estimates.

Table II provides numerical evaluation of the observers in means by the Root Mean Squared (RMS) error. The CEOF observer has lower RMS in the estimates of the attitude than the GTOF observer. NoCV has the least accurate estimates in attitude, and is outclassed by CEOF. There are no major differences in estimated position and velocity. However CV seem to slightly increase the accuracy in estimated position. The estimated gyro bias is most accurate with the CEOF and least accurate with NoCV. The crab angle and flight path angle error are reduced significantly with CV. This is because NoCV assumes zero crab and flight path angle, which is not the case.

Overall the CEOF observer proves to be much more reliable than GTOF and NoCV, with a robust and accurate performance. The GTOF performs better than NoCV, which supports the use of CV in the observer. However the validity of the GTOF observer is restricted to horizontal planar terrain, which limits the range of use in practice. CEOF is not restricted by the same limitations and thus more applicable in practice.

VII. Conclusions

In this paper two different vision-aided nonlinear observers, and one nonlinear observer without CV, for estimation of position, velocity and attitude have been evaluated on real experimental data obtained by flying a fixed-wing UAV with a custom-made payload of sensors. The nonlinear observers have also been tested on simulated data to compare the performance of the observers with the presence of non-planar terrain and with an exact known reference for comparison. The results show that using CV increases the accuracy of the nonlinear observer, especially in estimated attitude. This is because CV provides useful information about the direction of the body-fixed velocity. Furthermore the CEOF nonlinear observer has shown to be a more robust option than the GTOF nonlinear observer, as it is terrain independent.

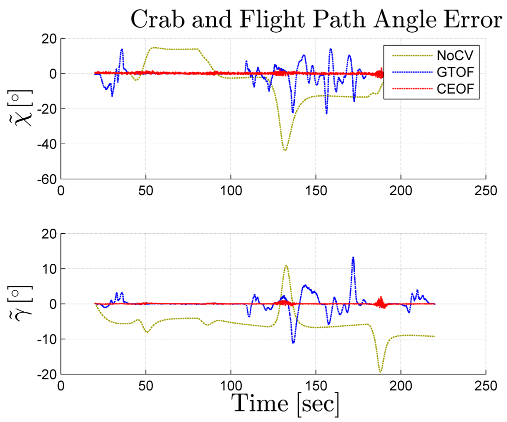


Fig. 12. Error in crab ($\tilde{\chi}$) and flight path angle ($\tilde{\gamma}$) for the measured normalized body-fixed velocity.

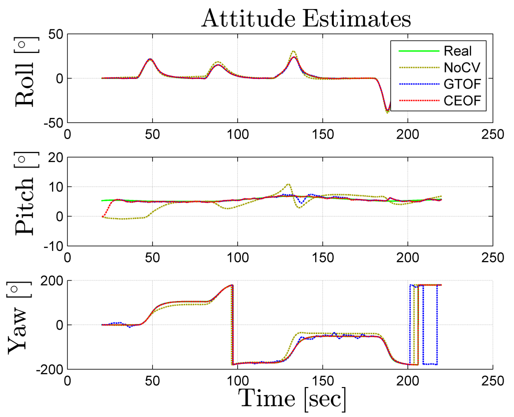


Fig. 13. Estimated attitude. When the UAV flies over the rugged terrain, the GTOF observer fails to produce correct estimates of the attitude.

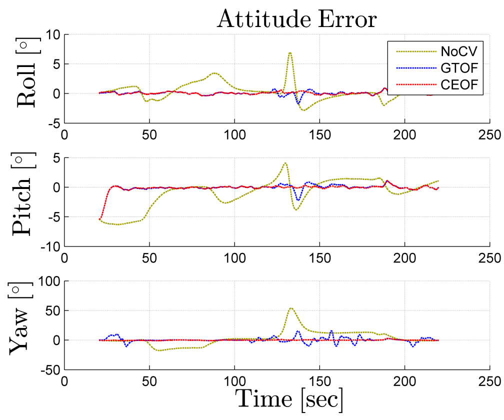


Fig. 14. Error in estimated attitude. When the UAV flies over the rugged terrain, the GTOF observer fails to produce correct estimates of the attitude. The NoCV observer fails to estimate correct pitch angle.

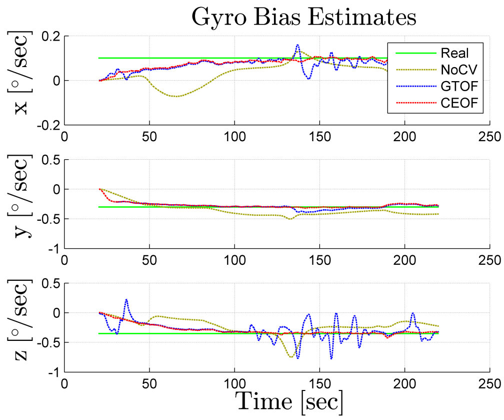


Fig. 15. Estimated gyro bias together with the real gyro bias. After 100 sec the gyro bias has converged. When flying over the rugged terrain the GTOF observer produces erroneous gyro bias estimates, while the CEOF observer is unaffected.

ACKNOWLEDGEMENTS

The authors are grateful for the assistance provided by the UAV engineers at NTNU and Maritime Robotics AS, in particular Lars Semb and Carl Erik Stephansen. Significant contributions to the construction of the UAV payload was made by the rest of the navigation team at NTNU, in particular Sigurd M. Albrektsen, Jakob M. Hansen and Kasper T. Borup.

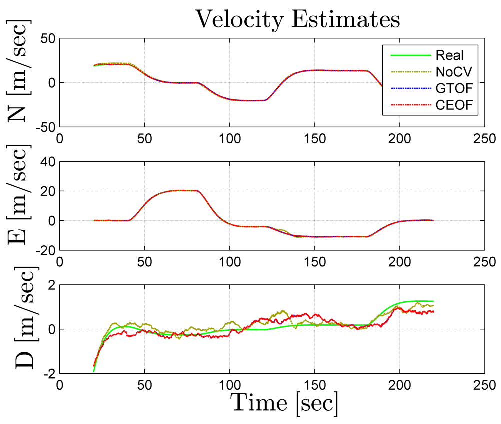


Fig. 16. Estimated velocity.

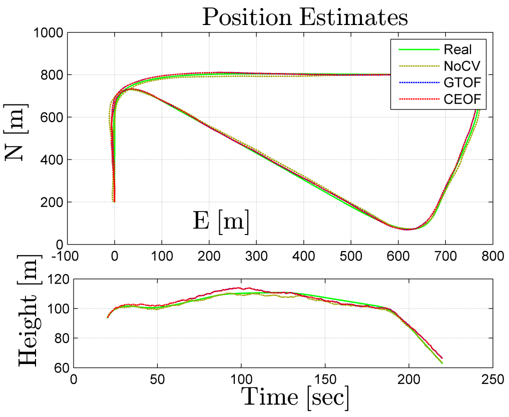


Fig. 17. Estimated position.

REFERENCES

[1] R. Mahony, T. Hamel, and J. M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. Xx, pp. 1203–1218, 2008.

[2] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of Nonlinear Attitude Estimation Methods," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[3] L. Fusini, T. I. Fossen, and T. A. Johansen, "A Uniformly Semiglobally Exponentially Stable Nonlinear Observer for GNSS-and Camera-Aided Inertial Navigation," in *Proceedings of the 22nd IEEE Mediterranean Conference on Control and Automation (MED'14)*, 2014.

[4] M. D. Hua, "Attitude estimation for accelerated vehicles using {GPS/INS} measurements," *Control Engineering Practice*, vol. 18, no. 7, pp. 723–732, 2010.

[5] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi, "A nonlinear observer for integration of {GNSS and IMU} measurements with gyro bias estimation," *American Control Conference (ACC)*, pp. 4607–4612, 2012.

TABLE II. RMS VALUES FOR THE ESTIMATED STATES IN THE DIFFERENT CASES USING THE GROUND TRUTH OPTICAL FLOW (GTOF) AND THE CONTINUOUS EPIPOLAR OPTICAL FLOW (CEOF) OBSERVER REPRESENTATION. $\tilde{\chi}$ AND $\tilde{\gamma}$ ARE THE CRAB ANGLE- AND FLIGHT PATH ANGLE ERROR IN THE BODY-FIXED VELOCITY MEASUREMENT FROM THE COMPUTER VISION (CV), GIVEN IN DEGREES. THE GYRO BIAS CONVERGES AFTER APPROXIMATELY 100 SECOND, HENCE THE RMS VALUES OF THE ATTITUDE AND BIAS IS CONSIDERED FROM 100 SECONDS AFTER START.

		Nonlinear Observer		
		NoCV	**GTOF**	**CEOF**
$\Theta_{b/n}$ (° RMS)	**Roll**	1.1255	0.48963	0.16426
	Pitch	1.2942	0.3906	0.15134
	Yaw	16.1414	4.9359	0.31014
$p^n_{b/n}$ (m RMS)	**North**	9.5084	6.9281	6.9506
	East	4.4114	4.2439	4.2623
	Down	1.2397	0.79603	0.80051
$v^n_{b/n}$(m/s RMS)	**North**	0.63969	0.1526	0.15025
	East	0.32982	0.10317	0.084081
	Down	0.1619	0.15283	0.15257
b^b_{gyro} (°/s RMS)	**Roll**	0.029879	0.018871	0.0054807
	Pitch	0.063454	0.021199	0.0062694
	Yaw	0.095493	0.10302	0.0088266
CV (° RMS)	$\tilde{\chi}$	12.1701	4.7667	0.46633
	$\tilde{\gamma}$	6.8836	2.2066	0.16025

[6] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2011.

[7] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 340–345, 2008.

[8] M. Mammarella, G. Campa, M. L. Fravolini, and M. R. Napolitano, "Comparing Optical Flow Algorithms Using 6-DOF Motion of Real-World Rigid Objects," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1752–1762, Nov. 2012.

[9] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV navigation through indoor corridors using optical flow," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 3361–3368.

[10] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 20–25, 2011.

[11] D. Dusha, W. Boles, and R. Walker, "Attitude Estimation for a Fixed-Wing Aircraft Using Horizon Detection and Optical Flow," *Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*, pp. 485–492, 2007.

[12] S. Weiss, R. Brockers, and L. Matthies, "4DoF drift free navigation using inertial cues and optical flow," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4180–4186, 2013.

[13] J. C. Zufferey and D. Floreano, "Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, pp. 2594–2599, 2005.

[14] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a UAV," *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 302–309, 2005.

[15] P. C. Merrell, D.-J. Lee, and R. W. Beard, "Obstacle avoidance for unmanned air vehicles using optical flow probability distributions," *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 5609, no. 1, pp. 13–22, 2004.

[16] J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous Robots*, vol. 27, no. 3, pp. 189–198, 2009.

[17] F. Ruffier and N. Franceschini, "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 3, no. April, pp. 2339–2346, 2004.

[18] P. C. Merrell, D.-J. Lee, and R. W. Beard, "Statistical analysis of multiple optical flow values for estimation of unmanned aerial vehicle height above ground," *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 5608, pp. 298–305, 2004.

[19] R. Brockers, S. Susca, D. Zhu, and L. Matthies, "Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs," *SPIE Defense, Security, and Sensing*, pp. 83 870Q–83 870Q–10, 2012.

[20] B. Herisse, F. X. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a VTOL Unmanned Aerial Vehicle using optical flow," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 801–806, 2008.

[21] J. J. Kehoe, A. S. Watkins, R. S. Causey, and R. Lind, "State estimation using optical flow from parallax-weighted feature tracking," *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, vol. 8, pp. 5030–5045, 2006.

[22] R. J. D. Moore, S. Thurrowgood, and M. V. Srinivasan, "Vision-only estimation of wind field strength and direction from an aerial platform," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4544–4549, 2012.

[23] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 957–964, 2012.

[24] D. A. Mercado, G. Flores, P. Castillo, J. Escareno, and R. Lozano, "GPS/INS/optic flow data fusion for position and Velocity estimation," *International Conference on Unmanned Aircraft Systems, ICUAS - Conference Proceedings*, pp. 486–491, 2013.

[25] M. Bibuli, M. Caccia, and L. Lapierre, "Path-following algorithms and experiments for an autonomous surface vehicle," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 7, no. 6, pp. 81–86, 2007.

[26] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2643–2648, 2009.

[27] P. Batista, C. Silvestre, and P. Oliveira, "{GES} Attitude Observers - {Part I}: Multiple General Vector Observations," *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 2985–2990, 2011.

[28] ——, “{GES} Attitude Observers - {Part II}: Single Vector Observations,” *Proceedings of the 18th IFAC World Congress*, vol. 18, pp. 2991–2996, 2011.

[29] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi, “Attitude Estimation Using Biased Gyro and Vector Measurements With Time-Varying Reference Vectors,” *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1332–1338, 2012.

[30] J. Guerrero-Castellanos, H. Madrigal-Sastre, S. Durand, L. Torres, and G. Muñoz Hernández, “A Robust Nonlinear Observer for Real-Time Attitude Estimation Using Low-Cost {MEMS} Inertial Sensors,” *Sensors*, vol. 13, no. 11, pp. 15 138–15 158, 2013.

[31] M. D. Hua, G. Ducard, T. Hamel, R. Mahony, and K. Rudin, “Implementation of a Nonlinear Attitude Estimator for Aerial Robotic Vehicles,” *IEEE Transactions on Control System Technology*, vol. 22, no. 1, pp. 201–213, 2014.

[32] R. Mahony, T. Hamel, J. Trumpf, and C. Lageman, “Nonlinear attitude observers on {SO(3)} for complementary and compatible measurements: A theoretical study,” *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 6407–6412, 2009.

[33] R. Mahony, M. Euston, J. Kim, P. Coote1, and T. Hamel, “A non-linear observer for attitude estimation of a fixed-wing unmanned aerial vehicle without {GPS} measurements,” *Transactions of the Institute of Measurement and Control*, vol. 33, no. 6, pp. 699–717, 2011.

[34] S. Salcudean, “A globally convergent angular velocity observer for rigid body motion,” *IEEE Transactions on Automatic Control*, vol. 36, no. 12, pp. 1493–1497, 1991.

[35] J. Thienel and R. M. Sanner, “A coupled nonlinear spacecraft attitude controller and observer with an unknown constant gyro bias and gyro noise,” *IEEE Transactions on Automatic Control*, vol. 48, no. 11, pp. 2011–2015, 2003.

[36] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi, “Globally exponentially stable attitude and gyro bias estimation with application to {GNSS/INS} integration,” *Automatica*, vol. 51, pp. 158–166, 2015.

[37] B.Lucas and T.Kanade, “An iterative image restoration technique with an application to stereo vision,” *Proc. DARPA Image Underst, Workshop*, pp. 121–130, 1981.

[38] B. K. Horn and B. G. Schunck, “Determining optical flow,” pp. 185–203, 1981.

[39] D. Lowe, “Object recognition from local scale-invariant features,” *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999.

[40] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded-Up Robust Features,” in *9th European Conference on Computer Vision*, vol. 110, 2008, pp. 346–359.

[41] D. D. Diel, P. DeBitetto, and S. Teller, “Epipolar Constraints for Vision-Aided Inertial Navigation,” *IEEE Workshop on Application of Computer Vision*, pp. 221–228, 2005.

[42] J. C. Bazin, C. Demonceaux, P. Vasseur, and I. S. Kweon, “Motion estimation by decoupling rotation and translation in catadioptric vision,” *Computer Vision and Image Understanding*, vol. 114, no. 2, pp. 254–273, 2010.

[43] M. Meingast, C. Geyer, and S. Sastry, “Vision based terrain recovery for landing unmanned aerial vehicles,” *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 2, 2004.

[44] M. Sanfourche, J. Delaune, and G. Besnerais, “Perception for UAV: Vision-based navigation and environment modeling,” *Aerospace Lab Journal*, pp. 1–19, 2012.

[45] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[46] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, “An Invitation to 3D Vision,” *Springer*, 2001.

[47] V. Grabe, H. H. Bulthoff, and P. Robuffo Giordano, “Robust optical-flow based self-motion estimation for a quadrotor UAV,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 2153–2159, 2012.

[48] V. Grabe, H. H. Bulthoff, and P. R. Giordano, “On-board velocity estimation and closed-loop control of a quadrotor UAV based on optical flow,” *2012 IEEE International Conference on Robotics and Automation*, pp. 491–497, 2012.

[49] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. The Press Syndicate of the University of Cambridge, 2003.

[50] J. N. Sarvaiya, S. Patnaik, and S. Bombaywala, “Image registration by template matching using normalized cross-correlation,” *International Conference on Advances in Computing, Control and Telecommunication Technologies*, pp. 819–822, 2009.

[51] C. S. Fuh and P. Maragos, “Region-based optical flow estimation,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings CVPR.*, pp. 130–135, 1989.

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

[53] E. Kreyszig, *Advanced Engineering Mathematics*, 2006, vol. 53, no. 386.

[54] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, *Nonlinear and Adaptive Control Design*. New York: Wiley, 1995.

[55] M. D. Shuster and S. D. Oh, “Three-axis attitude determination from vector observations,” *Journal of Guidance, Control and Dynamics*, vol. 4, no. 1, pp. 70–77, 1981.

[56] G. Bradski, “The OpenCV Library,” *Dr Dobbs Journal of Software Tools*, vol. 25, pp. 120–125, 2000.

Appendix A
Parameter Projection

The parameter projection $\mathrm{Proj}(\cdot,\cdot)$ is defined as:

$$\mathrm{Proj}(\hat{b}^b,\tau)=\begin{cases}\left(I-\frac{c(\hat{b}^b)}{\|\hat{b}^b\|^2}\hat{b}^b\hat{b}^{bT}\right)\tau, & \|\hat{b}^b\|\geq L_b,\ \hat{b}^{bT}\tau>0\\ \tau, & \text{otherwise}\end{cases}$$

where $c(\hat{b}^b)=\min\{1,(\|\hat{b}^b\|^2-L_b^2)/(L_{\hat{b}^b}^2-L_{b^b}^2)\}$. This operator is a special case of that from Appendix E of [54].

Appendix B
Camera Simulator

A camera is simulated in order to get measurements of the OF. Here it is our objective to describe how a camera and OF algorithm can be simulated without having access to real images. The objective is to find the displacement of a projected point in the image plane between time t_{k-1} to t_k, that is $dr = r(t_k) - r(t_{k-1})$ and $ds = s(t_k) - s(t_{k-1})$. Lets first consider how one can choose features to project given the UAVs attitude, position and a elevation profile of the terrain. These features are the one that we wish to find the OF of.

At a time t_k a ray is drawn in the camera z-axis as shown in Figure 18. The ray intersects the ground plane at a point $t^n_{\text{centre}} = [x^n_{\text{centre}}, y^n_{\text{centre}}, 0]^T$ or expressed in $\{\text{C}\}$ $\underline{t}^c_{\text{centre}} = (T^n_c)^{-1}\underline{t}^n_{\text{centre}}$, T^n_c being the homogeneous transformation matrix relating $\{\text{C}\}$ and $\{\text{N}\}$. The point t^c_{centre} is named the "centre ground point".

Points are chosen deterministically around the centre ground point, t^c_{centre}, distributed on a plane perpendicular to the ray from $\{\text{C}\}$ to the centre ground point. This plane is named the field of view (FOV) plane. The points are distributed on the FOV plane, ranging from the centre ground point -30 to 30 meters in camera x-direction -40 to 40 meters in camera y-direction, separated with 10 meters in both dimensions. Lets call these points "FOV features" and denote them by p^c_{FOV}. The FOV features in camera coordinates is then defined as $p^c_{\text{FOV}} \in t^c_{\text{centre}} + [x, y, 0]^T, x \in [-30, -20, \ldots, 20, 30], y \in [-40, -30, \ldots, 30, 40]$. Lets now consider only one of the FOV features, and denote this FOV feature p^c_{FOV}.

The FOV feature p^c_{FOV} is transformed to $\{\text{N}\}$ by $\underline{p}^n_{\text{FOV}} = T^n_c \underline{p}^c_{\text{FOV}}$. Let the FOV feature be defined as $p^n_{\text{FOV}} = [x^n_{\text{FOV}}, y^n_{\text{FOV}}, z^n_{\text{FOV}}]$. The FOV feature is then projected onto the terrain by using $x^n_{\text{FOV}}, y^n_{\text{FOV}}$ and the elevation h at the $x^n_{\text{FOV}}, y^n_{\text{FOV}}$ coordinate of the elevation profile. The projected point is then $p^n = [x^n_{\text{FOV}}, y^n_{\text{FOV}}, h]^T$, which is called a "feature".

Now that the feature location in $\{\text{N}\}$ is found, it is in our interest to find the projection of this feature at time t_k and t_{k-1}. The camera moves between t_{k-1} and t_k, meaning the homogeneous transformation matrix T^n_c is time variant. The feature can then be transformed to $\{\text{C}\}$ by $\underline{p}^c(t_k) = (T^n_c)^{-1}(t_k)\underline{p}^n$ and $\underline{p}^c(t_{k-1}) = (T^n_c)^{-1}(t_{k-1})\underline{p}^n$. The points $p^c(t_k), p^c(t_{k-1})$ represents the feature on the surface of the terrain given in camera coordinates at time t_k and t_{k-1} respectively.

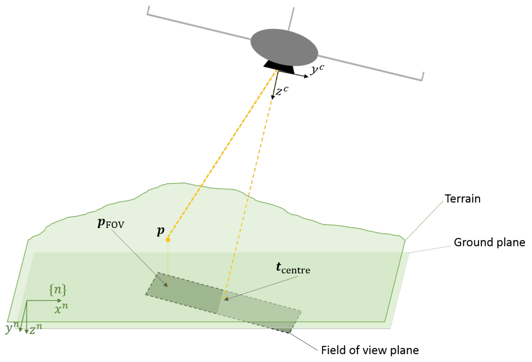


Fig. 18. Features on the surface of the terrain are chosen based on the attitude and position of the UAV. A ray along the camera z-axis intersects the ground plane at t_{centre}. A plane denoted field of view (FOV) is constructed perpendicular to the ray. FOV features are distributed along the FOV plane. Features are constructed with z^n-component from the elevation profile and x^n, y^n coordinate from the corresponding FOV feature. Features p are projected onto the image plane by the pinhole camera model to find the image plane coordinate $x^m = [r, s]^T$. This is done at time t_k and t_{k+1} with the same features, p, to get the discrete OF dr and ds.

The feature at time t_{k-1} and t_k can then be projected onto the image plane by the pinhole camera model from (1), yielding $x^m(t_k) = [r(t_k), s(t_k)]^T$ and $x^m(t_{k-1}) = [r(t_{k-1}), s(t_{k-1})]^T$. The discrete OF can then be found as $dr = r(t_k) - r(t_{k-1})$ and $ds = s(t_k) - s(t_{k-1})$.

Fig. 18 illustrates the relationship between the "centre ground point" t_{centre}, "FOV features" p_{FOV}, and "features" p.

Appendix C

Transformation of Position From NED to Camera Frame Using Matlab Symbolic Toolbox

The following listing shows how the expression for $\mathbf{p}^c$ in (2.23) is found.

Listing C.1: Matlab code showing how $p^c = [x^c, y^c, z^c]$ is related to UAV height and attitude, focal length f and pixel positions s and r.

```
%% Calculate tranformation
clc;
clear;
cn = sym('cn', [3 1]); % Camera position in NED
Theta = sym('Theta', [3 1]);

Rnc = Rbn(Theta)';



Tnc = [Rnc cn; 0 0 0 1];
syms xc yc zc; %Position P in Camera-frame
syms xn yn zn; %Position P in NED-frame




tn_= [xn; yn; zn; 1];
tc_ = inv(Tnc)*tn_;
syms r s;
syms f;

xc = tc_(1); yc = tc_(2); zc=tc_(3);

eqn = [r;s] ==(f/zc)*[yc;-xc]; % Pinhole
[xnT, ynT] = solve(eqn, xn,yn);
tnT= [xnT; ynT; zn; 1];
tcT = inv(Tnc)*tnT;
tcT = simplify(tcT);

pc = tcT(1:3)
```

Bibliography

[1] S. Atev, H. Arumugam, O. Masoud, R. Janardan, and N. Papanikolopoulos. A vision-based approach to collision prediction at traffic intersections. *IEEE Transactions on Intelligent Transportation Systems*, 6(4):416–423, 2005.

[2] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[4] J. Berclaz, F. Fleuret, E. Tretken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[5] B.K.P.Horn and B.G.Schunk. Determining optical flow. *Artif. Intell.*, 17:185–204, 1981.

[6] B.Lucas and T.Kanade. An iterative image restoration technique with an application to stereo vision. *Proc. DARPA Image Underst, Workshop*, pages 121–130, 1981.

[7] G. Bradski. *Dr. Dobb's Journal of Software Tools.*

[8] R. Brown and P. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley and Sons, Inc., 4 edition, 2012.

[9] H. Chao, Y. Gu, and M. Napolitano. A survey of optical flow techniques for robotics navigation applications. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 73(1-4):361–372, 2014.

[10] C.Harris and M.Stephens. A combined corner and edge detector. *Proc. 4th Alvey Vis. Conf.*, pages 147–151, 1988.

[11] Y.-C. Chung and Z. He. Low-complexity and reliable moving objects detection

and tracking for aerial video surveillance with small uavs. pages 2670–2673, 2007.

[12] I. Cohen and G. Medioni. Detecting and tracking moving objects for video surveillance. volume 2, pages 319–325, 1999.

[13] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[14] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. volume I, pages 886–893, 2005.

[16] S. Denman, C. Fookes, and S. Sridharan. Improved simultaneous computation of motion detection and optical flow for object tracking. pages 175–182, 2009.

[17] P. Doherty and P. Rudol. A uav search and rescue scenario with human body detection and geolocalization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4830:1–13, 2007.

[18] G. Farnebäck. Polynomial expansion for orientation and motion estimation. 2002.

[19] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2749:363–370, 2003.

[20] T. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* 2011.

[21] L. Fusini, T. Fossen, and T. Johansen. A uniformly semiglobally exponentially stable nonlinear observer for gnss-and camera-aided inertial navigation. 2014.

[22] H. Grip, T. Fossen, T. Johansen, and A. Saberi. Attitude estimation using biased gyro and vector measurements with time-varying reference vectors. *IEEE Transactions on Automatic Control*, 57(5):1332–1338, 2012.

[23] H. Grip, T. Fossen, T. Johansen, and A. Saberi. Nonlinear observer for gnss-aided inertial navigation with quaternion-based attitude estimation. pages 272–279, 2013.

[24] H. Grip, T. Fossen, T. Johansen, and A. Saberi. Globally exponentially stable attitude and gyro bias estimation with application to gnss/ins integration. *Automatica*, 51:158–166, 2015.

[25] H. Grip, T. Fossen, T. Johansent, and A. Saberi. A nonlinear observer for

integration of gnss and imu measurements with gyro bias estimation. pages 4607–4612, 2012.

[26] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. Von Seelen. Computer vision for driver assistance systems. volume 3364, pages 136–147, 1998.

[27] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[28] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 34(3):334–352, 2004.

[29] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[30] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[31] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[32] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. volume 1, pages I/900–I/903, 2002.

[33] D. Lowe. Object recognition from local scale-invariant features. *Proc. Int. Conf. Computer Vision*, pages 1150–1157, 1999.

[34] Y. Ma, S. Soatto, J. Košecká, and S. Sastry. An Invitation to 3D Vision. *Springer*, 2001.

[35] Y. Mae, Y. Shirai, J. Miura, and Y. Kuno. Object tracking in cluttered background based on optical flow and edges. volume 1, pages 196–200, 1996.

[36] R. Mahony, T. Hamel, and J.-M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[37] M. Mammarella, G. Campa, M. L. Fravolini, and M. R. Napolitano. Comparing optical flow algorithms using 6-dof motion of real-world rigid objects. *IEEE Transactions on systems, Man, and Cybernetics*, 42(6):1752–1762, 2012.

[38] D. Meyer, J. Denzler, and H. Niemann. Model based extraction of articulated objects in image sequences for gait analysis. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 78–81 vol.3, Oct 1997.

[39] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[40] A. Ortiz and N. Neogi. Color optic flow: A computer vision approach for object detection on uavs. 2006.

[41] E. Parrilla, D. Ginestar, J. Hueso, J. Riera, and J. Torregrosa. Handling occlusion in optical flow algorithms for object tracking. *Computers and Mathematics with Applications*, 56(3):733–742, 2008.

[42] R. Prokop and A. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Models and Image Processing*, 54(5):438–460, 1992.

[43] P. Rudol and P. Doherty. Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. 2008.

[44] S. Salcudean. A globally convergent angular velocity observer for rigid body motion. *IEEE Transactions on Automatic Control*, 36(12):1493–1497, 1991.

[45] H. Schneiderman and T. Kanade. Statistical method for 3d object detection applied to faces and cars. volume 1, pages 746–751, 2000.

[46] J. Shi and C. Tomasi. Good features to track. pages 593–600, 1994.

[47] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*, 11(3):204–218, 2005.

[48] M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[49] A. Talukder and L. Matthies. Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. volume 4, pages 3718–3725, 2004.

[50] K. Tieu and P. Viola. Boosting image retrieval. *International Journal of Computer Vision*, 56(1-2):17–36, 2004.

[51] L. Van Gool, G. Szekely, and V. Ferrari. Computer vision, 2011. Written for machine vision course at ETH Zrich.

[52] B. Vik and T. Fossen. A nonlinear observer for gps and ins integration. volume 3, pages 2956–2961, 2001.

[53] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. volume 2, pages 734–741, 2003.

[54] T. Wen, H. Deng, and J. Liu. Recovering ego-motion from optical flow for aerial navigation. volume 8003, 2011.

[55] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[56] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007.

[57] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.

[58] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. Mav navigation through indoor corridors using optical flow. pages 3361–3368, 2010.