Viability assessment of satellite navigation filters for close-proximity operation

Selection, implementation and testing of alternatives to established satellite navigation filters

Noel Weber

Presenting an end-to-end approach for filter performance judgment
Including filter alternative identification, selection, testing, performance assessment and judging against performance criteria

Viability assessment of satellite navigation filters for close-proximity operation

Selection, implementation and testing of alternatives to established satellite navigation filters

by

Noel Weber

in partial fulfilment of the requirements to obtain the degree of
Master of Science in Aerospace Engineering
at the Delft University of Technology,
to be defended publicly on December 17, 2019.

Student number:	4211251	
Project duration:	Jan 1, 2019 – Dec 1, 2019	
Thesis committee:	Dr. A. Cervone,	TU Delft, chairman
	Dr. ir. J. Guo,	TU Delft, supervisor
	Ir. R. Noomen,	TU Delft
	Dr. H. Benninghoff,	DLR

Preface

The project this report is based on was performed as a Master Thesis as part of the MSc Track "Space Engineering" at the Faculty of Aerospace Engineering at TU Delft. The report is aimed at readers with an engineering background in general and at readers with a background in aerospace engineering and control theory in particular. Readers ideally have an interest in visual navigation systems and state estimation problems with real-time application in autonomous satellite applications. More specifically, the report is also aimed at readers working on projects requiring the selection of satellite navigation filters since it can serve as a baseline for how to identify viable filters for different applications.

The author would like to thank Dr. Jian Guo, TU Delft, for supervision and ongoing support during the project and the writing of the report, Heike Benninghoff, DLR, for ongoing supervision and support throughout the research phase of the project, as well as the team at DLR Oberpfaffenhofen for support and advice related to the technical aspects of the project and beyond. In addition, the author is grateful for TU Delft and DLR Oberpfaffenhofen for the use of facilities and research materials, and for the opportunity to undertake the project.

Noel Weber
Delft, November 2019

Summary

As satellite systems become more and more complex and interact with each other in space, close proximity operation becomes an important aspect of many satellite missions. Simultaneously, systems are becoming increasingly autonomous, for example in rendezvous and docking operations. This poses harsh requirements for the guidance, navigation and control systems on-board of satellites, and especially on satellite navigation filters which estimate the state of the satellite and of other systems and objects that it is interacting with, often based on information input from visual sensors. The commonly used Extended Kalman Filter (EKF) performs well but is not necessarily ideally suited for these emerging challenges, which is why the viability of potential filter alternatives in close-proximity satellite operations was studied.

The project was conducted in cooperation with DLR Oberpfaffenhofen in Germany, where currently an EKF is used in close-proximity satellite operations. This filter serves as a baseline for performance testing conducted throughout the project. Potential filter alternatives were identified based on an extensive background study and the mission needs for close-proximity satellite operation. Furthermore, the purpose of the project is to identify and document concrete mismatches between the different testing methods used to serve as a reference in future projects. From the background study two filters, the Extended Kalman Filter with intermediate smoothing step (EKFS) and the Unscented Kalman Filter (UKF) were selected based on a qualitative performance trade-off that focussed on the expected performance under the demands of close-proximity operation in space. The filters were judged based on the available documentation. For the selected filters, as well as for the EKF currently used by DLR, performance criteria were formulated, with a focus on the accuracy of satellite state parameter estimation and filter convergence speed. To collect test data, two approaches were taken: a newly developed simulation test assessing the theoretical performance of the filters in different test scenarios; and a hardware-in-the-loop test in the EPOS 2.0 facility in Oberpfaffenhofen where approaches using two physical satellite models can be performed. The latter test is used to identify problems in the filter performance that have not been found using the simulation test and to validate the filters for more representative real-world performance.

An analysis of the test results from the simulation performance test have shown that the EKFS and the UKF can outperform the EKF in the convergence speed and the estimation of some, but not all satellite state parameters. However, it was also identified that the UKF using its current implementation struggles to assess the attitude of the satellite state accurately. Apart from the attitude estimation from the UKF the filters were considered verified and were implemented in the hardware test facility. The hardware test could not confirm the previously seen performance consistently and both filters showed state estimation divergence at close-proximity of the satellites. Thus, their performance could not be validated and they cannot yet under the current implementation be called viable filter alternatives to the EKF. This is due to the fact that sudden state estimation divergence is potentially catastrophic, especially at close distances of the satellites. In addition, differences in the quality of the measurements between the tests of the different filters highlighted potential problems in the comparability of test results. It was concluded that the UKF is the more promising alternative filter for the future since it showed better performance in the hardware test prior to divergence than the EKF and outperformed the EKFS in all hardware tests. In addition it converged the fastest of the three filters. Further studies need to be performed to correct implementation problems, however. Possible approaches for validating the suggested approaches are presented. Different test approaches should also be taken to make the hardware tests more representative and the simulation test should be updated to be more reflective of real world conditions.

Several observations on the challenges in moving from simulation to hardware testing were identified and are presented. Primarily, challenges were found to arise from the faulty selection of test cases for comparability, the neglect of certain inputs observed in hardware testing with previously unpredicted effects in the simulation test and the continuous change of inputs in the real world which were modelled constant in the simulation test.

Contents

List of Figures

List of Tables

List of Acronyms

Acronym	**Explanation**
AOCS	Attitude and Orbital Control System
AHP	Analytical Hierarchy Process
CCD	Charge Coupled Device
DLR	Deutsches Zentrum fuer Luft und Raumfahrt
ECI	Earth Centred Inertial
EKF	Extended Kalman Filter
EKFS	Extended Kalman Filter with intermediate smoothing step
EnKF	Ensemble Kalman Filter
EPOS	European Proximity Operations Simulator
GEO	Geostationary Equatorial Orbit
GNC	Guidance, Navigation and Control
GPS	Global Positioning System
KF	Kalman Filter
LEO	Low Earth Orbit
MKF	Mixture Kalman Filter
PMD	Photonic Mixer Device
RBPF	Rao-Blackwellization Particle Filter
SLAM	Simultaneous Localisation and Mapping
SVDKF	Single Value Decomposition Kalman Filter
UKF	Unscented Kalman Filter

1 Project introduction and definition

This chapter introduces the research project and the outline of the report. In addition, it gives an overview over visual navigation filtering in satellite applications that is based on the literature study presented in [29] and outlines the project approach.
First, the project introduction and the report outline are given in 1.1. This is followed by a general overview over navigation filtering in space applications in 1.2, describing the need for such systems, different types of approaches and examples of applications of visual navigation systems. Afterwards a breakdown of the different physical and functional aspects of visual navigation systems in satellites is given in 1.3. Finally, the navigation filters that are currently used or could be used for state estimation purposes in these systems are presented in 1.4. Based on this, the technical management and the research questions are defined in 1.6 and 1.7, followed by the research objective in 1.8.

1.1. Introduction to the research project

Since the early days of satellite operations in space, satellite systems have had the need to accurately determine their state with respect to other systems or bodies in orbit in order to interact with them or avoid them and perform the purpose of their mission. Nowadays, systems are increasingly becoming more complex and rely on the operation of multiple satellites with each other in close proximity. In order to do so, many of these systems are equipped with visual sensor systems, such as cameras, to identify other objects early on and be able to react accordingly. However, this also requires a more and more accurate assessment of the state of the other body (target) with respect to the satellite itself. In order to perform this estimation of the state, navigation filters are applied. Navigation filters take sensory inputs and an original estimation of the state of the target as well as its expected dynamic behaviour to predict and update the state estimate and thus allow the satellite to get a better and better understanding of the relative position.
Navigation filters have been developed and improved for many decades since the early days of space exploration. The Kalman Filter (KF) and Extended Kalman Filter (EKF) are well established and well understood filters that deliver good performance under a multitude of circumstances. However, both filters are designed for linear systems or, in the case of the EKF, linearise non-linear systems to perform state estimations. Furthermore, more accurate filters have been developed in the past. Nevertheless, the EKF is still an often applied filter option, for example by DLR in Oberpfaffenhofen. However, as demands for more accurate and faster filters grows due to the more stringent requirements posed by close-proximity satellite operations there is growing reason to determine how viable other filter alternatives are for such operations.

The purpose of this report, which was created as part of a master project conducted in cooperation with TU Delft and DLR Oberpfaffenhofen, is to present the identification, development and testing of potentially viable filter alternatives to an established EKF. After selecting and trading off multiple filter alternatives from a wider pool of satellite navigation filters, a small selection on promising filters is implemented. Afterwards, the filters are tested in an existing GNC structure using a newly designed testing approach, and the subsequent comparative assessment of these filters to determine which is most suited for the requirements in close-proximity, real-world, real-time operation is performed. The filter alternatives are assessed based on their performance with respect to identified criteria relevant for such applications, focussing in particular on the accuracy of the estimate and the timeliness with which the filter can deliver the state estimation. Results from both simulation testing and hardware-in-the-loop testing are taken into account for judging the filter performance. All filters are implemented into a wider GNC framework at DLR Oberpfaffenhofen to allow for testing in the same facilities and using the same filter functionalities. In addition, the research serves to identify and highlight the concrete challenges when moving from a simulation testing stage to a hardware-in-the-loop testing stage. While this may seem straightforward, there are always mismatches between such testing approaches which can greatly affect the filter quality perception, depending on what type of test is used. The outcome can thus be considered a direction for future work to improve the move from theoretical phases and simulation stages to actual implementation and hardware testing.

The structure of the report is presented below. The remainder of this chapter gives a general overview over visual navigation and navigation filters in satellite applications. It presents an outline of the technical management, explaining the research question, objective and focus as well as the methodology for the project. This is followed by an assessment of the existing systems and their required and performed alterations at the start of the project and subsequently identified requirements for the development of further filter options in Chapter 2. Based on the knowledge of the existing system and development specifications, the viable filter alternatives are shown and traded off, from which two filter options for further implementation are identified. The trade off and the technical description of the new filter options is shown in Chapter 3. Chapter 4 and Chapter 5 show the test approach and test results of the filter performance in the simulation performance test and the hardware-in-the-loop test, respectively. The latter also explains the correlation of the test results, and the project results are qualified and interpreted. Furthermore, the research questions are addressed. The report ends with recommendations for future research presented and the presentation of the conclusions from the project in Chapter 6.

1.2. Navigation filtering in satellite applications

A brief overview of the need for visual navigation in satellite systems and the different approaches and applications is given in this section.

1.2.1. The need for visual satellite navigation systems

Visual navigation is used in many different applications for a multitude of reasons, for example for observing the surroundings, building maps or databases of elements in the environment of an object, for assessing the position and attitude (pose) of objects relative to oneself or for determining safe paths through the environment and interacting with other objects.[7][34] Since this report deals with satellite navigation, the object performing the visual navigation tasks and collecting the data will simple be referred to as "the satellite" in the following.

Visual navigation has been used in satellite applications for many years to allow for performing complex missions. Examples include the demonstration of rendezvous and docking capabilities of the Japanese Engineering Test Satellite 7 in 1997 and 1998, the use in orbital docking manoeuvres in GEO and interaction with, and landing on, celestial bodies. Over the years, visual navigation has been established as an important element of operations in space.[25][24][30][27]

Visual navigations are essential for satellite operations for multiple reasons. Satellite operate in an unstructured environment. This means that the environment is not known upfront and there are not necessarily constant points of reference that a satellite can use to orientate itself or place other objects in a frame of reference. This means that a satellite not only needs to identify its environment and its own position in that environment, but it also needs to identify other objects and its position relative to these objects constantly as the surroundings are changing.[11]
Since satellites are in orbit they cannot constantly be accessed or communicated with. Thus, satellites need to be able to operate autonomously for certain periods throughout an orbit to react to changing environments. Visual navigation systems have several advantages for autonomous navigation and space operations in particular.
They are very well suited for unknown environments since they allow to identify the environment without prior knowledge and identify objects to interact with or to avoid. Furthermore, visual navigation systems can track objects across time, which is crucial for systems in space that operate at high relative velocities in order

to avoid potential crashes. Autonomous navigation is also highly reliant on tracking objects and assessing threats since human interference is not possible.[11][18][22]
Another great advantage for satellite operations is the great versatility of the systems. Modern visual navigation systems allow for reacting to different types of scenarios in space, but since they operate using visual sensors they are often also capable to perform other mission relevant operations and tasks, such as collecting imagery for observation on Earth.
Since satellites encounter a multitude of unforeseeable scenarios in space, especially when multiple satellites operate in close-proximity (as is the case for rendezvous and docking or distributed systems), this great versatility of the systems furthermore allows for a wider range of navigation tasks that can be performed. Visual navigation systems are designed for object observation, tracking, monitoring, and can even be used for safe path building through the environment, map building and as a basis for reacting to different requirements.[25][4]

All the aspects considered above highlight the need for well performing visual navigation systems in satellite systems, particularly in autonomous systems. However, the demands for safe operation of satellites and increasingly difficult mission requirements also require research and development in areas related to visual navigation, such as hardware developments in sensors and hardware used for computational tasks (for example reducing system size, power usage and increasing computational capabilities) and developments in software (making systems more efficient, increasing accuracy and adding system capabilities). These developments go hand in hand with changing mission types and environments, which is why different approaches have been developed to best be able to operate satellites autonomously in space using visual navigation systems.

1.2.2. Different approaches to visual satellite navigation

This section explains some of the approaches found in visual navigation systems that are used in satellite applications. They can be used to categorise visual navigation.

Autonomous and non-autonomous visual navigation:
As was mentioned above, satellites can usually not always be reached and interacted with from a ground station on Earth. Thus, a decision needs to be made whether the navigation of the satellite is performed autonomously or non-autonomously. Autonomous systems operate without input from an operator and, more importantly, without data review before the system reacts on incoming information. This requires systems to be trained up to react correctly to incoming information or be able to dynamically learn about changing requirements and adapt to them.
Advances in autonomous visual navigation systems usually require advances in hardware capabilities as well, since such systems are computationally more demanding than others for efficient data assessment. Also, improvements in sensor technologies and image processing are crucial for better navigation. However, autonomous systems can also be faster and more responsive since no data packaging, sending, review and reaction to the data is necessary, and the satellite can thus be safer and more responsive in changing environments.[7][1][3][13]

Relative and absolute navigation:
Visual navigation can, as almost all navigation methods, be categorised as either relative or absolute, or a combination of the two. Relative navigation refers to the process of determining and tracking the position and attitude of a satellite with respect to other objects in the surrounding environment. Since the focus is on the interplay of the objects in the immediate neighbourhood of the satellite, this can help to safely navigate a fast changing environment and be highly reactive. This approach can be combined well with landmark tracking.[7]
While relative navigation only requires a rough knowledge of the position or type of objects around the satellite, absolute navigation is usually based on the knowledge of maps of the surroundings. Here, the satellite position and attitude is determined within a wider reference frame.
Most commonly, an integrated navigation method is used, combining the two approaches. This has several advantages over using the individual techniques. Relative navigation on its own relies on object path estimation and propagation, which is inherently prone to errors. The subsequent error accumulation over time can be counteracted by resetting the satellite and object state estimations using absolute positioning within a reference frame at intervals. This can be done through terrain identification, GPS or celestial navigation. Other approaches that combine multiple sensor inputs and are less dependent on resetting from a wider absolute

navigation approach are for example presented by Andert et al in [2]. Research in these areas is ongoing, aiming to reduce system demands, accumulated error and increase accuracy and computation speed.[28]

Simultaneous Localisation And Mapping (SLAM):
SLAM is a visual navigation approach in which a map of the environment is built. Simultaneously the satellite (or system) is positioned within this map. This approach thus combines relative and absolute navigation as described above. If a map is already fully or partially available this technique can be sped up. Based on available information, a state prediction is performed. Simultaneously, image processing extracts landmark features from collected imagery and feeds it into the system to update the predicted state to a corrected state, which is used as the predicted state in the next system iteration. This approach is often called V-SLAM, or visual SLAM, since it combines the conventional SLAM approach with visual odometry. In visual odometry, the state of a satellite is estimated based on optical flow and feature matching which is required to extract landmark information in dynamic environments.[7][1]

1.2.3. Functional structure

From the discussion above as well as from literature, two aspects that are crucial for visual satellite navigation can be extracted.

Image processing: Image processing is performed to extract information from collected images. This can be done on individual images or a sequence of images, usually referred to as optical flow, in order to extract information on dynamic behaviour in the surroundings of the satellite. The image processing unit usually receives data directly from the sensors, such as cameras and optical depth measurement devices. After the data is processed it is passed to the state estimation system. Image processing can be a highly computationally demanding process and may lead to measurements being passed to the remaining system with a delay.[19][17]

State estimator: State estimators, or navigation filters, perform the state prediction and correction tasks. The state estimation is based on an existing estimate of the system state within an environment (relative or absolute) and the information from the image processing unit. It is passed to the wider GNC system to perform course correction tasks or interact with the environment as required.[4][3]

1.2.4. Different applications for visual satellite navigation

Before expanding on the structure of the navigation system, this section highlights a couple of examples of visual navigation uses in satellite applications.

Applications of visual navigation systems in satellite operation are shaped by the mission and system requirements. While the lack of objects and atmospheric disturbances in space improved image quality, which is great for visual sensor data collection and thus lends itself to visual navigation systems, the inaccessibility of space also demands for highly autonomous and reactive systems. This makes systems more complex, especially since they are often applied in scenarios where multiple objects, other satellites and even formations are visually tracked.

While visual satellite navigation applications have been used in the past, many modern systems benefit from new developments.[11] Maintenance, refuelling, restocking de-orbiting missions greatly benefit from versatile navigation systems and visual observation. The engagement with space debris, which may have unknown attributes, shapes, poses and velocities is another field that deserves more and more attention where visual navigation systems can be applied.

Rendezvous and docking operations have been performed by Mokuno et al. in 1998 and 1999, where the Japanese Engineering Test Satellite 7 performed multiple successful approach and docking operations based on visual navigation. Multiple sensors and cameras were used during the mission.[24]
Since this early success, other developments have taken place. Bennighoff et al. suggest the distribution of the approach into distinct phases, from a far range rendezvous (more than 500 metres distance between satellite and object) where mono cameras are used, to mid and close-range, where more mono or stereo cameras are used. The foundation for this idea is the fact that 3-dimensional feature assessment at a large distance is near impossible since the observed object is a mere dot in the collected image. Thus, there is no need for

stereo cameras. At close operation, Bennighoff et al. suggest the use of an illumination system to highlight 3-D features on the target even further.[4][3]
Moving further out into space, Wen et al. suggest an autonomous R-Bar approach for geostationary satellites based on visual navigation. Their research shows the potential of an approach from a distance of several kilometres and addresses several problems that are encountered when using visual navigation over long distances, such as losing line of sight at minor deviations of sensor orientation.[30]
Even further away from Earth, Polle et al. and Cui et al. present autonomous visual navigation systems that can be used in interplanetary missions, for example for cruise, approach and landing. They focus on the identification of feature points on or around the target celestial bodies which can be referenced with stored maps and images. Such systems can provide accurate pose estimation with simple passive visual sensors.[25][9]

The examples above show that there are multiple use cases and thus multiple challenges for the onboard navigation systems. The project at hand is concerned with in-orbit close proximity satellite operations.

1.3. The onboard navigation structure

Having identified use cases and having discussed general visual navigation systems, this section gives a more detailed overview over the system architecture. A high level breakdown of the visual navigation system is shown in Figure 1.1.
The breakdown of the visual navigation sensor is done according to the different physical aspects of the sys-

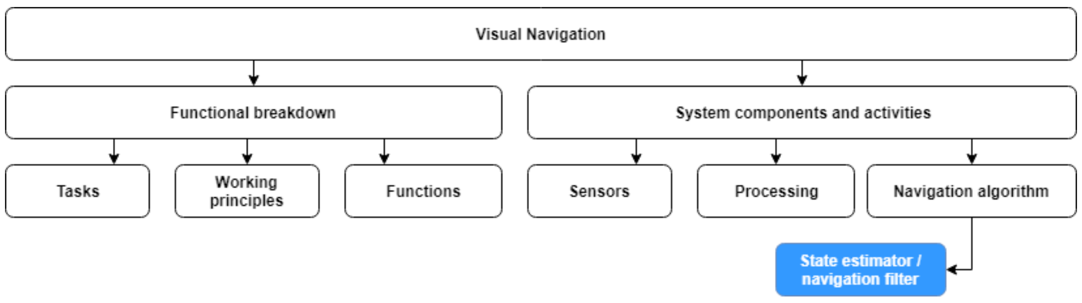


Figure 1.1: Breakdown of components of visual navigation systems (high level)

tems. The functional aspects have been discussed previously.
Each visual navigation system consists of a sensor system which extracts visual information from the environment, an image processing system and a state estimator (or filter).
The sensor system acquires an image, sends it to the image processing unit which processes the information and transfers it to the navigation filter, which returns a state estimate to the other systems. There are some minor alterations to this architecture in some applications, such as the pipeline architecture Boluda presents in [6], where, depending on the scenario, different image processing algorithms can be used. This shows that some of the subsystems can be decoupled.
Figure 1.2 shows an example of a system architecture as presented by Boluda in [6]. Cameras serve as sensors, multiple image processing units can be selected, and the navigation filter is represented via an algorithm result, which is the filter estimation outcome.

Sensors: Sensors in navigation systems are distinguished by either being proprioceptive (measuring internal states) or exteroceptive (measuring the conditions around the spacecraft). For visual navigation systems, the latter are more relevant since they measure the states of objects around the satellite and can thus place the satellite relative to these objects. Proprioceptive sensors may be used to propagate the spacecraft state internally. To evaluate this estimate an outside measurement is, however, always required.
The exteroceptive sensors can again be categorised by being either active or passive. Active systems, such as Laser Detection and Ranging (LADAR), Radio Detection and Ranging (RADAR) and Laser Image Detection and Ranging (LIDAR) actively emit energy and capture the reflections. These systems are heavy and expensive but offer insights on objects in any direction around the satellite. Passive systems, such as monocular cameras and stereo cameras just capture energy that is reflected off of an objects surface. They are simpler and cheaper than active systems. Monocular cameras are well suited for far-range application where the identification of 3-dimensional features is not crucial yet, whereas stereo cameras (or binocular cameras) are better for object feature tracking and are thus applied more in close-proximity operation where detailed

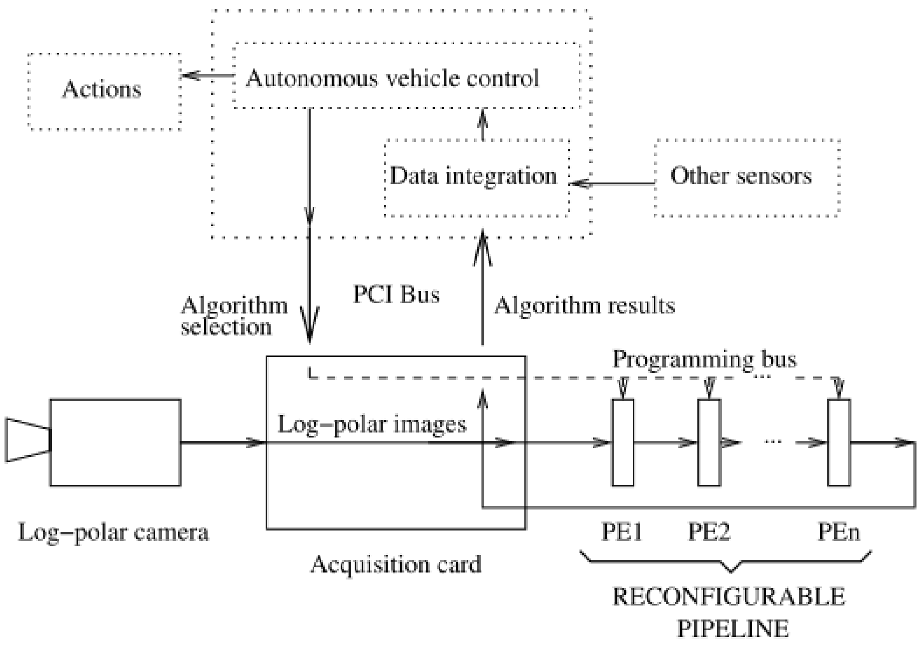


Figure 1.2: Example of a pipeline system architecture presented by Boluda in [6]

object features are crucial to identify. Star trackers are another popular passive sensor type often applied in satellite systems. They are also particularly suited for far range rendezvous scenarios where an object needs to be placed in a reference frame based on its position among celestial bodies in the background.[1][9]

Image processing units: Drawing data from collected images is a functional aspect of the navigation system for which the physical architecture is usually highly purpose built. System platforms and hardware and software requirements are presented by multiple authors.[6][15][8][16][33][32][14][20]
While accurate image processing is an essential part of visual navigation already it is still the focus of much research. The challenge in image processing is often the identification and tracking of features across one or a series of images. Hereby, it can be distinguished between a reconstruction based approach where the entire 3-D environment is recreated, or an active perception based approach, where only information relevant to the current satellite pose is obtained. The latter is computationally less extensive and usually relies on optical flow across multiple collected images, making it suitable for close-proximity satellite operation. The optical flow is translated into 3 directional velocities and 3 rotational velocities.
Since extracting such information from images is difficult and computationally demanding, many researchers have focussed on improving image processing techniques and algorithms. Examples include identifying geometrical features and prioritising them in tracking (see research by Kundur in [19]), modelling the human vision by using low resolution peripheral view and a focussed central view (Jung in [17]) or assessing the difference in image entropy levels to track changes (Wang in [28]).
As can be seen, many different approaches to image processing are available.

State estimator - the navigation filter: Navigation filters are essential to predict and update the state based on a propagation model and incoming processed measurement information. They feed back the best estimate of the current system state of either the satellite itself or a target (depending on what state is assessed by the system) to the remainder of the GNC system. There are different types of navigation filters, from fairly simple linear filters to highly complex non-linear filters with state space sampling steps. In conventional satellites operation, the most prominent filters have long been the Kalman Filter and the Extended Kalman Filter. However, the development is shifting to more purpose selected filters. This is due to growing demands on the filters as mission requirements become more difficult to meet. Especially for close-proximity satellite operations, there are needs for highly accurate and fast updating filters since a state estimation offset or a delay in estimation update could lead to failure and potential loss of the mission. Remaining problems such as propagation errors, faulty measurements and data unavailability need to be accounted for as well.

Different types of satellite navigation filters are discussed in the following.[1]

1.4. Current state and viable alternatives in navigation filtering

Many different filtering techniques for visual navigation systems are available and are being researched. Conducting a more extensive literature study performed in [29] it was found that navigation filters on a high level can be distinguished by being either linear or non-linear estimators. This means that the propagation method uses linear propagation or not, which is essential when estimating non-linear systems. A linear estimator used on a non-linear system accumulates error. Figure 1.3 shows this breakdown. The linear filtering methods have, in space applications, long been dominated by the Kalman Filter, which remains to be one of the most widely used state estimators. Zhe [35] presents the non-linear filters as either finite (where an exact solution can be found, which is assumed to not be the case in the satellite navigation discussed here) or general. Another added category of filters is the linearising filter, which estimates non-linear systems by linearising them and using linear estimators. The most prominent example is the Extended Kalman Filter (EKF), which is applied by DLR Oberpfaffenhofen for state estimation.[4][3]
General non-linear state estimation methods require a numerical approximation method to determine state estimations since no exact solution is found. State estimations are extracted based on their likelihood from a pool of possible states, or state space. Many different approximation methods are possible that result in different filters. Prominent examples are the Unscented Kalman Filter (UKF), the Single Value Decomposition Filter (SVDKF) and numerous different particle filters.

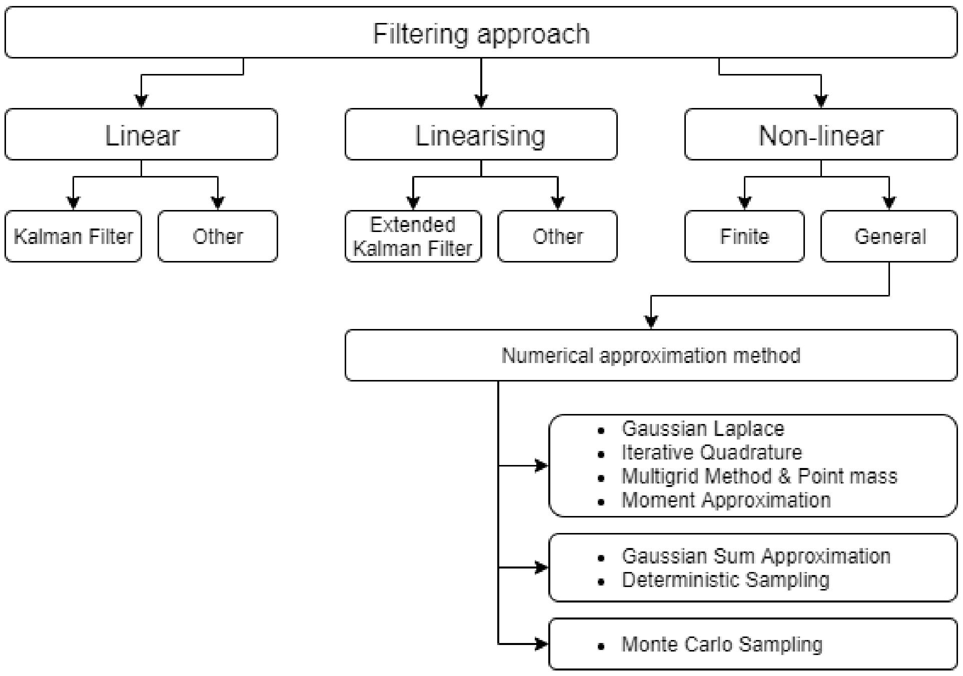


Figure 1.3: Breakdown of different filter options

In space applications, usually non-linear filters are required for higher estimation accuracy. However, this usually comes with higher complexity and thus higher system demands as well as longer computation times. The challenge of identifying and testing different filters for a given scenario is addressed in the research project at hand.

1.5. Research goal

As was seen in the previous section, many different navigation filters are available for visual satellite navigation systems. Not all of them may be suitable for close-proximity operation though, which has strict performance requirements for state estimation and the need for computational resources. The conventional EKF is the most widely established navigation filter for such satellite operations, but, since it is linearising non-linear systems, there may be other filters available that are more suited for close-proximity satellite operation.

The aim of the research project at hand is to identify, implement and test potential filter alternatives that could be more suited to this kind of satellite operation and assess their performance in comparison with the EKF. This aims to collect quantitative performance parameters which allows for a meaningful filter comparison. The filters are tested in simulation tests for their theoretical performance and in a hardware-in-the-loop test where the filters are tested under conditions representing real-world applications more accurately. This allows to address another goal of the research: The identification and highlight of concrete mismatches between simulation and hardware implementation of filters. While this may seem straightforward, the research through hardware testing is important since it highlights issues in filter application that are not captured by simulation testing alone. The research can thus be used by future projects to improve their respective theory application and testing processes.

1.6. Technical management of the research project

In the previous sections the background of navigation filtering in general and for satellite applications in particular was explained. Based on previous developments in this field as well as ongoing research, questions were defines as well as aim and objective for the research project described in this thesis report.
These are presented in the following.

The research questions, as well as aim and objective, were defined in line with the research goal presented in Section 1.5. Thus, they all serve to determine viable alternatives to the Extended Kalman Filter, which has become an industry standard. By presenting new filter options and their respective advantages and disadvantages, insight into potential alternatives or improvements can be gained.

The project benefits from a collaboration of the master student with DLR Oberpfaffenhofen. DLR operates with an existing Extended Kalman Filter and the student is able to implement and trial new filter alternatives in the hardware-in-the-loop facility, and compare the new filters to the real-life existing one. Test data is thus relevant as it is generated in comparison to a working real filter.

1.7. Research questions

The research question is defined as a main question from which two sub-questions are derived, each with its own set of sub-sub-questions. Answering each question aids to structure and perform the research project. The main question for the research project is:

What are 2-3 alternative filtering methods for established navigation filters for visual navigation systems in close proximity space system operations, and how do they perform?

This research question is chosen following the assessment of the state of satellite navigation filtering presented earlier. A focus is set on 2-3 navigation filter methods in order to limit the scope of the project and enable one student to perform the required research within the time frame of a master thesis. The focus to answer this question is set on identifying potential filter alternatives through theoretical performance in simulation as well as implementation of the filters in a hardware in the loop facility and subsequent testing. In order to address these aspects of the project better, the research question is broken down into sub-questions as shown below.

1.7.1. Sub-question 1: Narrowing down filter options

The first sub-question is presented below:

Based on their theoretical performance assessed from simulation, what are 2-3 viable filtering technique alternatives for established navigation filters in visual navigation systems for close proximity space operations?

Based on the working principle and current application of navigation filters in satellite operations, as well as ongoing developments and new filtering approaches, a multitude of potential filter alternatives were identified in preparation for the research project (see [29]). Since fully developing and testing all these filter alternatives is deemed too work intensive for the research project at hand, an initial selection is made. This selection of filters is then fully developed and assessed through simulation to determine their respective performance and to identify whether they can reasonably be trialled in hardware tests. The following sub-sub-questions are posed to aid in answering this first sub-question.

Question 1.1 What are the 5 most promising options for EKF alternatives judging from most recent publications and applications?

Question 1.2 How does the currently used EKF work at DLR according to DLR literature and experts, and why was this navigation filter chosen over others?

Question 1.3 For simulating the performance of navigation filters, what system model should be used?

Question 1.4 What are useful performance parameters that can be used to narrow the selection of alternatives down to 2 - 3 alternative filters?

Question 1.5 What are DLR navigation filter development specs, including expected and required inputs and outputs?

Question 1.6 How do the original individual navigation filter alternatives perform against the chosen performance parameters in the chosen performance simulation?

Question 1.7 Based on their theoretical performance in simulated scenarios, which 2-3 navigation filter alternatives are the most viable for close proximity space operations?

Having determined the simulated performance of the selected filter options, hardware testing can proceed.

1.7.2. Sub-question 2: Determining the quality of filters

The second sub-question is presented below:

Based on performance experiments through implementation and testing in a hardware test-facility, how do the remaining 2-3 navigation filter alternatives perform compared to the established EKF in different mission scenarios, and which are proposed as viable alternatives?

This sub-question is focused on determining the comparable performance of the different filter options with respect to each other and to the baseline filter. Furthermore, the answers to the sub-sub-questions shown below enable a performance assessment in different scenarios. Answering this question and its sub-questions thus also allows to answer the overall research question and to give recommendations on further filter development and application.

The sub-sub-questions are:

Question 2.1 What are the hardware test facilities for satellite navigation filters at DLR, and how are they used?

Question 2.2 What are the performance criteria for navigation filters that need to be tested based on literature review, expert input and mission performance requirements, as well as test possibilities?

Question 2.3 How can the chosen performance criteria be tested in the available test facilities?

Question 2.4 How can the simulated performance be validated and verified through hardware experiments?

Question 2.5 How do the hardware experiments need to be set up, following documentation and experts?

Question 2.6 Based on the mission and literature on navigation filter performance, and expert assessment, what are assessment criteria for comparing navigation filters?

Question 2.7 What are strengths and weaknesses of the navigation filters considered?

Question 2.8 How do the individual tested navigation filters perform compared to the current EKF in the comparison categories that are assessed?

Question 2.9 What applications are the individual navigation filters most suited and least suited for?

At the end of the research project at hand all these questions should be answered.

1.8. Objective and focus of the research project

The objective and the focus of the research project are defined in order to allow for a structured approach to answering the research questions. The research project is scheduled for 10 months, of which 3 months have been spent in preparation as a theoretical study of the subject matter. This report is focussed on the development, implementation and testing phase that is conducted in cooperation with DLR Oberpfaffenhofen in order to answer the research question.

1.8.1. Research objective

The research objective for the research project at hand is to determine the viability of 2-3 possible navigation filter alternatives for close proximity satellite operations to the currently often used Extended Kalman Filter (EKF). Hereby, the EKF currently used by DLR Oberpfaffenhofen serves as a baseline filter for the performance comparison. The assessment of viability is conducted by identifying, designing, implementing, testing and comparing different navigation filters in both simulated performance test environments as well as in a hardware-in-the-loop facility. The judgement of a filters performance is conducted by considering multiple performance parameters.
In addition, the objective is to identify and highlight the concrete differences between simulation and hardware implementation and testing. While the transition from simulation to hardware testing seems straightforward, there are clear mismatches between the two approaches. The necessity for this research arises from the fact that future research projects can benefit from knowing the challenges of such a transition upfront and improve simulation and the assessment of their theoretical performance.

The increased demand for distributed space systems and more and more small satellites has led to a demand in autonomous satellites that can navigate cluttered environments autonomously, since satellite operation through constant operator-control is not practical. Navigation filters are an important element of the autonomous navigation system structure. The research performed in this project is thus aimed at suggesting better filters for such operations, as well as serve as a baseline for further filter development in the future.

1.8.2. Research focus

As was already mentioned, the project is focused on satellite to satellite and satellite to third body operations in close proximity. Hereby, however, the quality of the filter is heavily influenced by the quality of sensory inputs, meaning that a filter will be able to deliver more and more accurate pose estimations if sensors can yield better measurements. The distance to target has a more significant effect on the latter. Nevertheless, the focus of the research is close-proximity operation of satellites using visual navigation systems. Such systems are suited for highly autonomous systems that perform precise tasks. They thus require a high level of accuracy and speed to ensure safe and purposeful operations. Navigation filters should therefore be designed to be fast, efficient, and results need to be easily interpretable by the system computer.

Simultaneously, the research focusses on identifying the challenges in the transition from software to hardware testing. This is a crucial step in any research project that attempts to draw meaningful conclusions on the real-world behaviour of a system from simulation tests, or for research projects that attempt to validate theoretical results in hardware facilities. The transition from a theoretical model to a hardware based testing approach or a real-world application may be considered straightforward. However, since any simulation is a simplification, factors that only occur in real life are neglected in simulation and may affect the performance of a system in an unexpected manor. Thus, an identification of these unobserved mismatches between simulation and hardware testing is important to qualify the quality of the simulation and improve it in future projects, to assess the performance differences between different tests, and to prepare a system better for real-life application before moving into further testing stages.

1.9. Project Methodology

Based on the research questions and objective, this section describes the approach to answer the research questions presented in the previous sections, how the project is set up and how success of filter identification and implementation is assessed in excess of the answering the research questions.

1.9.1. Addressing the research questions and objective

In order to answer the research questions in a satisfactory manor, several steps are taken before work on filters starts:

- **Identify success criteria:** The project is considered a success if all research questions can be answered, since this should allow for both insight into the development, implementation and testing process, as well as for the wider scientific community to benefit from the project at hand. This is why the research questions were determined extensively before the start of the project execution phase.

- **Identify status quo:** The project can only be approached and research questions can only be answered once an extensive understanding of the current status quo is gathered. This primarily includes the satellite navigation filter environment at DLR within which the main part of the development, implementation and testing of filter alternatives takes place. The work performed in the literature study before the start of this part of the project is used as a reference.

- **Determine project structure:** The project structure has been determined in advance as part of the literature study and is presented here again for reference.[29]

- **Determine filter performance criteria:** Since many of the research questions rely on the qualitative and quantitative assessment of different navigation filters a set of performance criteria is defined. This allows for a transparent performance assessment to enable the judgment of the filters.

1.9.2. Key stakeholders and key criteria

Key stakeholders were identified in preparation for this project and it was found that DLR and TU Delft are the key stakeholders that can affect project success. Their key criteria for the project success are investigated here:

- DLR: DLR aims to identify the viability of navigation filter alternatives for its existing Extended Kalman Filter for application in close proximity operation. The criteria is to clearly assess the potential of filter alternatives in close proximity satellite operation and identify how much better or worse the state estimation approach of a filter alternative is. This intent started the project, which is why it is covered by the research questions.

- TU Delft: TU Delft is the university under which this project is performed. TU Delft primarily has scientific and quality expectations for the project. A clear systems engineering project approach should be taken and the scientific reasoning behind design decisions and project decisions should be shown. To address this criterion, a strict systems engineering approach is followed and the project reporting and presentation are conducted following the TU Delft quality guidelines.

The key criteria identified above are both covered by answering the research questions and by following a consistent systems engineering approach. Thus, no extra steps need to be taken that are not covered in this report.

1.9.3. Structure of the research project

The research project contains a literature study and the research itself. The literature study terminated in the report seen in [29]. The document at hand details the research performed to answer the research questions. In order to answer the research questions, a project work flow and breakdown are followed, which can be seen in Figures 1.4 and 1.5.

Project work flow

Breakdown of the project

The research is performed in distinct phases: A filter development stage, a filter testing stage and a filter assessment stage are the decisive steps in the research activities. They are preceded and followed by a research phase and a comparison phase, as shown in Figure 1.5. This section discusses the different steps

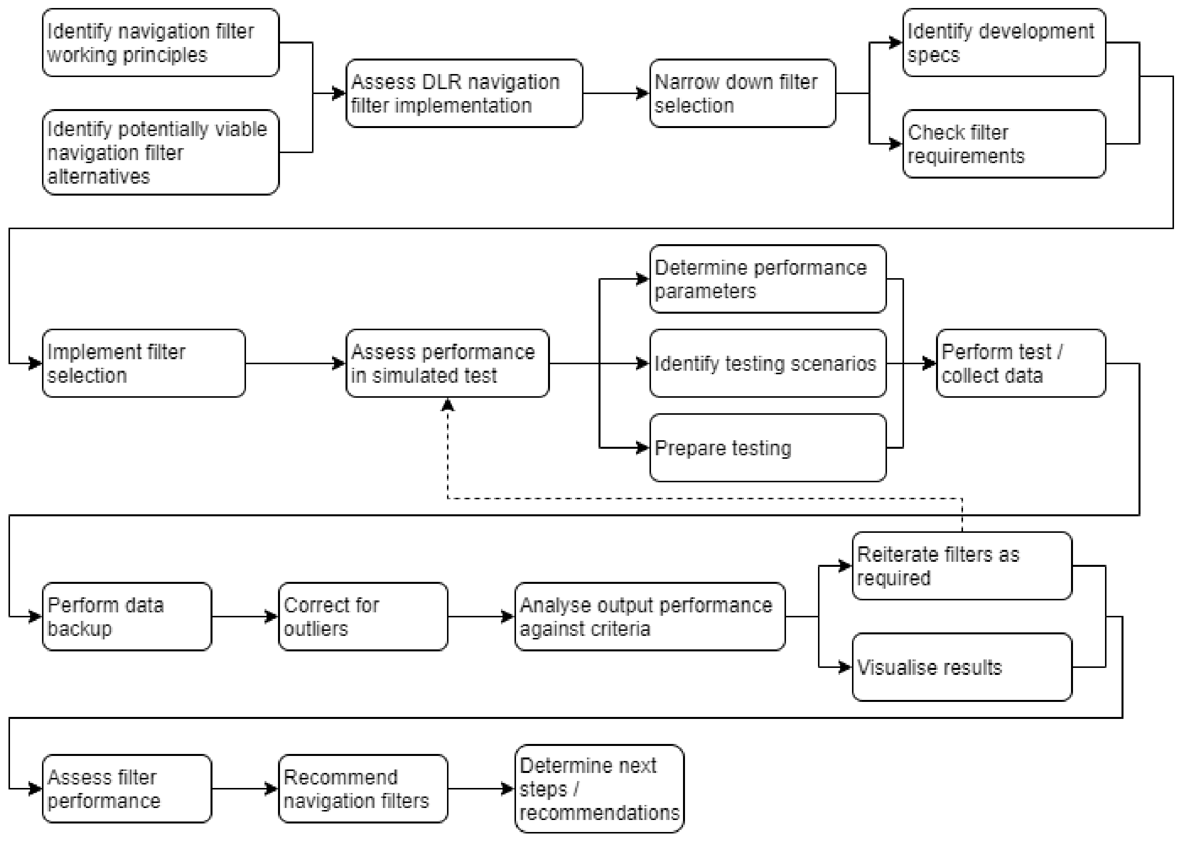

Identify navigation filter working principles
Identify potentially viable navigation filter alternatives
Assess DLR navigation filter implementation
Narrow down filter selection
Identify development specs
Check filter requirements
Implement filter selection
Assess performance in simulated test
Determine performance parameters
Identify testing scenarios
Prepare testing
Perform test / collect data
Perform data backup
Correct for outliers
Analyse output performance against criteria
Reiterate filters as required
Visualise results
Assess filter performance
Recommend navigation filters
Determine next steps / recommendations

Figure 1.4: Project flow diagram

taken in preparation, development, testing, filter assessment and comparison and why the individual steps are taken in the context of answering the research question.

1.9.4. Development stage

The development stage is the phase in filter development where different filter options are selected for coding and are implemented in the DLR navigation filter environment. This is performed for each new filter option, or filter-mode, separately. A filter mode is developed in parallel with its unit test and is thus ensured to be functional with the wider gnc system from the start.

Verification and validation

Verification and validation are crucial in the development of filters. Verification ensures the compliance with requirements, whereas validation shows that the filter accomplishes the intended task and is in fact ready for a mission or a specific operation.

Since the filter development is in the early stages of the development phase, verification is still in the qualification phase. In order to ensure proper filter operation from the early points in the filter development, verification efforts are undertaken early on in the form of system level tests. Unit tests are performed on all functions in a filter, ensuring that the individual components compute the correct results based on given inputs. These unit tests are implemented in the GNC system and are developed and executed whenever the system is compiled and built. Once all the individual functions of a filter are sure to work and compute the correct results, the unit tests are expanded to determine intermediate results of a filter execution correctly, given certain inputs. This includes the check of intermediate results of the correct interaction of *if* statements. Finally, the entire filter iteration functionality is checked given a specific set of inputs. Thus, the code is verified through system level tests and analysis.

However, this only checks the filter internal functionalities and verifies the filter working. Next, validation efforts are performed in the form of mission scenario tests. These are implemented as a simulation perfor-

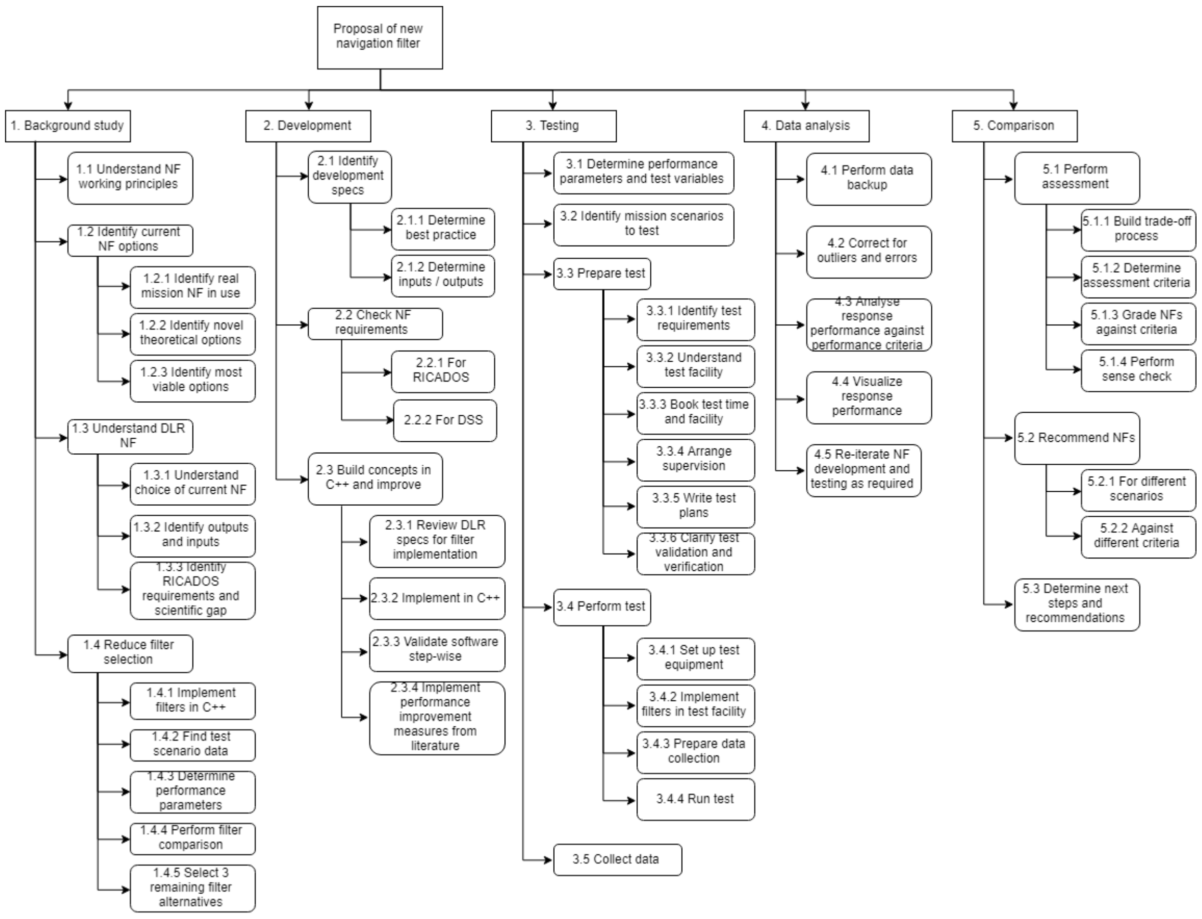

Proposal of new navigation filter
1. Background study
1.1 Understand NF working principles
1.2 Identify current NF options
1.2.1 Identify real mission NF in use
1.2.2 Identify novel theoretical options
1.2.3 Identify most viable options
1.3 Understand DLR NF
1.3.1 Understand choice of current NF
1.3.2 Identify outputs and inputs
1.3.3 Identify RICADOS requirements and scientific gap
1.4 Reduce filter selection
1.4.1 Implement filters in C++
1.4.2 Find test scenario data
1.4.3 Determine performance parameters
1.4.4 Perform filter comparison
1.4.5 Select 3 remaining filter alternatives
2. Development
2.1 Identify development specs
2.1.1 Determine best practice
2.1.2 Determine inputs / outputs
2.2 Check NF requirements
2.2.1 For RICADOS
2.2.2 For DSS
2.3 Build concepts in C++ and improve
2.3.1 Review DLR specs for filter implementation
2.3.2 Implement in C++
2.3.3 Validate software step-wise
2.3.4 Implement performance improvement measures from literature
3. Testing
3.1 Determine performance parameters and test variables
3.2 Identify mission scenarios to test
3.3 Prepare test
3.3.1 Identify test requirements
3.3.2 Understand test facility
3.3.3 Book test time and facility
3.3.4 Arrange supervision
3.3.5 Write test plans
3.3.6 Clarify test validation and verification
3.4 Perform test
3.4.1 Set up test equipment
3.4.2 Implement filters in test facility
3.4.3 Prepare data collection
3.4.4 Run test
3.5 Collect data
4. Data analysis
4.1 Perform data backup
4.2 Correct for outliers and errors
4.3 Analyse response performance against performance criteria
4.4 Visualize response performance
4.5 Re-iterate NF development and testing as required
5. Comparison
5.1 Perform assessment
5.1.1 Build trade-off process
5.1.2 Determine assessment criteria
5.1.3 Grade NFs against criteria
5.1.4 Perform sense check
5.2 Recommend NFs
5.2.1 For different scenarios
5.2.2 Against different criteria
5.3 Determine next steps and recommendations

Figure 1.5: Project flow diagram

mance test, testing the filter state estimation given a simulated set of inputs and scenarios, and a hardware-in-the-loop test, where real sensor information is fed to the filters which in this case are implemented in the wider GNC system.

The purpose of verification and validation efforts is to show the correct working and the fulfilling of the purpose of the product, in this case the filter. As will be discussed later, the purpose of the hardware-in-the-loop test is not to fully confirm the test results of the simulation performance test (both systems operate differently, which is why generating the exact same results would not be possible), but to validate the filter performance in a test closer to mission scenarios.

The verification and validation efforts are described in more detail throughout the report. The simulation and hardware-in-the-loop tests are described in Section 1.9.5, Chapter 4 and Chapter 5. The unit tests that are used in all filters are described in further detail in Section 2.1.8.

1.9.5. Performance testing stage

Once filter modes are developed, they are tested for their performance relative to the baseline filter. This is required in order to be able to answer the research question and achieve the research objective, as it can only be determined through testing whether a new filter mode is a viable alternative to the established system. It is also part of the verification and validation efforts described above. Performance testing is conducted through two means: A simulated performance test, in which an target orbit is simulated through propagation and an erroneous measurement is deducted and fed to the filter; and a hardware-in-the-loop test, in which the navigation filter is fed into the DLR gnc system and is used in tests in the EPOS2.0 facility. In the latter, the filter is fed by real sensor inputs gained from test runs in the facility in real time.

1.9.6. Simulated performance test

Simulation testing for the filter mode performance is developed especially for the purposes of this project. It is developed as an application that is written into the gnc repository which only works with a selected filter mode. It does not require the wider gnc system but only depends on the filter itself. It simulates a target satellite and a servicer satellites in their own respective orbits. Measurements are generated by propagating the orbits and determining the measurements including white noise that are then fed to the filter, which in return determines a state estimate of the target. By comparing the "real" propagated target and the estimated target state the performance of the filter can be determined independently of the remaining gnc system. More detail on this is given in Chapter 4.

Simulated performance tests were chosen as an approach for multiple reasons before performing hardware tests. These can be split into practical reasons, for example the availability of resources, and project critical reasons, such as the ability to test filters parallelly.

- Simulation tests are most importantly used since the input parameters can be easily controlled (which is not possible in hardware-in-the-loop tests to the same extent) and since it is possible to run filters parallelly. This means that filters can be tested under the same conditions and a fair comparison of the output performance is possible. Thus, simulation tests can be used both as a sense check as well as a reliable performance comparison. The hardware tests are thus used more for validation purposes.

- Simulation tests allow for a fast assessment of the output quality of a state estimation from a filter mode: Hardware tests take a lot of time and resources. Thus, it is critical to be certain that filters can deliver results close to the expected accuracy and speed to not either slow down the testing process or potentially damage the hardware. Furthermore, simulation tests can be performed in the development stage already and swiftly, and are thus the preferred means to improve filter performance if the quality of the filter output is not satisfying.

- Simulation tests are resource independent: Simulation tests can be performed easily without having to access testing facilities and can thus be performed at any time. This is particularly important since the hardware-in-the-loop facility is not always ready for testing or is used by different projects. As long as the simulation test delivers representative data it can be used to work on and judge new filters.

- Simulation tests are a cost-free sense check for filter performance: Performing a sense check before implementing new filters in a hardware environment is crucial to ensure the general working of the filter. Diverging state estimates would be unacceptable for a hardware test run.

- The simulation test allows for testing the filter independent of the wider gnc system, allowing for faster and more efficient fault identification than in a full hardware test. This also means that tests can easily be performed without having to first develop a full interface with the wider gnc system, such as telecommands for filter changes.

- Simulation tests allow for a fast prioritisation of which filters to test under which conditions in the hardware facility.

1.9.7. Hardware-in-the-loop facility test

Hardware performance tests are conducted using the EPOS2.0 facility, in which an approach of a servicer satellite to a responsive or non-responsive target can be tested.[4][3]

The servicer features multiple optical sensors that can be used to collect measurements during the approach. Thus, the gnc system is fed with real data that contains real noise, both from sensors and from the actual position of the robots standing in for the satellites. This also means that the system works based on real satellite positions rather than the propagated orbits that are used in the simulation test. This has the advantage that the system is therefore independent of a faulty propagation model when it determines the actual servicer position.

The hardware test is needed to validate the working of the overall gnc system. For the filters they are important since it is the closest testing approach to real mission application in terms of data feed and resulting state estimate output. The advantages over the simulation test are:

- Generally, the hardware test is performed to assess the performance of filters in conditions that cannot easily be modelled or caught in a simulation test. Thus, the test includes more practical elements such as realistic noise, sensory inputs, and in the case of the EPOS 2.0 facility the use of far more complex orbital models.
- The test is performed in real time with real data flows to and from the filter and actually moving satellite models. This truly tests the filter in a real time environment rather than just on simulated orbits. This is also the reason why performing the simulation test first is critical to avoid malfunction and damage due to diverging filter results.
- The test works with actual sensor inputs and actual satellite model positions rather than using the same propagation that is used in the filter to propagate the orbit position. The estimated filter performance may thus be inferior, but also more realistic than the simulation test results.
- The test is used to validate the working of the filter in the wider gnc system.

1.9.8. Performance criteria for assessing filter performance

The performance criteria for comparatively assessing navigation filter performance result from the need for numerically assessing the different filters based on their test performance.
Several aspects are of importance when comparatively assessing the quality of satellite navigation filters. In the early project stage, when no tests have been performed yet but filters are chosen for implementation, a qualitative comparison is needed, whereas later in the project, when filters have been tested in simulation and hardware tests, a quantitative comparison is possible.

The relevant assessment criteria are listed below. They are deduced from decision trees which split the relevant parameters of navigation performance. The criteria are assigned rating factors by using an analytical hierarchy process. The process is performed until a consistency ratio $< 10\%$ is achieved.
Each criterion is graded with a rating factor between 1 (low importance) and 5 (high importance). This can later be used to trade-off the different filters.

1.9.9. Qualitative comparison

The trade-off tree for determining the different qualitative criteria can be seen in Figure 1.6.

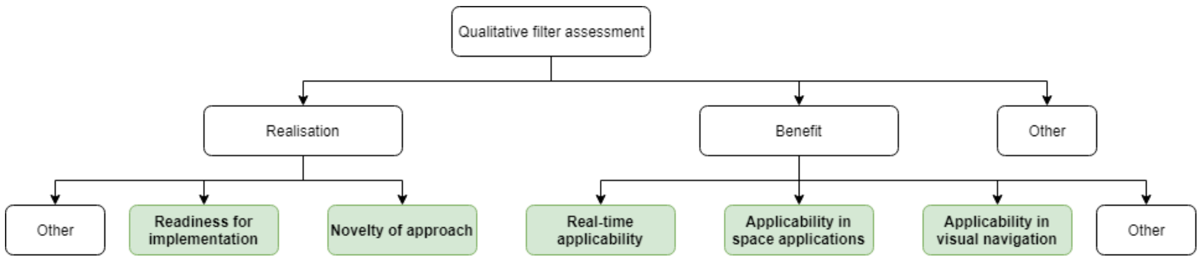

Qualitative filter assessment
Realisation
Benefit
Other
Other
Readiness for implementation
Novelty of approach
Real-time applicability
Applicability in space applications
Applicability in visual navigation
Other

Figure 1.6: Decision tree used to determine qualitative filter trade-off criteria

The two main factors for assessing the filters without actually having numbers available for a rating are the ability to realise the filters as well as the expected benefit from realising the filters. This answers the two questions: can it be built?; and: what is the expected benefit of building it?
To answer the question whether a filter could be implemented, two factors are considered as primarily important. Firstly, it is checked whether the research on a filter is at a stage where the filter can be implemented. If the filter is only in a development stage and not working yet, it would be very difficult to realise. Secondly, the novelty of the approach in navigation systems is assessed. This is assessed through the availability of relevant documentation and a documented track-record of working filters. If this cannot be shown it is assumed more likely that the filter does not work or is very difficult and time-consuming to implement. However, novelty of a concept can also be a benefit since it is a more interesting concept if it has not been shown before.
To assess the expected benefit, it is determined which factors are deciding whether a filter is suited for the intended application in satellite navigation. Three distinct factors can be deduced from stakeholder criteria and from the project objective. The filter should be suited for real-time applicability, for visual navigation and for space applications. Ideally, the filter should have a proven record in some or all of these areas, except

for space application (see below).

The AHP matrix for the qualitative criteria comparison has a consistency ratio is 5.6%. Below, a more detailed description of the criteria can be found.

Readiness to use in real-time application: Rating factor 5
Real-time applicability is key for each considered filter, since satellite navigation systems for close proximity operations need to be highly reactive and operate autonomously. There is a high danger of impact when operating two satellites close to each other, in particular at high relative velocities. Therefore, real-time applicability is crucial for a potential navigation filter alternative to the existing system.
Since satellite navigation filters are only relevant for the project at hand if they can perform in real-time application, this criterion is critical for project success and is thus rated highly.

Ease of implementation: Rating factor 2
In order to allow for the student to implement multiple filters over the course of the projects, filters that are more easily implemented with otherwise similar performance are prioritised over filters that are unnecessarily complex with little added benefit. This allows for an increased focus on test development and test performance, and for the development of further filter options.
While ease of implementation is decisive for the extra work spent in the development phase, it is not overly important for the success and relevance of the project outcome and is thus ranked lower than the other criteria.

Past use in space applications: Rating factor 4
If a filter has a longstanding, proven track record in space applications there is a reduced interest and less added scientific benefit in implementing and testing it, as compared to a potential new filter option that, while promising new benefits, has few applications in existing satellite navigation systems. This criterion ensures that the benefit of the project is of scientific relevance and adds value to the wider scientific body.
Since it is important to work with filters that are not overly exposed to space applications in order to add new insights to the scientific body dealing with satellite navigation, this criterion is ranked with high importance.

Novelty of approach: Rating factor 2
This criterion is an extension of the previous criterion and includes filters that are more recent, interesting developments which have seen little application.
This criterion is not as relevant as the others since a filter that has not been applied in space yet but is otherwise a well understood filter can still be interesting for the scientific community.

Readiness to use in visual navigation systems: Rating factor 4
It is important that all filters work well in a visual navigation system. This can either be shown through past applications of such a filter in visual navigation systems (whether in space applications or not is neglected in this criterion) or through documentation that reliably discusses the potential of the filter in visual navigation systems.
Since the project specifically aims at testing filters in a visual navigation system, this criterion is rated highly.

1.9.10. Quantitative comparison

Once test data has been collected on the estimated state of a satellite resulting from the individual filter options, the performance of the filters needs to be compared numerically. While there is much documentation on the performance assessment of individual filters, it is usually either performed qualitatively or rudimentary based on plotted graphs that show conversion characteristics. However, the goal of the project at hand is to determine clear numerical performance parameters for the individual filters to be able to assign performance values and compare these values. Determining numerical performance parameters for each filter is discussed later in Chapters 4 through **??**. The criteria by which filters are assessed are discussed below.

The breakdown of performance criteria discussed below is visualised in Figure 1.7. To determine the different relevant criteria, the navigation performance in satellite operations is broken down into general performance that would be applicable in any satellite operation, and close proximity operation, which is the focus of this project. General navigation is affected by two main factors: The accuracy of the navigation (how

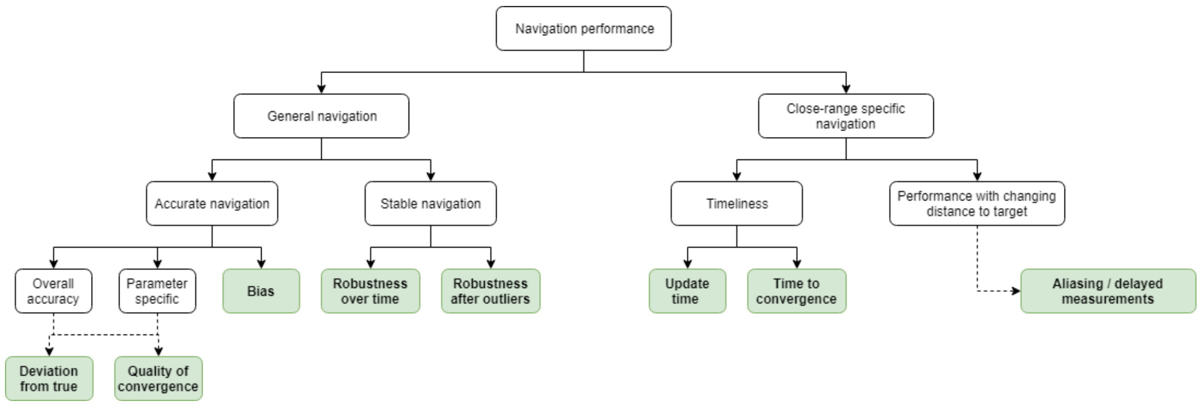


Figure 1.7: Breakdown of quantitative performance criteria

close is a filter estimate to the true state) and the stability of the navigation (how close does an estimate stay to the true state given time and disturbances). The accuracy can be further broken down into overall accuracy (a combination of several parameters) and the parameter specific accuracy, where each estimated parameter is compared to its true parameter equivalent. Both overall and parameter specific accuracy can be assessed through the deviation of the estimate from the true state at any given point in time as well as through the quality of the convergence, meaning how close the estimate gets to a true state over a long period of time when convergence is reached. Furthermore, accuracy can be assessed through observing potential bias in the estimates.
The stability of the navigation is assessed through the robustness over time (how far does an estimate fluctuate away from the true state) and the robustness as a response to outliers.

Close-range specific navigation is characterised by two aspects that are crucial for this type of operation. Firstly, the timeliness of the estimation from the filter is important for the quality of the navigation. Since satellites operate closely to each other, a long estimate delay could cause catastrophic failure, for example through crashes. Thus, the update time of the filter is assessed, meaning how long it takes to determine a new state estimate. Furthermore, the time to convergence is assessed, since in close operations it is crucial that the filter converges as quickly as possible to the true state.
Secondly, the performance of a filter with a change of distance to the target is assessed. This is crucial since incoming measurements, especially in visual navigation, may be distorted with slowly decreasing or increasing distance to target. If a satellite is far from the target satellite it will observe the target as little more than a couple of dots. As the satellite moves closer it will measure more and more detail. However, due to resolution limitations, the perceived edges and details of the target may not update significantly at every measurement step as perceived details move from one pixel to the next in the measured image. Thus, a significant measurement update may only occur every couple of measurements that are incoming. The filter still needs to be able to cope with this effect. This is both a data processing and filter performance issue.

Having determined the filter criteria for quantitative assessment, rating factors are applied using an AHP matrix. The consistency ratio for the matrix below is 5.0%.

Accuracy: Rating factor 5
This criterion assesses how accurately the filter can estimate the true state of the satellite under investigation. It is measured through the deviation of the estimate from the true target.
This criterion is considered the most important since it is critical in close proximity operation.

Time to settle: Rating factor 3
Since the project aims at finding a filter alternative that can quickly converge the state estimate to the true state, it is important to determine how quickly after a starting point the estimate settles. This goes hand in hand with a stabilization of the oscillation of the estimate around the target. A faster settling time is beneficial since in real time application there may be problems if a filter estimates a wrong state for too long.
In close proximity operations, the estimate should settle around the true state quickly to avoid any threat to

the systems. However, it is expected that the estimate will have settled by the time systems are close together, which is why this is not rated higher.

Stability of the estimate: Rating factor 4
Once the filter estimate has converged to the true state it should estimate the true state as accurately as possible throughout the remainder of the operation. The occurrence of extreme outliers as well as extreme fluctuations after convergence would be considered poor filter performance.
Stability is crucial for close-proximity satellite operation since fluctuation or even divergence of the estimated target state could lead to catastrophic failures.

Bias: Rating factor 2
A filter should deviate to the true target state and not have a clear offset or bias in any of the assessed parameters.
Bias is important to recognise and be aware off. If this isn't captured an corrected for it could lead to collisions in close-proximity operation. However, it is expected that this is identified during testing and can be corrected for before a real-life mission.

Computational complexity and requirement of resources: Rating factor 3
Since the filters should be designed for implementation in autonomous satellite systems with limited resources, filters that require fewer resources and compute the state estimate faster are considered superior to filters that take longer to compute an estimate and require more resources.
This is not quite as important as the accuracy of the estimate since it is expected that with increasing system size and improvement of computational capabilities this will be less of a problem. However, designing for systems nowadays, it needs to be considered that resources on board are limited and new filters shouldn't exceed the available capabilities.

1.10. Identifying simulation and hardware testing mismatches

Throughout the project, differences in implementation and approach, as well as expected and unexpected mismatches between the simulation testing approach and the hardware testing approach are documented. Furthermore, differences in test performances are collected from both tests. This information is used to highlight concrete mismatches between the two approaches and identify general means to improve the move from simulation to hardware testing stages in other research projects in the future. Furthermore, this shows what effects and issues are not captured by a simulation test in comparison to more realistic applications.

It is postulated how these effects arise and it is argued based on theoretical knowledge why the presented approaches to counter these effects may work to validate the postulates.

1.11. Conclusions from project methodology

This chapter details the approach that is taken to answer the research questions. As was presented, a filter alternative identification, development, testing and assessment needs to take place to identify the viability of potential alternatives to the EKF in close-proximity satellite operations.

Now that the criteria for the project success and the criteria for the filter performance assessment are clarified, the next step is to assess the current state and working principle of the EKF used at DLR at the start of the project. This is crucial since this existing filter serves as a baseline for both the development of new filters as well as for the comparative assessment. Furthermore, since the newly implemented filters need to work just as well with the wider DLR GNC system as the existing filter does, the inputs, outputs and specific filter functionalities need to be understood and translated into the filter alternatives when they are implemented.

2 Existing system and development requirements

Having discussed the approach to the project in Chapter 1, this chapter describes the working principles principle and implementation of the EKF used by DLR at the beginning of the research project which serves as a baseline for both the development and the assessment of the filters that are newly implemented.
Furthermore, an initial filter restructuring is described that is performed before the development of new filters.
After having assessed the current state of the filter and the required restructuring, general development requirements for further filter development are named in Section 2.4.

2.1. Study of baseline filter

The navigation filter described in this section was developed by DLR. It is an Extended Kalman Filter with several adjustments, for example a functionality that the filter can incorporate delayed measurements, as described by Benninghoff et al. in [4] and [3]. The filter is an industry standard and was chosen since it yields good performance at little computational effort.

2.1.1. Functional breakdown (high level)

On a high level, the navigation filter is part of the GNC system. It is fed by data from the system and feeds back the updated state estimate. This is shown in Figure 2.1, taken from [26], where it is shown that images are collected by LiDAR, CCD camera or PMD camera and are processed. The processed information is fed to the filter, which updates the state and passes it to the guidance system and the controller, which send the control force and the guidance to the attitude and orbital control system (AOCS) of the satellite system. This then relays the current state of the servicer, or chaser, satellite to the filter for the next iteration.

2.1.2. The DLR navigation filter system setup

The filter under consideration is part of the RICADOS project, which features a dedicated GNC system. The navigation filter is a sub-section of the wider gnc system, which, in its entirety, controls the satellite pose and motion.
The gnc system is developed by a team of researchers in a dedicated git repository for easy cooperation. The gnc libraries include image processing, navigation and control. The filter is a subsidiary of the navigation aspect of the project.

The filter and the system dynamic models are part of the navigation. In further detail, the filter directory includes the sensor classes (Bitmasks), state containers and state samplers, the covariance class and the filter itself. In summary, the filter directory includes all functionalities required only for the filter update.

2.1.3. Filter interfaces and communication

In the context of the wider gnc system the filter and its functionalities, such as the covariance class, are connected to the gnc system purely through the filter class. The filter class accepts inputs from the gnc system

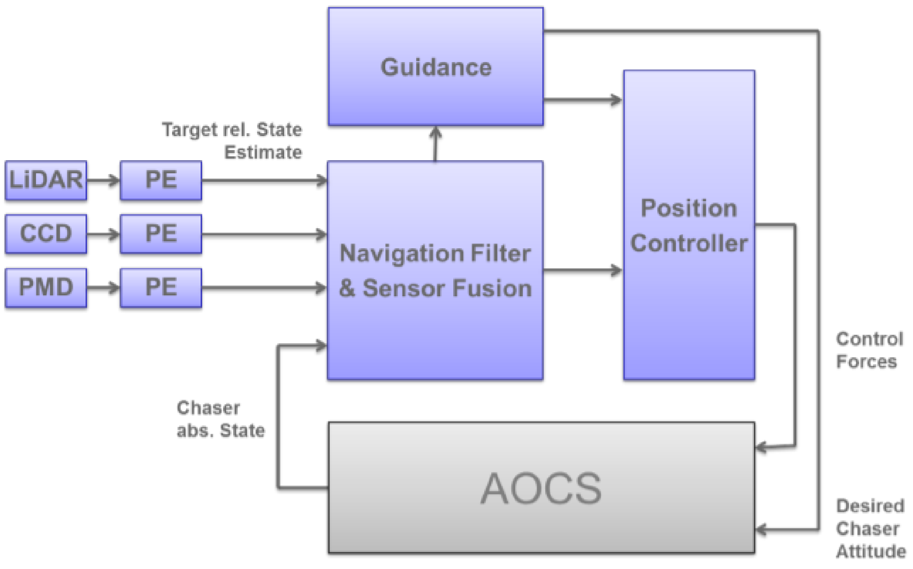


Figure 2.1: High level filter integration in GNC system, Figure from [26]

and outputs a state estimate of the target state. In order to do so, the filter class pulls information and functionalities from the other classes stored in the filter directory.

Inputs

In order to perform a state estimation the filter requires inputs collected by other subsystems. Furthermore, it relies on functionalities stored in external classes as was discussed before. Each of these classes is stored in the filter class as a private member, thus allowing the filter class to access the specific functionalities. This is shown in Figure 2.2.

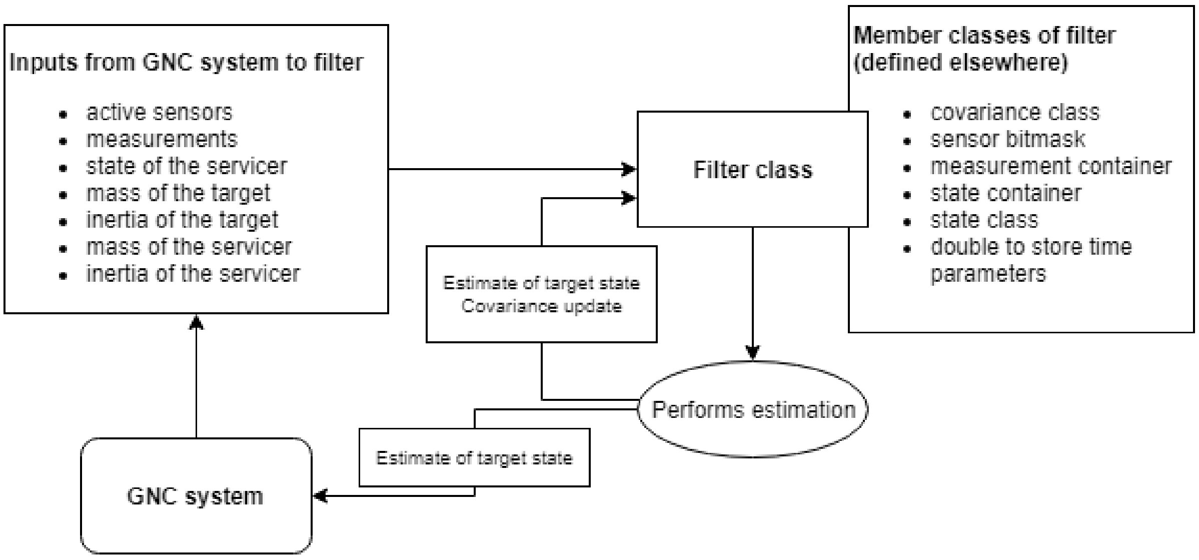


Figure 2.2: High level filter inputs and iteration approach

The functional inputs required for a state estimation update are:

- **activeSensors** This input is a bitmask that defines the currently active sensor system. This could, for example, be a ccd or pmd camera system. In the development stage of the gnc system a sensor dummy is also used to generate sensor inputs. The input allows the filter to adjust to the data quality from different sensors and to adjust the computational process in case of more than one sensor type being active (sensor fusion).

- **measurements** This input is a measurement container that stores the measurements of the latest sensor update. It is fed to the filter for updating the state estimate. The measurement container also includes timestamps for the measurements, which allows the filter to exclude invalid measurements, for example if they were generated too far in the past or if they have timestamps in the future.
- **stateServicer** This input defines the (known) state of the servicer satellite. It is primarily used for updating the filter internal time stamp to the current system time step. The state is given in the ECI frame.
- **massTarget** This input defines the mass of the target satellite. This is required for propagating the estimated state of the target. It is assumed that the mass of the target satellite is known.
- **inertiaTarget** This input is a 3 x 3 matrix that contains the inertia information for the target satellite. Just like the target mass it is required for state propagation.
- **massServicer** This is the mass of the servicer satellite, a known parameters, that is used for the propagation of the state of the servicer. Orbit and attitude are propagated.
- **inertiaServicer** This input is another 3 x 3 matrix that contains inertia information, this time for the servicer satellite. It is used for servicer state propagation.

Outputs

The primary output of the filter is the estimation of the current state of the target satellite. However, the other private member parameters of the filter can be accessed through get-functions, which return the private members outside of the state update loop. This is both required in unit tests and for performance assessment of the filter.
These include:

- Filter covariance matrix
- Filter internal time
- Sensors used by filters in current iteration
- Filter state prediction (in case the state update is not performed)
- Other intermediate results that are used primarily in unit tests

The filter state estimation contains the following parameters:

- Position of the target in x, y and z, given in ECI
- Velocity of the target in x, y and z, given in ECI
- Quaternion in x, y, z and s, used for target attitude determination
- Spin rate of the target in x, y and z

2.1.4. Orbital model

In the following discussion, the propagation methods used in all filters are presented. The orbit model used by DLR is a classic two body model using Kepler orbits for the two simulated satellite orbits, in which only the respective satellite and Earth are included as relevant factors. The influences of other bodies, such as sun and moon, are neglected, meaning that the Kepler orbits are used in its simplified form, the unperturbed Kepler orbits. All orbits are computed and estimated in an ECI coordinate frame.

2.1.5. Baseline filter working principle

The filter used as baseline filter in this project was developed by DLR Oberpfaffenhofen. It is a conventional Extended Kalman Filter as described in [3], with the added benefit that it can take delayed measurements into account and correct for them [4].

The filter is executed by calling the member function *runFilterEstimation()* with the inputs described above. *runFilterEstimation()* was later renamed *computeFilterEstimate()* and made into a state output function rather than a void function to more accurately describe the computation process and directly yield the most important output.

To start the filter, the state of the servicer (S-S) input and the current estimated state of the target (S-E) yielded from the previous iteration are saved in ringbuffers for further use. Then, they are used to determine the filter time *time_* and time step *dt_*. Underscores in the following discussion refer to filter attributes, such as time and time step. The time is taken from the state of the servicer directly, whereas the timestep is the difference between the previous time (stored in the estimation of the target state) and the new time stored in the state of the servicer satellite. Here, the underscore refers to the member of the filter class, meaning that the attribute is stored for future reference. This is important for the iterative nature of the filter, so that updates can be performed based on the relevance of the time stamp.

$$time\_ = time_{S-S} \tag{2.1}$$

$$dt\_ = time_{S-S} - time_{S-E} \tag{2.2}$$

If the time difference between the two iterations is large enough, the previous state of the target, stored in the state estimation, is propagated. Both the orbit and the attitude, both stored in the state class, are propagated.

Orbit propagation: The state includes information on position, velocity and acceleration, each stored in a three dimensional vector. Their propagation is shown in Equations 2.4, 2.5 and 2.7, respectively. Here, the subscript *new* refers to the newly propagated state, *old* refers to the previous state at the previous time. *Pos* is the position part of the state vector, in 3 dimensions, v refers to the velocity, and a is accelerations. The subscript *orbital* refers to conditions in orbit, so for example the acceleration (Equation 2.6) or change in acceleration (Equation 2.8) experienced in orbit by the satellite. F is external force acting on the satellite, and m_{target} is the mass of the target satellite. The acceleration calculation is based on the formulation of gravitational forces experienced by the satellite as shown in Equation 2.3.

$$m_1 \ddot{\vec{r}}_1 = \frac{-G m_1 m_2}{r_1^2} \hat{\vec{r}} \tag{2.3}$$

In the following, dt is the orbit propagation step, which is set by the filter operator (default 0.004 seconds).

$$\vec{Pos}_{new} = \vec{Pos}_{old} + dt * \vec{v}_{old} \tag{2.4}$$

$$\vec{v}_{new} = \vec{v}_{old} + dt * \vec{a}_{orbital} \tag{2.5}$$

$$\vec{a}_{orbital} = -\frac{\mu}{Pos_{old}^3} \vec{Pos}_{old} + \vec{F}/m_{target} \tag{2.6}$$

$$\vec{a}_{new} = \vec{a}_{old} + dt * \dot{\vec{a}}_{orbital} \tag{2.7}$$

$$\dot{\vec{a}}_{orbital} = -\frac{\mu}{Pos_{old}^3} * \left(\vec{v}_{old} - 3 * \frac{\vec{Pos}_{old} \cdot \vec{v}_{old}}{Pos_{old}^2} * \vec{Pos}_{old} \right) \tag{2.8}$$

Attitude propagation: The state also stores the attitude information of a satellite, more specifically the attitude (as a quaternion) and the attitude rate w (as a 3 dimensional vector). The estimated attitude and attitude rate of the target satellite are propagated as shown in [4]. For this, the quaternion differential equation is used, as presented in [31].

Initially, the quaternion is transformed into vector form, denoted as $\vec{q}_{att}$. Furthermore, the old attitude information is used to generate a new derivative of the quaternion, $\dot{q}_{att}$, which is then again turned into vector form.

$$\dot{\vec{q}}_{att} = 0.5 * \Omega \cdot \vec{q}_{att} \tag{2.9}$$

$$\Omega = \left\{ \begin{matrix} 0.0 & -w(1) & -w(2) & -w(3) \\ w(1) & 0.0 & w(3) & -w(2) \\ w(2) & -w(3) & 0.0 & w(1) \\ w(3) & w(2) & -w(1) & 0.0 \end{matrix} \right\} \tag{2.10}$$

$$\vec{q}_{att-new} = \vec{q}_{att-old} + dt * \dot{\vec{q}}_{att} \tag{2.11}$$

This 4 dimensional vector is then transformed back into quaternion form.

The attitude rate is propagated using the old attitude rate $\vec{w}_{old}$, the inertia of the target I_{target} (3 x 3 matrix) and a torque T acting on the target (3 dimensional vector). The torque is assumed 0 by default in all 3 dimensions.
The new attitude rate derivative $dot\vec{w}_{new}$ is determined by solving 2.12.

$$I_{target} * \dot{\vec{w}}_{new} = T - \vec{w}_o ldx(I_{target} * \vec{w}_{old}) \tag{2.12}$$

The new attitude rate w_{new} can then be computed using 2.13.

$$w_{new} = w_{old} + dt * \dot{w}_{new} \tag{2.13}$$

Having found the newly predicted state parameters, position, velocity, acceleration, attitude and attitude rate, as well as the new time for the current iteration from the input state of the servicer, the new predicted state (P-S) of the target satellite is determined.
Next, the covariance for the current iteration is determined, based on the predicted target state, the inertia of the target, and the filters update time step $dt\_$. First, the state dynamics matrix F, a 16 x 16 matrix, is updated. The covariance matrix P is then predicted using 2.14, where Q is the covariance matrix of the state error.

$$P = Q + F * P_{old} * F^T \tag{2.14}$$

Thereafter, the sensor information is used to determine whether the propagation is the only step that can be taken or whether a full filter update is possible. If the update ensues the covariance is fully updated first.
For this, the measurement matrix H is set first. It is a 16 column matrix with a dynamic amount of rows increasing in steps of 7 rows, depending on the amount of active sensors. For 1 sensor active it has 7 rows, for 2 sensors active it has 14 rows. The "basic" form of H shown in Equation 2.15 is repeating for each sensor.

$$H = \left\{ \begin{matrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \end{matrix} \right\} \tag{2.15}$$

Similarly, the covariance matrix of the measurement R is set depending on the amount and type of sensors that are active. Next, the gain matrix K is determined using equations 2.16 through 2.18

$$K = P_{xz} * P_{vv}^{-1} \tag{2.16}$$

$$P_{xz} = P_{predicted} * H^T \tag{2.17}$$

$$P_{vv} = R + H * P_{xz} \tag{2.18}$$

Having updated H and K, the new state covariance matrix can be determined from equation 2.19.

$$P = (Identity_{16x16} - K * H) * P_{predicted} \tag{2.19}$$

Finally, the new state vector is determined for the target satellite. The equation that is used is shown below:

$$x_{new} = x_{predicted} + K * \zeta \tag{2.20}$$

Here, ζ represents the measurement residuum, the parameter that determines the difference between the actual and predicted measurement. The measurement residuum is computed from the measurement in an

ECI frame and the predicted measurement. The latter is determined using 2.21, 2.22 and . The output is a 7 dimensional vector that scales with the amount of sensors being active. This makes sense since The width of K also scales up with the number of sensors.

$$\vec{\zeta}(1-3) = z_{ECI}(1-3) - z_{predicted}(1-3) \tag{2.21}$$

$$\vec{\zeta}(4-7) = \pm z_{ECI}(4-7) - z_{predicted}(4-7) \tag{2.22}$$

To determine the predicted measurement, a 7 dimensional vector is generated. It is extended by its own length by the amount of sensors active (1 sensor active: 7 dimensional vector, 2 sensors active: 14 dimensional vector).
If the measurement has no delay, the measurement prediction is computed from the H-matrix shown in Equation 2.15 and from the predicted target state $x_{predicted}$, if there is a delay in the measurement the state vector of the target at the time of the measurement is used instead. These inputs are used as in 2.23.

$$z_{predicted} = H * x_{predicted} \tag{2.23}$$

Thus, all parameters for a state update have been determined. The iteration can start new, with x_{new} computed in Equation 2.20 being the new estimated state of the target satellite, $x_{estimated}$.

2.1.6. Existing filter code structure

The initial code for the extended Kalman Filter used is not structured into particular sections but rather written as one long loop generating an estimation of the target satellite state based on the incoming inputs at each time step. In between, there are distinct decision points that either continue or stop the current iteration, depending on the inputs and intermediate results. At such points, the current state estimation is given out as an output. These decision points are:

- If the time difference between the previous iteration and the current iteration is less than 0.00001 seconds the previous state estimation is returned
- If the measurements are old, in the future, or insufficient and don't match sensor information the predicted state rather than the updated state estimation is returned

While this code structure works well for one single type of filter, it does not scale well if functionalities are changed or if certain aspects of the filter are expanded into other filter modes. For example, since functionalities of the filter are not split up into separate sections an alteration of code requires a repetition of all test cases in the filter unit test rather than just an update of the affected part of code. This makes the unit tests prone to errors. Additionally, the code is not easily legible and doesn't allow for fast navigation. The original code breakdown structure at the start of the project is shown in Figure 2.3. Black arrows indicate decision points, green boxes indicate primary outputs (if on the left side they come from the previous iteration) and red boxes indicate system inputs from the wider GNC system.

A split into different sub-functions that, combined, make up the functionality of computing a new state estimate mitigates this problem.

2.1.7. Update of filter-internal code structure

In order to make the used filter more easily readable and decouple separate aspects of the filter to facilitate alterations and improve sustainability, the functionality to compute the state estimate of the target satellite (function computeStateEstimate()) is is broken down into three distinct functions.

Preparing the current filter iteration

The filter is prepared for a new iteration by running the command *runFilterPreparation* which takes the current state of the servicer as input and stores it together with the latest state estimation. In addition, the time stamp is extracted from the state of the servicer and compared to the time stored in the latest state estimation. The difference is used as time difference between the current and the previous iteration.

Predicting the state of the target and the covariance

The prediction of the state of the target is performed using the function *runFilterPrediction*, which takes the

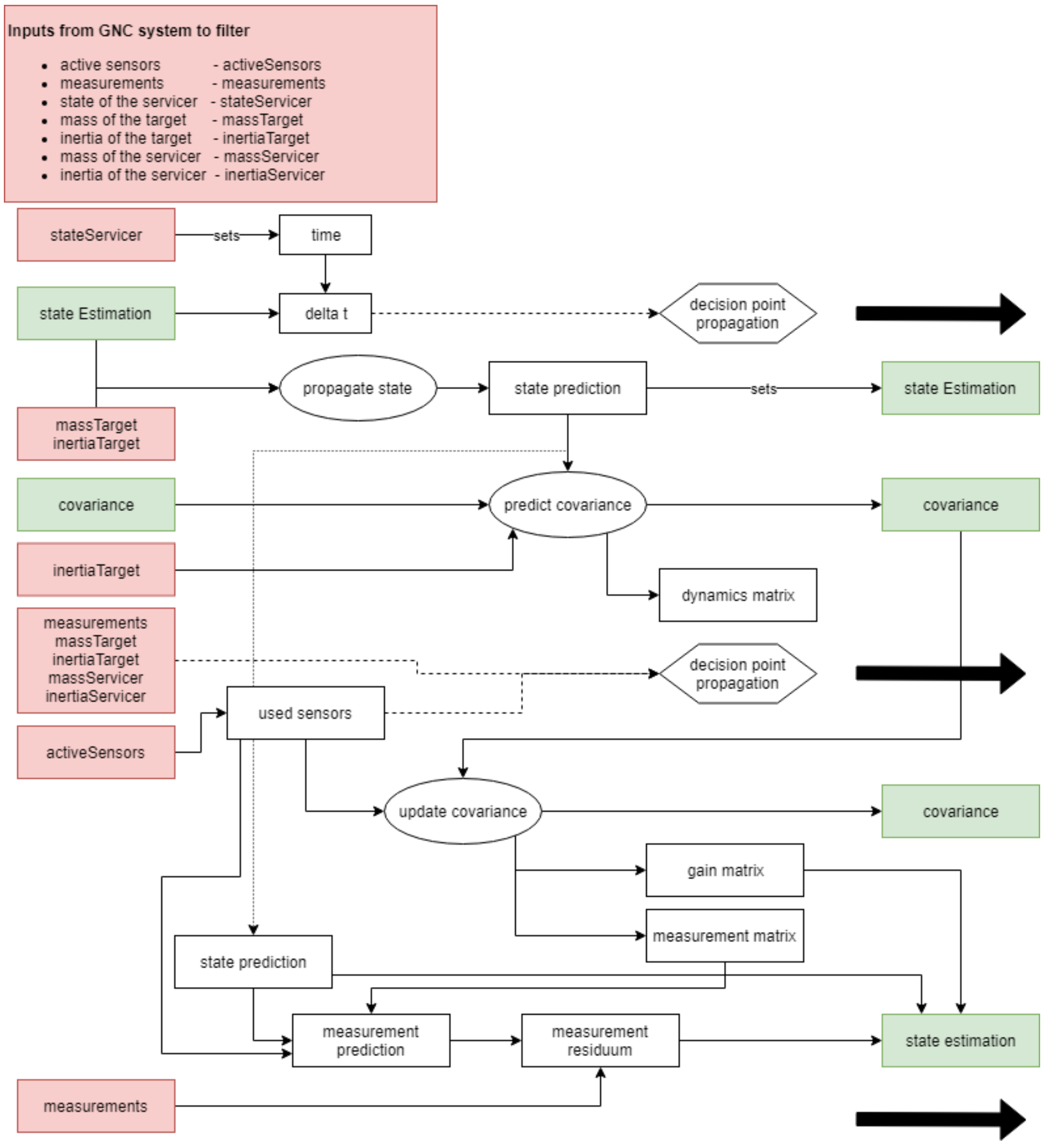

Inputs from GNC system to filter
active sensors - activeSensors
measurements - measurements
state of the servicer - stateServicer
mass of the target - massTarget
inertia of the target - inertiaTarget
mass of the servicer - massServicer
inertia of the servicer - inertiaServicer
stateServicer
sets
time
state Estimation
delta t
decision point propagation
propagate state
state prediction
sets
state Estimation
massTarget inertiaTarget
covariance
predict covariance
covariance
inertiaTarget
dynamics matrix
measurements massTarget inertiaTarget massServicer inertiaServicer
decision point propagation
used sensors
activeSensors
update covariance
covariance
gain matrix
measurement matrix
state prediction
measurement prediction
measurement residuum
state estimation
measurements

Figure 2.3: Original code structure of the existing EKF, difficult to analyse

mass and inertia parameters of the target as input. The class member storing the predicted state is set to the current estimated state. Thereafter, the orbit and the attitude stored in the state is propagated, following the dynamic model of the system, and the covariance is predicted. This results in a predicted state for the target and a predicted covariance. In case of faulty, non matching measurements and sensor data these can be used as outputs for the iteration at hand.

Updating the state estimation

Having found a prediction for the new state estimate, an update based on the measurements needs to be performed. However, this is only done if both the sensor and measurement information is correct. These checks as well as the state update (if inputs are accepted) are performed in function *runFilterPropagation*. First, the sensor information used by the filter is updated to the latest input. Then, based on the measurement and sensor information, it is checked whether the measurements are correctly initialized and can be used. Then it is checked whether the information can be used for state propagation. If not, the predicted state is returned (see above).

If the update through the filter can be executed, the covariance is updated. Thereafter, the measurement is predicted, transferred to ECI and the measurement residuum is computed. The state prediction from the

previous function is then used with the Kalman Gain and the residuum to determine the new state estimation of the target.
The new code breakdown is shown in Figure 2.4. The black arrows indicate potential exit points, depending on the decision points explained above, at which a state estimate can be returned. These are usually *if* statements in the code that are also implemented in all subsequently developed filters to maintain the general structure and purpose of the state estimation update. Furthermore, they are tested in filter unit tests to verify the filter working principle on a system level. The blue boxes indicate parameters that are determined throughout the iteration and may be used in subsequent steps.

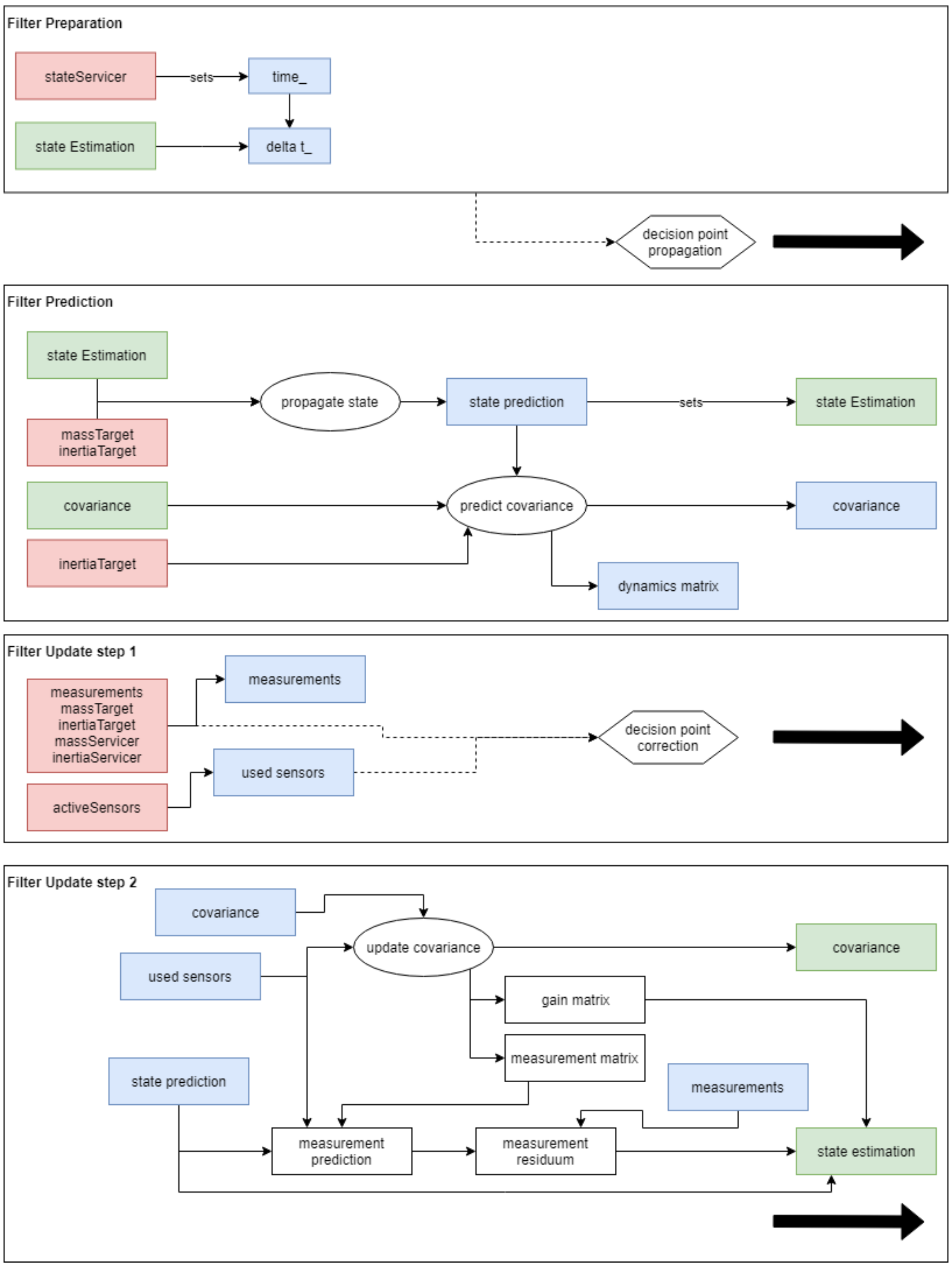


Figure 2.4: Restructure of original EKF

2.1.8. Filter unit tests and quality control

This section describes the efforts taken to show the correct working of the filters and the validation of it performing its intended purpose that are used throughout filter development. All methods presented hereunder are used on the original EKF as well as on all subsequently developed filters.

General filter unit tests

Each aspect in the DLR GNC system, whether application or functionality, has its own dedicated unit test that checks the functions in the header and the cpp file. Hereby, the focus lies on the following points:

- Do all functions accept and return the correct inputs and outputs; this is tested via dedicated functions to set and retrieve the relevant parameters
- Is the overall input and output to the application or functionality correct and does it make sense?
- Are all if statements and causalities tested and assessed for correct throughput, and do operations proceed and stop as expected based on their computational performance?

In order to satisfy these, unit tests are fed with different input parameters to assess the performance of the application or functionality to be tested.

The unit test for the filter is focused on the overall performance of the filter (if it is fed with the correct input, does it return the correct output, if it is fed with faulty data does it stop the estimation process) and the correct working of the filter sub-functions, namely filter preparation, state and covariance prediction and state and covariance correction.
The testing framework that is applied is the BOOST framework, a collection of C++ libraries.

Test compute filter estimate

Initially, the basic_filter class creator is tested by creating a filter without input and checking its default parameters against the expected ones. Thereafter, a test of the set and get functions for the state estimation is performed by first generating a random state and then setting the state of the filter to the random state. Then the filter state is retrieved using the getState() function and the state is compared to the random state that was used for initialisation.
The computeFilterEstimate function of the basic_filter class is the primary function that returns the state estimate. Initially, the filter was simply implemented as one large script and was tested as such, as discussed above, with all tests discussed in the following applied to one large function. After breaking down the filter into separate functions, these could be tested independently, meaning that the test could be rewritten and shortened.

The computeFilterEstimate function executes the functions runFilterPreparation(), runFilterPrediction(), and runFilterCorrection() and checks whether the timestamp on the incoming information is correct as such that the filter should perform a state estimation. Thus, the function only needs to be tested for three distinct cases: The filter propagates the state only and does not perform correction based on the incoming measurements (this is the case in the initialisation of the filter); the filter state estimate remains unchanged due to no time difference between incoming information (this avoids that the filter estimates twice at the same time step); The filter updates the state as required (this is tested using the sensor dummy as data source but could also be performed with other data sources); the impact of the input from different data sources is tested in the test for the filter correction (see below).

All subsequent tests for filter functions are performed by defining two separate basic_filters, filter_1 and filter_2. They are run by letting filter_1 run through the function as it would in the normal filter application, and by running the relevant computations separately and explicitly for filter_2. Thus, by only running the right relevant computations for filter_2 and comparing the outputs of filter_1 and filter_2, for every decision point in a filter function it can be determined whether it performs correctly.

Test filter preparation

The key class member set in the runFilterPreparation() function is the time difference between the time of the new state of the servicer and the time of the state estimation from a previous iteration. Thus, two states are generated, one for the estimate and one for the servicer, and a known time for either is implemented.

The filters are initialised with these inputs. runFilterPreparation() is executed for filter_1. Thereafter, the estimated state of the target is retrieved from filter_2 and compared to the state of the servicer. The time difference is calculated and compared to the retrieved time difference from filter_1. It is crucial that the time differences match since otherwise the filter may propagate the state estimate when it shouldn't.

Test filter prediction

The state prediction and the predicted covariance matrix are the elements of the runFilterPrediction() function which need to be tested. Again, a random state for target and servicer with distinct times (as such that the filter would execute) are generated and used to initialise both filters. Measurements and active sensors are set for the filters, and both filters run the function runFilterPreparation(), to ensure that the starting conditions for the following test match. Next, filter_1 executes runFilterPrediction() predicted target state is retrieved. For comparison, the state estimation of the target is drawn from filter_2 and propagated separately to become the state prediction of the target at that timestep. Thereafter, the states are compared. The covariance can be extracted from filter_1 and compared to the predicted covariance based on the new state prediction (the function covariance.predictCovariance() is already tested in the covariance unit test).

Test filter correction

The function runFilterCorrection() is the most complex filter function and thus requires the most test cases in unit testing. Several distinct cases are performed, for all of which the setup is the same. Just as in the previous test case states are generated and used to initialise the filters. In addition masses and inertias for the target and the servicers are generated, measurements and active sensors are set. Afterwards, each test case goes through the same routine: The time of the servicer state is updated and both filter_1 and filter_2 execute runFilterPreparation() and runFilterPrediction(). Then only filter_1 executes runFilterCorrection(), while the indivdiual steps of runFilterCorrection() are performed separately and independently for filter_2. Afterwards, the resulting state estimations for the target state are compared.

The cases that are tested result from the different possible conditions that can occur. They differ by whether the state is solely propagated or corrected and which filters are used. The test cases are listed below:

- Case 1: Propagation only due to non-existent prior measurement
- Case 2: Update state using sensor dummy: at the second time step, previous measurements are available, so the filter should update (correct) the state estimation in the runFilterCorrection() step
- Case 3: Update state using the CCD camera: It is tested whether the filter can change to a different sensor type based on the given information input on active sensors, and whether the update is performed correctly
- Case 4: Update state using the PMD cam: this is the same approach as in case 3 but for a different sensor type
- Case 5: Update the state using a merger of CCD and PMD camera (sensor fusion): This is similar to the previous tests, but here it is tested whether the filter can correctly deal with both sensor inputs at the same time
- Case 6: no new PMD measurement is supplied, only the CCD measurement should be used for an update of the state estimate: This is tested for by only updating the timestamp for the CCD measurement; it needs to be tested to be sure that even in sensor fusion mode no outdated measurements are used that could disturb the estimation
- Case 7: No new PMD measurement and faulty time stamp on the CCD measurement (in the future), so the state estimate for the target should only be propagated: This is similar to the case 6, with the added twist that now also the CCD measurement is faulty, so the filter should go back to propagation only

These test cases cover all filter functionalities for state estimation. Sub-functionalities such as covariance class specific functions are tested in separate unit tests that were generated before the start of this research project. Covering them all would exceed the scope of this report. The test cases discussed here are applied for all filter modes to be developed as appropriate.

Quality control

Quality control and continuity during the development of new functionalities in the GNC system is ensured through several steps taken by all developers.
Whenever a new branch is developed and compiled it needs to pass not only its own unit tests, but all other unit tests are run in parallel. As such it is avoided that new functionalities or changes to existing code affect other aspects of the GNC system negatively before merging the changes into the main branch. Errors are thus caught early.
In addition, a review process is in place in which each branch is reviewed separately by another developer who was not involved in the writing of the code. This ensures that code is legible and clear, which facilitates understanding code even for new team members since documentation and structure are kept consistent. This also ensures that merger problems, such as conflicting changes, are caught early and by multiple developers.
Furthermore, regular hardware tests in the hardware-in-the-loop test facility EPOS2.0 make sure that the GNC delivers the desired results and performance. The system has the ability to reset to an earlier (working) version in case something does go wrong, so there is always a functioning version at hand for testing is need be.

Since when developing new filter modes it is crucial to be able to retrieve performance figures faster than only when running hardware tests, a new performance test is developed specifically for filters. This is important for the following reasons:

- A filter yielding unrealistic results could pass a unit test but may require an emergency shutoff in the hardware facility if results stand to damage machinery; a simulation test allows for mitigating this risk and testing filter performance in a safe environment
- A simulation performance test can quickly give information on how realistic filter results are and whether results are converging or diverging
- Many different scenarios and input conditions can be tested very quickly, which can help to identify possible problems early

For these reasons it was concluded that in addition to the hardware tests simulation tests would be added to the project.

2.2. Filter alterations: utility functions and filter modes

In order to be able to develop further filter modes and integrate them effectively into the existing gnc system, several alterations are undertaken with regard to the existing filter structure.
At the start of the research project only one filter exists, the currently applied extended Kalman Filter. The filter was developed as a single large C++ application, with a header and cpp file. Since it was deemed not practical to write large filter mode files for the development of further filter modes, it was decided that the existing filter would be split down into filter modes and filter utilities. The filter utilities would include functions used by most or every filter mode and thus serve as a pool of functions that the different filter modes to be developed could draw from as need arises. As such, repeating the same code in every filter mode is avoided and a better overview over the code can be gained. Furthermore, alterations to such filter utility functions only need to be applied in one place, the filter utility function itself, in order to affect all filter modes that utilise them, saving time and avoiding continuity errors. This makes the development of further filter modes considerably easier.

From the existing filter, the following functionalities are outsourced to general filter utility functions:

- Get measurement vector ECI: converts measurements from the servicer body frame to ECI, returns a 7-D measurement vector in the ECI coordinate frame; the function is used in generating the measurement residuum
- Get measurement prediction; computes the measurement prediction; the function is used in the state update step and returns a 7-D measurement vector containing the predicted measurement
- Get single measurement prediction: this function is used as an intermediate helper function in $Getmeasurementpredictio$

- Get measurement residuum: computes the measurement residuum used in the filter update step and returns $meas\_residuum$ stored as a measurement vector

- Get measurement residuum sub: this function is used as an intermediate helper function during the determination of the measurement residuum

- Set used sensor: this function sets the sensors used by the filter based on the active sensor that is passed through from overhead filter class, which in turn gets sensors set via telecommands. It returns a bitmask with the used sensor information. The function is applied in all filters and each filter has its own member function that calls setusedsensors, since it is crucial that the private member of each filtermode which describes the used sensors can be targeted specifically. Thus each filter mode has a setusedsensor function that sets its specific used sensors using the filterutility function.

- Execute filter update: This functions checks whether the sensor inputs are satisfying to perform a full filter update or whether to only propagate the state estimate. This function was outsourced to the filter utilities when more filter modes were implemented in order to avoid repeating unit tests specifically for this function in every single unit test for each filter mode.

Having outsourced the filter functions that are to be used by all filter modes, a sub-directory is opened in the filter directory of the gnc repository. In this sub-directory, the existing EKF is included as a baseline filter (in the form of header and cpp files called basic_filter). As is shown in Figure 2.5, the overarching filter then draws the critical information from the filter mode basic_filter, where all relevant filter calculations are performed. The basic_filter filter mode in turn uses functions saved under the filter_utility directory to perform these calculations. This setup has the following advantages:

- New filter modes can be implemented in the same way as the currently used basic_filter in the filter_mode directory

- New filter modes can all draw from the same pool of filter functions in filter_utilities and thus avoid continuity errors

- Since the overarching, already existing filter file in the filter directory pulls the relevant information from the filter mode it is told to use, the wider interaction of the filter with the gnc system is not affected; the only change is internally contained in the filter and the filter modes

- In order to change the used filter mode, the overarching filter simply needs to be told what filter mode to select and draw data from; this can be done via a simple telecommand or by using a local configuration file that is specified in test runs. Before every test run, the configuration file containing the right filter is selected.

This setup lends itself to add more filters in an easy and fast way, and the selected filter mode can quickly be changed for another one. If it becomes apparent during the development of new filter modes that functionalities are repeating in multiple filters these can be added to the filterutilities directory, avoiding the repetition of code.

2.3. Expanding filter options

This chapter described the setup and functionalities as well as initial changes to the existing EKF used by DLR at the start of the project. The detailed computation steps were explained and the setup within the wider GNC system was outlined.

Having described the filter setup and restructure, the next step is to determine the approach for developing more filter modes that can be viable alternatives for the existing EKF.
In order to allow for a comparison between the different filter modes and to test them using the same testing conditions and environments they are structured similarly and integrated in the DLR GNC system as well. This is explained in the next chapters.

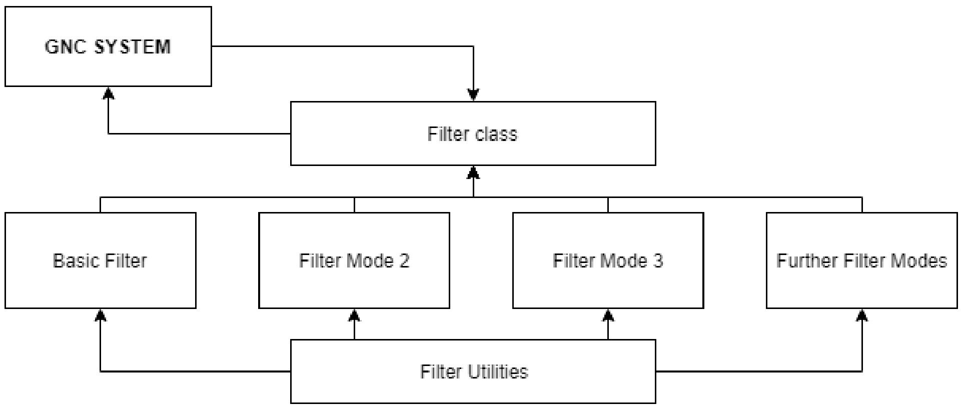


Figure 2.5: Split of filter class to filter modes, the filter class draws information from the relevant filter mode and functions

2.4. Considerations for navigation filter development

Having explained the background to navigation filtering in satellite applications and the existing Extended Kalman Filter that is used as a baseline for this project, this section briefly summarises and expands on the themes that need to be considered for further filter development.

2.4.1. Requirements on interfaces

To be integrated in the wider GNC system and to be successfully tested, all filter modes that are implemented as a potential alternative to the existing EKF should be able to incorporate the same inputs and outputs as the EKF. This means that they estimate a target satellite state that has the same dimension and parameters. Furthermore, all new filters need to process the same measurement information and sensor information, and should be able to output the same intermediate information as the existing filter. This is controlled via the overarching filter class, which, for example, has functionalities extracting and setting parameters such as the active sensor information, covariance and time information for each filter mode. Controlling each filter mode using the same interfaces with the overarching filter class that the existing EKF is already using allows for only adjusting the new filter modes and this overarching class rather than adjusting the wider GNC functions.
A big advantage of this approach is that new filter modes can easily be added and the filter structure can be standardised.

All new filters should have a functionality equivalent to the *computeFilterEstimate* function of the original EKF which only returns the latest state estimate based on the input parameters. The inputs should match the EKF.

Furthermore, an important requirement is to be able to switch between filter modes for testing purposes. To do so, the overarching filter class needs to be provided an input parameter indicating which filter mode to execute (meaning which filter mode computes an estimate). This can be done via configuration files used for GNC initialisation.

2.4.2. Technical requirements

Only few technical requirements can be defined clearly at this point. They are primarily based on the general performance expectations for navigation filters as a whole.

Generally, the state estimation performance of new filter modes should match or exceed the performance of the existing EKF. However, since the new filter modes need to be implemented and tested first, and their performance may be worse (which wouldn't defeat the purpose of the project), the following more general requirement is stated:

- State estimate converges from any realistic initialisation state provided measurements of the target state: This means that the state estimate of any given new filter mode converges to the state of the target satellite. The initial state given to the filter mode as an original estimate is hereby irrelevant.
- The filter should be able to perform a state estimation based on realistic filter inputs: Throughout a

close-proximity satellite operation the inputs passed to the navigation filter by the GNC system vary across time. Especially the measurement information quality will change and include occurrences of outliers, delayed measurements and changing update frequencies. New filter modes should react to these realistic input conditions that could be expected throughout a real mission in space in a way comparable to the EKF, meaning that outliers and delayed measurements do not cause instantaneous filter estimate divergence.

- Filter modes should allow the GNC system to perform autonomously in the same way the EKF allows it to at the start of the project. This means that the same test procedures can be performed as for the EKF using the new filter modes. No additional human interaction should be required that exceeds the current standard to run the filter modes.
- The filter modes should be able to perform a filter iteration step at a frequency of around 10 Hz. This matches the current EKF frequency and should not be exceeded greatly since otherwise a stable performance in the test facility may not be ensured any more.

2.4.3. Development requirements

Throughout the filter development and implementation, several requirements need to be fulfilled. They are presented here.

- The filter modes should be structured similarly (preparation, prediction, update steps) to ensure easy comparability
- All filters need to include progression checks similar to the EKF that check whether, based on the quality of the incoming information, a new filter iteration is performed, and whether this iteration is a full iteration or just contains a state prediction step
- All filter functionalities need to have a dedicated unit test that checks the correct working of the individual function under different input conditions, the same way as described for the original filter in Chapter 2.
- All filter modes need to be included and tested in a simulation performance test and show state estimation convergence from different starting conditions and under realistic input conditions before they are used in hardware tests to ensure safe operation of hardware facilities

3 Development of navigation filter concepts

Having determined the structure and working principle of the existing navigation filter and wider implementation in the GNC system, as well as the resulting requirements on the filter development and filter performance in previous chapters, this chapter describes the potentially viable navigation filter concepts to the existing EKF. First, the viable alternatives found in literature are described, followed by a qualitative comparison of the different options to decide on the best filter concepts for development and implementation with subsequent testing. Then, the selected most promising filter alternatives are developed.

3.1. Viable filter alternatives for currently used filter

As was discussed previously, a wide variety of navigation filters has been developed over the past decades. Many of them are used in real-life applications, some of which have specifically been developed for certain applications, while others are rather theoretical with no real world hardware application. Several potential filter alternatives were chosen for a qualitative comparison as viable alternatives for the EKF in close-proximity satellite operations. They were chosen from a pool of possible alternatives described in [29]

The 6 filters presented in this section were chosen based on several requirements presented below. The filters were selected based on their expected or documented performance in these categories. For the full filter decision process the reader is referred to [29].

Visual navigation filter requirements

Throughout an extensive literature study presented in [29], several reoccurring criteria were found that determine whether a navigation filter performs well. These were presented in Chapter 1.

Criteria for close-proximity space operation

In addition to the normal requirements for visual navigation systems, space environments and specifically applications in close-proximity operation set challenges that need to be overcome. The autonomy of a system is important since systems are difficult to access. This requires navigation filters to operate fast using relatively few resources, since system resources may be limited. Close-proximity operation requires state estimates from visual navigation filters to be highly accurate. This is crucial since high relative velocities between a target and a chaser object could lead to dramatic true state to estimated state offsets if a filter executes too slowly or doesn't provide a high accuracy.
The different mission needs also play a role, as a satellite navigation filter estimating multiple states at once has different requirements than a satellite estimating a single state. For the purpose of this selection process a single target was imagined.

Filters are purely selected based on their documented or expected performance given the challenges and criteria presented above. No quantitative comparison is possible at this point, which is why a multitude of filters from different areas of the design space are chosen. Based on the design space for filtering presented in [29], filter options from both the "linearising" and "nonlinear" branch are chosen. From nonlinear estimation methods several filters are chosen that feature different numerical approximation methods. The chosen filters are shown below.

3.1.1. EKF with smoothing

The conventional EKF uses linearisation to approximate nonlinear systems and uses simple steps for state prediction. The Extended Kalman Filter with smoothing (EKFS) presented by He et al. offers an alternative to the EKF that is similar in its setup, apart from the fact that it uses different Taylor series explosion points and Jacobian matrix calculation points. By using a preliminary state prediction and update, and recalculating these points based on this preliminary update He et al. have shown improved performance over the original EKF in [13].
This filter mode was chosen as it was seen as a viable alternative and promising performance parameters in real-time applications, in particular in space environments. The increased computational complexity should not be extreme enough to limit the application dramatically.

3.1.2. Unscented Kalman Filter

The EKF propagates the mean of a range of possible states. In a non-linear system, the state space may propagate differently than the mean of the state space, thus resulting in a misrepresentation of the potential states by its mean after an iteration step. The Unscented Kalman Filter (UKF) aims to address this issue by propagating multiple possible states, or sigma points, of the state space and thus representing the possible state distribution. The output state estimation, or most likely state, is the weighted mean of this state space after an iteration step.
Choi et al. could show improved accuracy over the EKF using this method in [8]. This method was chosen for its potential performance improvement. However, it remains to be seen whether the increased computational demands exceed the EKF as such that the filter is impractical to apply. Nevertheless, Theil et al. already showed real-time application and implementation in space environments in [27].

3.1.3. Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF) as presented by Evensen in [12] was selected initially since it offers an interesting alternative to the EKF in navigation application. The filter has thus far primarily been used in areas like data management, phase transition tracking and environmental assessment but few examples of navigation applications are known. Nevertheless, Yousif et al. mention this as an opportunity for the filter, which has potential for real-life real-time applications.[34]
The EnKF works in a similar way as the UKF and aims to address the issue of error introduction through linearisation and neglecting of third-order moments encountered in the EKF. It does so by assessing the possible state space represented by state vectors that are collected in ensemble clouds (similar to the UKFs sigma points) but differs from the UKF through the use of heuristically chosen ensemble members.
To make this filter work the distribution is Gaussian and around 100 ensemble members are sufficient to propagate a state. The filter is initialised using randomly chosen conditions and is corrected once measurements are available. Miller et al. have showed the improved performance of the EnKF over the EKF particularly in highly non-linear systems.[23] However, the greatly increased computational effort may be a limiting factor for this filter in autonomous real-time application.

3.1.4. Rao-Blackwellization Particle Filter

Particle Filters, also referred to as bootstrap filters or Monte Carlo filters, use a stochastic sampling approach to estimate state propagation in analytically intractable systems. They are thus highly suitable for non-linear systems. These filters allow for multi-modal tracking and have often been applied in target-tracking, positioning and navigation tasks that are complex and require re-localisation. Particle filters can be understood as a continuation of the UKF and the EnKF. Possible states are drawn directly from the state space at every time step of the iteration with non-Gaussian distributions, which means that more points are required than for the UKF and the EnKF but, with increased sample size, more accurate state estimates are possible.
Applying the particle filter is requires knowing a state vector, its likelihood, the sample size and the system dynamics. Multiple samples are drawn from the state space and their respective weights are allocated based on the available measurement. Iteration commences and particles are selected based on their weights, resulting in a newly unweighted set of particles, which is used for re-sampling to predict a state. A detailed mathematical description of the sampling approach and propagation method is shown in [35].
While some real-life navigation applications could show superior performance especially in convergence

speed and accuracy over EKF and UKF, the Particle Filters are severely limited by their highly extensive computational demands. This makes PFs more suitable for long sampling intervals, which may not be possible in autonomous close-proximity satellite operation. Furthermore, PFs are dependent on the chosen sampling approach. For the PF at hand, the Rao-Blackwellization sampling method was chosen, which has been shown to have reduced computational times compared to other filters and has found some, if limited, application in real-life localisation, map-building and autonomous robot navigation. However, these examples are usually in highly complex environments that requires tracking of a multitude of landmarks, which exceeds the demands of state propagation and estimation.

3.1.5. Mixture Kalman Filter

Mixture Kalman Filters are a combination of RBPFs and EKFs using resampling and rejection methods. They can be adjusted to be applied in non-linear systems, as was presented by Chen and Liu in [20]. This filter can be seen as an extension of the RBPF that has been adjusted more for real-time application which makes it applicable in navigation tasks. However, the complex sampling approach is still computationally demanding. The filter is in very early stages of development and has not found wide-spread application yet. Nevertheless, it is a promising alternative for the EKF in navigation tasks, in particular in larger systems with more resources.

3.2. Filter selection for further study

In order to be able to implement and test filters in the existing GNC system within the given time for the project, the filter selection shown above is narrowed down further. This is done since it is decided that only 2 - 3 filters are realistic to be implemented and tested properly within the given time, and will still allow for answering the research question. The original plan to implement and test all potential filter alternatives had to be changed since it was not deemed realistic to perform these tasks within the given time frame. The selection criteria for narrowing down the selection are qualitative since no consistent test data is available yet that would allow the quantitative comparison of each filter. The same criteria that were shown in chapter 1 are used. The rating of each filter in each category based on the available documentation is shown below.

3.2.1. Qualitative filter rating

The filter rating by category is shown and reasoned below. Filters are rated from 1 (poor expected performance) to 10 (outstanding expected performance).

Readiness to use in real-time application

EKFS: Rating 8: The EKFS can show proven examples in real-time application and is expected to have computational requirements that do not exceed the EKF by much.

UKF: Rating 6; Some examples of real-time applications are available. This filter is expected to perform better than the EnKF and the RBPF since these require the processing of more state space representation points. However, the computation time is expected to be longer than for the EKFS.

EnKF: Rating 4; A large amount of state space members need to be tracked, causing increased computational effort which could be a problem for real-time application. Not many examples of applications in real-time real-world autonomous systems are available.

RBPF: Rating 5; This filter has already been used for real-world and real-time application, however, not under as stringent requirements as are present in close-proximity satellite operation. The filter is expected to perform better than other particle filters but not as well as EKFS and UKF due to a highly increased number of data points to track and propagate.

MKF: Rating 8; The MKF was specifically designed for real time application and is thus expected to perform well. However, little real-time real-world application could be observed yet.

Ease of implementation

EKFS: Rating 7; Since the EKFS implementation is expected to essentially require an adjustment to the already existing EKF through the calculation of an intermediate smoothing state and the new calculation and explosion points of jacobian matrix and Taylor series expansion it is expected to be easy compared to the other filters.

UKF: Rating 6; The UKF is well documented and has found numerous applications. It is slightly more complex than the original EKF since it requires the determination of the sigma points and their individual propagation.

EnKF: Rating 5; The EnKF is similar to the UKF but has found less use cases and real-world applications are thus less documented. The Gaussian sampling from the state space should be comparable in implementation effort as the UKF.

RBPF: Rating 5; The same reasoning applies for the RBPF as for the EnKF.

MKF: Rating 4; The MKF has found very little real-world application and is thus expected to be the most tricky to implement. Furthermore, the parallel implementation of multiple Kalman Filters is expected to provide additional difficulty.

Use in space applications

EKFS: Rating 8; The EKFS has a proven track-record of applications in space systems. It runs fast and provides accurate results and is thus suitable for navigation filtering ins satellite systems. Since the filter requires little resource in addition to the EKF the filter is expected to work well in autonomous systems as well.

UKF: Rating 5; Theil et al. have already hinted at implementation in space applications in autonomous systems ([27]) but further applications need to be seen. It is more complex than the EKF and EKFS and may thus not work quite as well.

EnKF: Rating 2; No shown track record in space applications and only little real-world and real-time applications make the EnKF an uncertain filter for space applications. Furthermore, the high computational complexity may require a large systems and hinder autonomy.

RBPF: Rating 3; While the filter is similarly complex as the EnKF it has a couple of real applications and has been used in autonomous systems. Still it may not be suitable for small system autonomy and the requirements for close-proximity operation.

MKF: Rating 1; There is no substantial track record of real-world application and the applicability for space applications cannot be assessed. It is expected that the high computational complexity (despite being designed for real-time application) would counteract an application in an autonomous space system.

Novelty of approach

EKFS: Rating: 3; the EKFS has been applied in a few systems and its application, while new for the purpose at hand, is not novel in itself or even the field of satellite operation.

UKF: Rating 5; While the UKF has been applied in a few systems and the application in space systems has been discussed the approach is not as common for real-world real-time space applications as for example the EKF and EKFS.

EnKF: Rating 7; The EnKF would be a novel approach in the area of satellite operation and real-time visual navigation. Little other documentation is available.

RBPF: Rating 6; The same reasoning as for the EnKF applies to the RBPF. However, this filter has found application in more examples found in documentation and is thus ranked slightly lower.

Table 3.1: Assessment of filter options by qualitative scoring

Criterion	EKFS	UKF	EnKF	RBPF	MKF
Real time use [5]	8	6	4	5	8
Implementation [2]	7	6	5	5	4
Use in space [4]	8	5	2	3	1
Novelty [2]	3	5	7	6	8
Visual systems [4]	5	7	6	8	5
Final rating	112	100	76	91	88

MKF: Rating 8; The MKF would certainly be a novel filter approach in the field of autonomous satellite operation. It has been envisioned for use cases like the one at hand, but the fact that it is still in early stages give it an interesting advantage over the other filters.

Accuracy in visual navigation system

All filters can be, and have been, used in visual navigation. Differences arise primarily from computational complexity and accuracy of estimation, since these are the deciding criteria in close-proximity visual navigation of satellite systems.

EKFS: Rating: 5; The EKFS has shortened computational times compared to the other filters and is expected to outperform the EKF in accuracy of estimation. While the latter aspect may not be quite as good as for the other filters it is ranked highly for the expected combination of speed and accuracy.

UKF: Rating 7; The UKF has found application in visual navigation systems and delivers very accurate results, more so than the EKFS, but it features increased computational complexity.

EnKF: Rating 6; Similar to the UKF, the EnKF is expected to deliver very accurate results, but at far longer computation times than the UKF.

RBPF: Rating 8; The RBPF would be expected to deliver the most accurate state estimation if given the longest computation times. Since the project at hand requires a balance between speed and accuracy though the RBPF is rated with an 8 instead of a 10.

MKF: Rating 5; Little can be said about the application of the MKF in visual navigation systems. From the documentation it is expected that it performs well.

3.2.2. Qualitative filter comparison and trade-off

The criteria rating of the different filter options is summarised in Table 3.1.

The trade-off shows that the EKFS and the UKF are the two filter options that are most suitable for implementation and testing given the project scope and requirements. This is not surprising since they have both been used in other systems and are thus well documented. They are expected to deliver results superior to the EKF with the least added computational complexity and are thus suitable for real-time application. As a third option, the RBPF could be implemented which would be a more novel and certainly more complex approach which would beat the other filter options through expected improvements in state estimation accuracy.

This trade-off was repeated by using an Analytical Hierarchy approach to cover multiple selection methods. The results are shown in Table 3.2. While the order of filter outcome is the same it can be seen that the UKF and the RBPF are closer together.

Two trade-off methods have shown that the best filters to start implementing and testing are the Extended Kalman Filter with Smoothing step and the Unscented Kalman Filter.

Table 3.2: Assessment of filter options using analytical hierarchy process

Criterion	EKFS	UKF	EnKF	RBPF	MKF	AHP Consistency
Real time use [5]	41.2	15.2	7.7	10.2	25.6	2.1%
Implementation [2]	43.8	27	11.1	11.1	7.1	1.6%
Use in space [4]	47.6	20.9	10.5	14	7	2.4
Novelty [2]	6.2	11.5	29.6	21.9	30.8	5 %
Visual systems [4]	8.9	22.2	19.5	35.5	13.9	2.8%
Final rating	532	325.4	239.9	315	287.4	NA

3.3. Development of viable filter alternatives

Having selected new filter modes for implementation and testing, this chapter explains the working principle, mathematical and functional breakdown of the new filter modes.
First, the Extended Kalman Filter with smoothing step is explained in 3.4, where the structure and calculation for a filter iteration are outlined. The same is then presented for the Unscented Kalman Filter in 3.5.

3.4. Extended Kalman Filter with smoothing

The Extended Kalman Filter with smoothing was implemented following the documentation presented by He et al. in [13].
The conventional EKF, as is also applied by DLR Oberpfaffenhofen, is a suitable and well-understood filtering technique for linear and simple non-linear systems. The filter works similarly to the Kalman Filter by can deal with non-linear systems through a linearisation approach. However, due to this linearisation, estimate errors can accumulate the more non-linear a system becomes. Furthermore, the uses the same point (or state) as calculation point of the Jacobian matrix and as the explosion point for the Taylor series. Thus, propagated and updated state estimates carry on errors from the approximates of these points. The EKFS tackles these problems by introducing and intermediate state computation (a smoothing state) which is used to determine more suitable Jacobian matrix calculation points and Taylor series explosion points. The smoothing state is determined based on an initial state update. Based on a combination of this smoothing state and the original state estimate of the respective iteration step, the new calculation point and explosion point are computed. The computation is described in the following.

3.4.1. Technical description

The following description of computation follows He et al.[13]

To determine the first state estimation and update, the EKFS algorithm follows the normal EKF algorithm. The dynamics matrix F is formulated as shown in Equation 3.1. The superscript 0 indicates that this is a preliminary parameter before the recalculation of the Jacobian calculation point and the Taylor series explosion point.

$$F^0_{k+1,k} = \partial f(x)/\partial x \tag{3.1}$$

Here, $x = \hat{x}_k$, meaning that the dynamics equation is found based on the state estimation resulting from the previous iteration step.
Having determined the dynamics matrix, the covariance matrix prediction can be computed, as shown in Equation 3.2.

$$P^0_{k+1|k} = F^0_{k+1,k} P_k (F^0_{k+1,k})^T + Q \tag{3.2}$$

Here, Q depicts the noise covariance matrix. Following from this, the gain matrix can be determined through Equation 3.3, where R is the measurement noise matrix.

$$K^0_{k+1} = P^0_{k+1|k} (H^0_{k+1})^T (H^0_{k+1} P^0_{k+1|k} (H^0_{k+1})^T + R)^{-1} \tag{3.3}$$

Next, the state prediction for the current iteration step is determined, see Equation 3.4.

$$x^0_{k+1|k} = f(\hat{x}_k) \tag{3.4}$$

Having determined the state prediction and the gain matrix, the first state update can be performed to determine the first preliminary state estimation, as shown in Equation 3.5. Again, this is only a preliminary state update.

$$\hat{x}^0_{k+1} = x^0_{k+1|k} + K^0_{k+1}[z_{k+1} - h(x^0_{k+1|k})] \tag{3.5}$$

This concludes the first step of the Extended Kalman Filter with smoothing, the determination of a preliminary state estimation. As can be seen, it is in principal the same as the normal EKF. The additional benefit arises from the determination of new explosion and calculation points and the repeated application of EKF theory, as is showcased in the following.

Determining the smoothing state

Having determined the approximated target state via the same algorithm as used in the EKF, a new starting point for the iteration, the smoothing value, is computed, using Equation 3.6.

$$\hat{x}_{k|k+1} = \hat{x}_k + P_k(F^0_{k+1,k})^T(P^0_{k+1|k})^{-1}(\hat{x}^0_{k+1} - x^0_{k+1|k}) \tag{3.6}$$

This smoothing value is now used to compute the new Jacobian matrix calculation point to be used in the second phase using the EKF algorithm. First, this point is determined, which forms the main difference to the normal extended Kalman filter, where it is assumed that the Jacobian matrix calculation point and the linear explosion point are the same. In the original algorithm by He et al., the measurement matrix is also updated via a new measurement matrix calculation point, however, with the filter at hand, this step can be omitted since the measurement matrix is only dependent on the state vector dimension, which does not change.

Repetition of EKF algorithm

The new Jacobian matrix calculation point is computed as presented in Equation 3.7.

$$x_{J-calc} = (\hat{x}_k + \hat{x}_{k|k+1})/2 \tag{3.7}$$

Having determined this new calculation point, a similar algorithm to the normal EKF is applied to incorporate the changes, starting with the recalculation of the dynamics matrix, shown in Equation 3.8.

$$F^0_{k+1,k} = \partial f(x)/\partial x|_{x=x_{J-calc}} \tag{3.8}$$

The new measurement matrix is the same as the old one, since the dimension of the relevant state vector does not change through the EKF smoothing step, as was discussed above.

$$H_{k+1} = H^0_{k+1} \tag{3.9}$$

Again, the covariance matrix prediction is determined, using the new dynamics matrix, as shown in Equation 3.10.

$$P_{k+1|k} = F_{k+1,k}P_k(F_{k+1,k})^T + Q \tag{3.10}$$

The gain matrix is also determined the same way as before (Equation 3.11).

$$K^0_{k+1} = P_{k+1|k}(H_{k+1})^T(H_{k+1}P_{k+1|k}(H_{k+1})^T + R)^{-1} \tag{3.11}$$

The new computation of the state prediction (Equation 3.12) incorporates the effect of having chosen the smoothing state as the new linear explosion point by using the new dynamics matrix in combination with the old and new explosion point.

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k+1}) + F_{k+1,k}(\hat{x}_k - \hat{x}_{k|k+1}) \tag{3.12}$$

As before, the state update is computed, as presented in Equation 3.13. In the algorithm presented by He et al. there is an intermediate step in which the measurement prediction is adjusted by the difference of the measurement equation $h(x)$ applied to the preliminary state update and the measurement matrix H multiplied with the preliminary state update. However, as was explained before, in the filter at hand the measurement equation $h(x)$ is simply equivalent to a matrix multiplication of H with the state x. This extra term is thus omitted.

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_{k+1}[z_{k+1} - H_{k+1}\hat{x}_{k+1|k}] \tag{3.13}$$

Finally, the covariance matrix is updated as in the normal EKF (Equation 3.14).

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k} \tag{3.14}$$

The next iteration can then be started based on the new state estimate $\hat{x}_{k+1}$ and the new covariance matrix P_{k+1}.

3.4.2. Functional breakdown

Following the implementation of the EKF used by DLR, several decision points are featured in the EKFS. They are essentially the same decision points as in the EKF. Generally, a state prediction and subsequent update are only performed when more time than 0.0005 seconds have passed. This is to ensure that a substantial state update can be performed, and to make sure that no faulty states of the servicer satellite (for example with wrong time stamps) are used to initialise a new state update. The time stamp that is used as a check is extracted from the servicer state. If the time difference to the time of the last state estimation is less than the time threshold the current state estimate is returned to the GNC system.

After the first preliminary state prediction step, a measurement compliance check is performed. The check is implemented through a functionality used by DLR on all satellite navigation filters. At this decision point, it is checked whether the measurement used currently by the filter has been initialised in a previous iteration step, which is critical to perform a measurement prediction. If this is not the case, the state prediction is returned instead of a new state update.

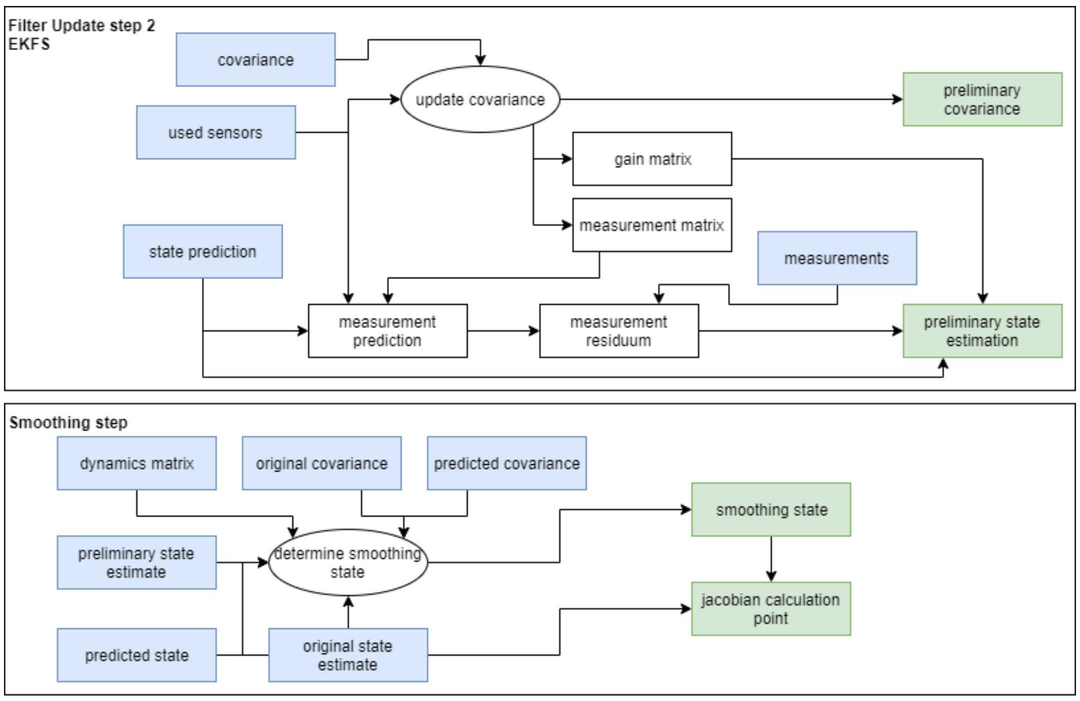


Figure 3.1: EKFS structure in addition to EKF structure

This decision point structure matches the EKF. However, the EKFS performs another step as was explained before, where a preliminary state estimate and a preliminary covariance matrix are used to find new explosion and calculation points. Figure 3.1 shows the differences of the EKFS to the EKF, starting from point "Filter update step 2". Before, all steps are the same as in the EKF (compare Figure 2.4).

After the smoothing step, the EKFS essentially repeats the same approach as the EKF, using the smoothing state as shown in the equations above. At this point, the decision points from the original EKF algorithm are skipped since it is already clarified that a prediction and update step are performed.

3.5. Unscented Kalman Filter

As was described before, the main difference between the conventional EKF and the UKF is that the EKF simplifies the distribution of possible states into one state estimate for propagation, the mean of the distribution cloud. By doing so, the distribution of potential states, or state space, which changes over the propagation will not actually be adjusted in the EKF, and the possibility distribution summarized by the mean (or state estimate) in the EKF may differ dramatically from the true state. The UKF addresses this by propagating multiple possible states, the sigma points, which represent the distribution of the state space, and then summarising them into a mean after the propagation. This preserves the actual distribution of possible states and reduces the propagation error, in particular over multiple iterations.

3.5.1. Technical description

The UKF is initialized with the same known parameters as all the other states, meaning a state estimation of vector size n and a covariance matrix of size nxn at a time step k. Furthermore, the process noise matrix Q, size nxn is available.

First, the sigma vectors at time k are determined and stored in a $nx(2n+1)$ matrix. The mean vector is taken directly from the state vector, as shown in Equation 3.15. The remaining $2n$ sigma vectors are computed following Equation 3.16 (for all sigma vectors $i = 1, ..., n$) and Equation 3.17 (for all sigma vectors $i = n + 1, ..., 2n$). A lower triangular Cholesky factorisation is used to calculate the matrix square root, following the documentation of Choi et al. and Wan and Van der Merwe.[8][10]

$$\kappa_{0,k} = \hat{x}_k \tag{3.15}$$

For $i = 1, ..., n$

$$\kappa_{i,k} = \hat{x}_k + \left(\sqrt{(n+\lambda)(P_k + Q_k)}\right) \tag{3.16}$$

For $i = n+1, ..., 2n$

$$\kappa_{i,k} = \hat{x}_k - \left(\sqrt{(n+\lambda)(P_k + Q_k)}\right) \tag{3.17}$$

In the equations above, λ is the composite scaling parameter. It is a constant parameter that determines how far the sigma vectors are spread around the mean. λ is calculated as shown in Equation 3.18, where $alpha$ and κ are constants, and n is the state dimension.

$$\lambda = \alpha^2(n+\kappa) - n \tag{3.18}$$

Computing the sigma vectors this way ensures an even distribution about the mean of the state space while simultaneously preserving the distribution which allows to propagate it. This is represented in Figure 3.2 where the propagation of the state space is shown. Starting from the same state estimation for the EKF and the UKF, the EKF simply propagates the state estimate. However, the state estimate is simply the representation of a space of possible states, which the UKF models through the sigma vectors. By propagating the sigma vectors, the UKF preserves a more accurate representation of the state space and thus draws a more accurate state estimate at the next iteration step in the form of a weighted mean of the propagated sigma vectors (red dot).

As explained above, each sigma vector is propagated separately using the same algorithm as in the other

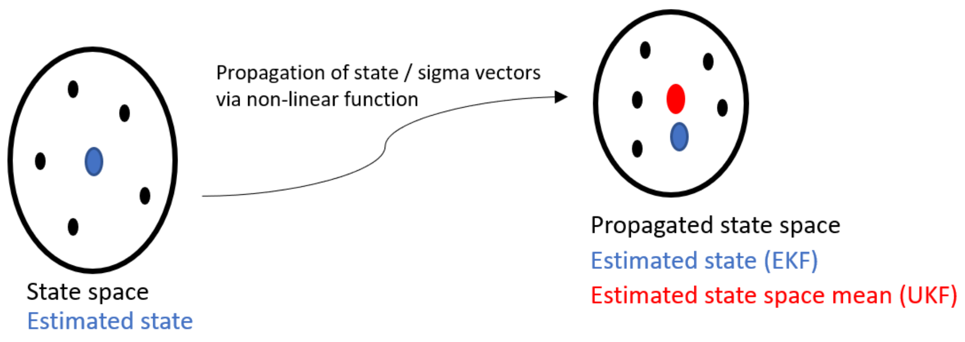


Figure 3.2: The UKF propagates a representation of the state space in form of the sigma vectors (black dots) which leads to a more accurate state estimate (mean of the state space) than for the EKF

filters (see Equation 3.19). This ensures that the propagated sigma vectors represent the actual propagated state distribution.

$$\kappa_{i,k+1} = f(\kappa_{i,k}, k) \tag{3.19}$$

Each vector in the matrix κ_{k+1} then represents a propagated sigma vector, meaning that κ_{k+1} represents the overall propagated distribution. The weighted mean of this distribution therefore is the propagated most likely state, so the state prediction from step k to step $k+1$ can be found from the weighted sample mean of the columns of κ_{k+1}. The same accounts for the predicted covariance, for which different weights are used. The weights for the state calculation (w^s) and for the covariance prediction (w^c) are determined as shown in Equation 3.20 through 3.22. Here, α is the same factor as before and β is a term adjusting the weights for higher order effects. It has been found to be ideally equal to 2.[8]

$$w_0^s = \lambda/(n+\lambda) \tag{3.20}$$

$$w_0^c = \lambda/(n+\lambda) + (1-\alpha^2+\beta) \tag{3.21}$$

For $i = 1, ..., 2n$

$$w_i^s = w_i^c = 1/(2(n+\lambda)) \tag{3.22}$$

Having found the weights, the state prediction is determined via Equation 3.23.

$$\hat{x}_{k+1|k} = \sum_{i=0}^{2n} w_i^s \kappa_{i,k+1} \tag{3.23}$$

Similarly, the predicted covariance matrix is populated using the sigma vectors and the covariance weights. Each combination of propagated sigma vector and state prediction yields a covariance matrix. The weighted average of all these individual covariance matrices then results in the predicted covariance matrix (Equation 3.24)

$$P_{k+1|k} = \sum_{i=0}^{2n} w_i^c (\kappa_{i,k+1} - \hat{x}_{k+1|k})(\kappa_{i,k+1} - \hat{x}_{k+1|k})^T \tag{3.24}$$

The predicted measurement vector is computed in a similar way as the state prediction. The measurement prediction from each sigma vector is determined and the weighted average of the result yields the measurement prediction (see Equations 3.25 and 3.26, respectively).

Incorporating measurement delay functionalities in the UKF
As was described before, all filters need to have the ability to incorporate delayed measurements for measurement prediction. This is due to the fact that in a realistic system environment the filter update and measurement update time may differ. Additionally, the image processing may take longer than a filter iteration step, meaning that the time stamp of a measurement input to a filter may not coincide with the current time step. Conventionally, this is fixed by saving a state estimate in a state buffer and, in case of a delayed measurement, extracting the state estimate from the past that corresponds to the measurement at hand. However, this does not work as such in the UKF, since not all sigma vectors from all timesteps in the past are saved. This would require too many resources. On the flipside, if all measurement predictions for a delayed measurements are based on the same state estimate the update step does not function. Thus, it was decided to only save and extract the single state estimates and reconstruct the corresponding sigma vectors using Equations 3.15 through 3.17 when they are needed to base the measurement prediction on. Minor errors are expected due to the use of the latest covariance matrix in the recalculation of the measurement updates. These errors are assumed small enough to be neglected at this stage.

$$\zeta_{i,k+1} = h(\kappa_{i,k+1}) \tag{3.25}$$

$$\hat{z}_{k+1|k} = \sum_{1=0}^{2n} \zeta_{i,k+1} \tag{3.26}$$

The predicted output covariance is then populated similarly to the predicted covariance matrix, but using the measurement prediction of the sigma vectors and the measurement prediction vector instead of the propagated sigma vectors and the state prediction vector, as seen in Equation 3.27.

$$P_{k+1}^{yy} = \sum_{i=0}^{2n} w_i^c (\zeta_{i,k+1} - \hat{z}_{k+1|k})(\zeta_{i,k+1} - \hat{z}_{k+1|k})^T \tag{3.27}$$

The innovation covariance matrix P^{vv} is found using Equation 3.28 with a combination of the predicted output matrix P^{yy} and the measurement noise matrix R.

$$P^{vv}_{k+1} = P^{yy}_{k+1} + R_{k+1} \tag{3.28}$$

In order to assess the difference between the state prediction and the measurement prediction, the cross correlation matrix P^{xy} is computed using Equation 3.29.

$$P^{xy}_{k+1} = \sum_{i=0}^{2n} w_i^c (\kappa_{i,k+1} - \hat{x}_{k+1|k})(\zeta_{i,k+1} - \hat{z}_{k+1|k})^T \tag{3.29}$$

Finally, the new gain matrix is determined from Equation 3.30.

$$K_{k+1} = P^{xy}_{k+1}(P^{vv}_{k+1})^{-1} \tag{3.30}$$

As was the case in the previous filters, now that the state prediction, the covariance prediction and the gain is known, the state and the covariance can be updated using Equation 3.31 and 3.32 ,respectively.

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_{k+1}[z_{k+1} - \hat{z}_{k+1|k}] \tag{3.31}$$

$$P_{k+1} = P_{k+1|k} - K_{k+1}(P^{yy}_{k+1} + R_{k+1})K^T_{k+1} \tag{3.32}$$

Now that a new state estimation and covariance for the following time step have been found the next filter update can commence.

Throughout the filter update of the UKF it can be seen that it is in parts similar to the conventional EKF. One of the differences is the use of multiple sigma vectors representing the state space rather than just a single estimation of the most accurate state. Furthermore, the update process differs since it does not involve the computation of a Jacobian matrix which makes the implementation more suitable to higher order non-linear systems.

Weighted average of sigma vectors to determine the new state estimate
As was explained in this section, the UKF relies on the propagation and update of sigma vectors. The weighted average of the propagated sigma vectors eventually yields the new state estimate. The satellite state used by DLR and in all the following chapters contains the position, the velocity, the attitude and the attitude rate. The attitude is given in quaternions rather than Euler Angles. Since quaternions feature a more complex interplay than Euler Angles and are inherently linked taking a weighted average of the quaternions in multiple sigma vectors may not yield an accurate representation of the true mean state of the state distribution. This potential problem was only realised after finishing the research for this report and could not be explored further any more, but should be noted for future tests and implementation.

3.5.2. Functional breakdown

The structure of the UKF is more similar to the conventional EKF rather than the EKFS, in that it features a prediction and an update step without an intermediate smoothing step. In the same way as for the other two tested filters, a propagation check is performed based on the time difference between the incoming servicer satellite state and the latest state estimation (omitted in Figure 3.3), followed by a measurement initialisation check after the prediction step.
The breakdown of the filter functions can be seen in Figure 3.3. The same colour coding as in previous breakdowns is used.

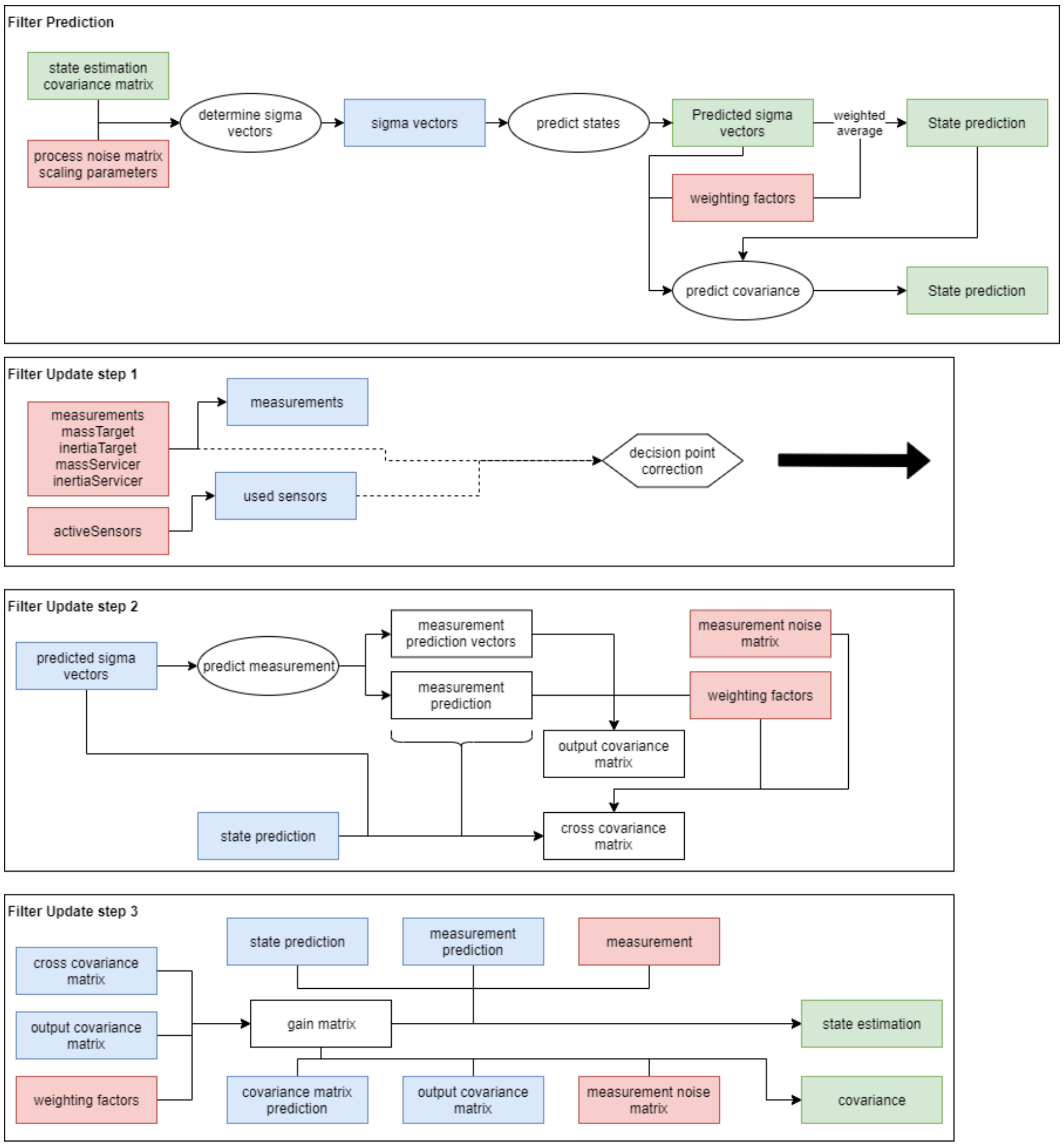


Figure 3.3: UKF structure

4 Filter performance simulation testing

After having developed two potential EKF alternatives as described in Chapter 3, filters need to be tested in order to collect data on their performance. As was discussed in Chapter 1, two distinct testing methods, a simulation performance test and a hardware-in-the-loop test, are applied. This chapter explains the simulation performance testing approach, the test outcomes and the rating of the test data against the performance criteria.

The reasoning why a simulation test is performed before the hardware-in-the-loop facility testing was outlined in Section 1.9.5.

4.1. Performance criteria for filter assessment

The selected and developed filter modes can be tested in a simulation performance test before the hardware test. In order to judge the filter modes, several criteria are developed. Since the filters can be fed with simulated input data, a quantitative comparison is possible based on the numerical output of the individual filter modes given the same input conditions are fed to each filter. The criteria for quantitatively assessing filter mode performances were presented in section 1.9.10 and are repeated here for convenience:

- Accuracy: determines how close the state estimate is to the true state (simulated true state in the simulation performance test)
- Time to settle: determines how fast the oscillations of the estimate around the true state converge
- Stability of the estimate: determines how close the estimate is to the target converges to the true state
- Bias: determines whether a parameter of the state estimate is consistently offset from the true value
- Computational complexity and requirement of resources: determines how much time it takes to run a filter per iteration

The new filter modes, EKFS and UKF, can be judged against the original EKF by comparing their respective performances. Hereby, the following rating criteria will be applied:

- Very poor performance (–): A filter performs 20 + % worse than the original EKF in a given parameter
- Poor performance (-): A filter performs 5% to 20% worse than the original EKF in a given parameter
- Neutral performance (0): A filter performs within a bound of ±5% of the original EKF in a given parameter; this bound is set since it is assumed that any performance difference of less than 5% is dependent on the test conditions and not all possible test cases can be assessed
- Good performance (+): A filter performs within 5% to 20% better than the original EKF in a given parameter
- Excellent performance (++): A filter performs more than 20% better than the original filter in a given parameter

In the eventual rating process, each (–) is scored as (-2), each (-) is scored as (-1), each (+) as (+1) and each (++) as (+2).

4.2. The simulation performance test

The simulation performance test is performed by simulating the pose of two satellites, a target satellite and a servicer satellite (or chaser). Their states are propagated parallelly, so that a relative orientation and distance between the two can be concluded at any given iteration step. Furthermore, the state of the target is not only simulated via propagation but is also estimated from the position of the servicer satellite using the satellite navigation filters EKF, EKFS and UKF at every iteration step. The satellite navigation filters are fed with the estimates from their respective previous iteration and simulated faulty measurements based on the relative "true" simulated position of the two satellites. Thus, all filters are supplied with the same measurements and servicer satellite positions, but their target state estimates can diverge over time, allowing for a fair comparison between the filters as long as they are initialised with the same parameters.

The orbits of target and chaser are Kepler orbits, and the orbital model is a classic two body model consisting of Earth and the individual satellites. Influences from other bodies, such as sun and moon, are neglected, as well as force inputs. This means the satellites propagate idly. The target and servicer orbits are both computed (and estimated by the filter) in the ECI coordinate frame.

The update process of the simulation performance test is visualised in Figure 4.1. Here it can be seen that from one time step to the next, the "true" states are propagated and the filter is updated each time, resulting in a new state estimation.

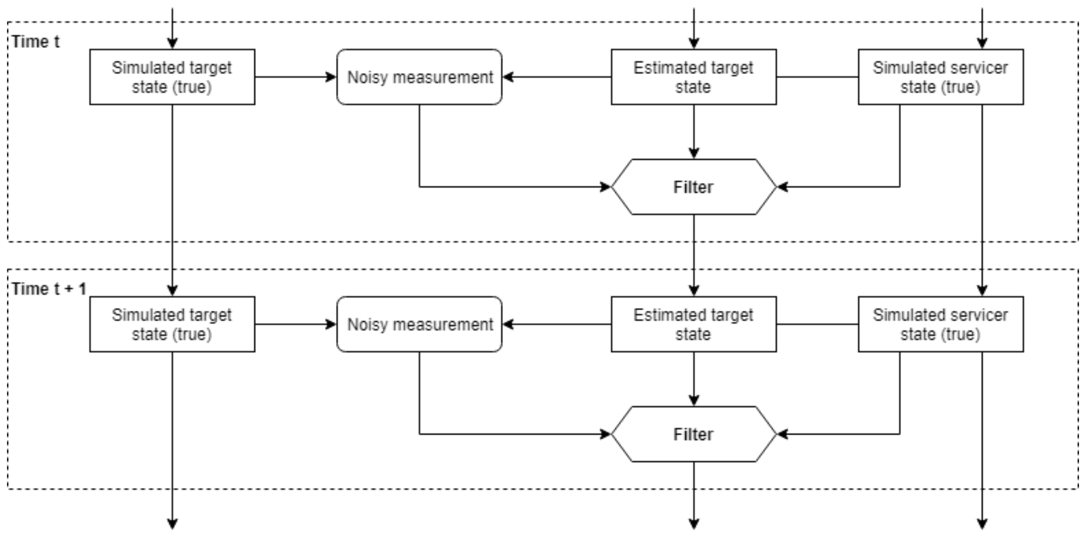

Time t
Simulated target state (true)
Noisy measurement
Estimated target state
Simulated servicer state (true)
Filter
Time t + 1
Simulated target state (true)
Noisy measurement
Estimated target state
Simulated servicer state (true)
Filter

Figure 4.1: Simulation test high level overview

4.2.1. Initialisation of performance test

Several parameters are fed to the filter to start a test run, which were mentioned in the description of the different test cases. They are described more detailed here.

State simulation: In order to initialise the simulation performance test, the state of the target satellite and the servicer satellite is simulated. This yields the "true" state for both satellites. These states are propagated through an orbital simulation model and will continuously be used as reference over the course of the simulation performance test to determine the measurements of the target state by the servicer satellite. After the test, the true target state is compared to the estimated target state resulting from the filter to determine the performance quality.

Initial target state estimate: The original state estimate of the target satellite needs to be simulated and fed to the filter to initialise the iterative process. The state estimate is usually chosen close to the true state, but can also lie far away, depending on the test case.

Satellite to satellite state difference: The filter is initialised using an original state estimate for the target satellite which is chosen with a set difference to the true state. Thus, two "true" satellite states are available, together with a measurement and an original state estimate of the target state. This is enough input for the satellite navigation filter to determine a state estimate for the next time step. By propagating the "true" states, the same inputs are found at the next time step. The difference between the satellite states determines the distance, relative velocities and spin rates, and thus serves to put the satellites either close together or far from each other.

Measurement error simulation: The measurement of the target state by the servicer satellite is performed using an available model from DLR. In this model, white noise is added to the measurement to simulate a measurement error. The magnitude of this error can be chosen.

Measurement outlier simulation: Outliers can be turned on or off for a test run. They are simulated by defining a measurement outlier and a specific iteration step at which they should occur. At this step, the outlier measurement replaces the normal measurement and thus offsets the filter.

Measurement update frequency simulation: Conventionally it is assumed that the for each iteration step of the filter a new measurement is available. However, this may not be the case. If a filter iteration step is 0.1 seconds but a sensor requires 0.5 seconds, the filter has to reuse the same measurement or purely rely on propagation for 5 iteration steps until a new measurement is available. The performance test is built as such that the update step for the filter iteration and the measurement can be set independently. If the update step for the measurement is longer the performance test does not generate new measurements until a new measurement is "allowed" based on its iteration time. This can be used to either simulate a measurement update delay or to simulate aliasing effects, since the measurements will come in more stepwise and less smoothly than the filter performs iterations.

Measurement delay simulation: It was observed in the past that real-life, real-time image processing can delay the input of a measurement into a filter by a substantial amount of time (up to several seconds). To simulate this, the simulation performance test is initialised with a set delay time of either 0 seconds (no delay, measurements are passed straight to the filter), or any number of seconds. In the latter case, the measurement is passed to a buffer that has the size of the ratio between the measurement delay and the filter update interval. Thus the buffer is passed a new measurement at every filter iteration. Once the buffer is full, the oldest measurement of the buffer is passed to the filter.

4.3. Test cases used in simulation test

In order to compare the different filter modes in a meaningful way, different test cases need to be developed. These help to ensure that the filters are tested under the same conditions and that a relevant range of test scenarios is covered. This is important since a filter mode may perform better than a second one in one test case, but may not perform well or even poorer in another one. Furthermore, different test cases should ensure that realistic conditions are assessed as may be encountered by filters in real missions. This is crucial since filter modes need to be judged on the quality of performance in scenarios that are relevant for them, rather than scenarios that they would never encounter. There is no point in performing well under completely unrealistic conditions.

Since the simulation performance test propagates the target and servicer satellite states idly, the initialisation conditions of the test are decisive for a test case scenario. By breaking down the different possible inputs test cases are defined. As was discussed before, test cases should reflect realistic satellite conditions that could similarly be encountered in a real mission. In addition to this, it is also interesting to use test cases that could be modelled in a similar way in the EPOS 2.0 facility to be able to compare filter modes. This helps to judge the quality of the simulation performance test.
The breakdown of the parameters defining test cases is shown in Figure 4.2. On the high level, the rudimentary parameters that define a test run in the simulation test are identified. The states of target and servicer satellite are decisive. Furthermore, the number of iterations in a test run and the update interval (time between separate iterations) affect the test at this level. They need to be set in the beginning of a test run. Finally, the type of measurement that is fed to the filter modes will affect the performance and thus needs to be con-

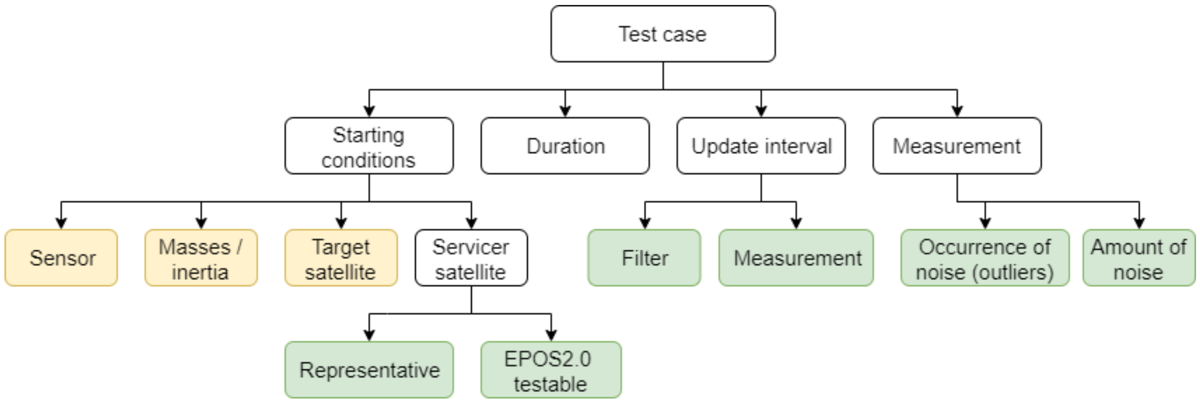


Figure 4.2: Breakdown of test case options, variable parameters (green) and constant parameters (yellow) across test runs

sidered for test cases. This includes the occurrence of outliers, the magnitude of noise, measurement delays and measurement update intervals.

One level further down, the starting conditions are differentiated for the target and servicer, the masses and inertia values of the satellites, and the used sensors. Since the target is only observed by the servicer and propagates unobstructed its starting state is kept the same for all test cases and can be fixed in the beginning to realistically represent a satellite pose in orbit. Once it is fixed it is irrelevant for the differences between test cases. The servicer state also needs to be realistic for a satellite in orbit and can be chosen either far from or close to the target satellite. This distinction is made to identify differences in filter performance when a filter mode estimates a target state under different conditions which will greatly affect the relative measurement quality. Furthermore, the distinction is interesting since the EPOS 2.0 facility can only simulate relative states less than 20 metres apart.
Moving on to the update interval, a distinction can be made between the update interval of the filter and the update interval of the measurement. The update times of filter and measurement can differ. In real-world operation, it was observed that the filter usually updates faster than the measurement. This means that multiple consecutive filter iterations may be supplied with the same measurement. DLR has implemented a functionality in their navigation filter that ensures to neglect outdated measurements and, in these cases, simply propagate the state estimate. To simulate this, filter and measurement update step times can be altered. Secondly, it was observed that sensor images can show aliasing effects, where sharp corners are sensed due to the image resolution, which are not actually there. As an example, a slow moving abject that is observed may be assumed to be in one position since it overlaps with the same pixel for several measurements and then suddenly "jumps" to the next row of pixels, which offsets the estimated position. This effect is common in signal processing and can be simulated by "withholding" measurements for several filter iterations.
In addition to the update interval, measurements can also be delayed due to image processing times. This means that a filter is always fed a slightly outdated measurement. This effect is incorporated in the simulation performance test by setting a delay time at initialisation. A ratio of the delay time divided by the filter update time is computed and rounded to the next integer, which is used to initialise a buffer of the size of this integer. The buffer is filled with the new measurement at every filter iteration. Once it is full it returns the oldest measurement and feeds it to the filter. Thus, a delayed measurement is simulated.
Measurement noise can be altered for a test case by setting the amount of noise and the time of occurrence of noise. This is used to simulate measurement outliers.

Having identified the possible differences between test cases, test cases can be designed to identify the performance of the filter modes under the changing conditions. This is done by adjusting one or more of the parameters mentioned above.

16 different test cases were defined for the simulation performance test. The different test cases are presented in Table 4.1.

In the following, the test case initialisation parameters are explained:

Table 4.1: Test cases for simulation test - initialisation

Test case	Range to target	Outlier	Filter dt [sec]	Measurement dt [sec]	Measurement delay [sec]
1	1000 m	-	1.0	1.0	-
2	1000 m	-	0.1	0.1	-
3	1000 m	-	0.1	0.5	-
4	14 m	-	1.0	1.0	-
5	14 m	-	0.1	0.1	-
6	14 m	-	0.1	0.5	-
7	1000 m	Large	0.1	0.1	-
8	14 m	Large	0.1	0.1	-
9	14 m	-	0.1	0.1	0.5
10	14 m	-	0.1	0.5	0.5
11	14 m	-	0.1	0.3	0.5
12	14 m	-	0.1	0.1	0.3
13	14 m	-	0.1	0.5	0.3
14	14 m	-	0.1	0.3	0.3
15	14 m	Small	0.1	0.1	-
16	14 m	Small (10)	0.1	0.1	-

Range: The range parameter sets the relative position of the servicer satellite with respect to the target. The servicer position is based on the fixed target position. In close operating condition, the distance is set to 14 metres in x direction behind the target satellite. In far operating condition, it is set to 1000 metres behind the target in x, y and z direction.
These two ranges were chosen to identify the accuracy in close and far operation, especially given that measurement accuracy changes with distance to target.

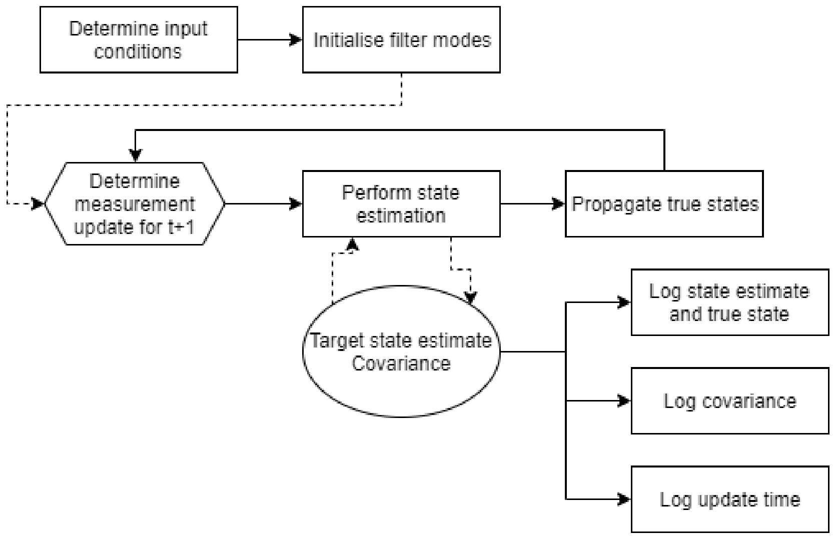
Measurement outliers: Measurement outliers are either large or small. They are chosen to reflect either slight offsets or great disturbances to identify whether filters can resettle the estimate around the true target states. Outliers are generated like normal measurements. However, they are not based on the true target state but rather on an intentionally faulty target.
Large measurement outliers are implemented after 100 iterations. The faulty target state has a height of 7e6 m, ascension of 0.7 pi and inclination of 0.8 pi.
Small outliers occur after 1500 iterations. They are based on a target state which is derived from the true target state but has an offset to the true state of 1 metre in x, y and z direction.
The small outliers occur later in the test run since their smaller offset may be hidden in the filter estimate error fluctuation early on in the test run. The amount of offset in the small outlier case was chosen based on expert advice. The large outlier offset was generated with random orbital parameters once and fixed afterwards.

Filter update time: From real-world experience it is known that the navigation filter run at around 10 Hz, whereas the measurement update interval can lie anywhere between 0.1 and 4 seconds. Thus, these parameters are altered across the different test cases.

Measurement delay: The measurement delay is varied in test cases 7 through 12 to identify whether there are accumulation errors between the same or different measurement update times and delays.

4.3.1. Structure of performance test

This section outlines the practical setup of the simulation performance test. The process-flow is shown in Figure 4.3. After the definition of the test cases (see Section 4.3) and the resulting definition of inputs required to perform the filter iteration, the simulation test can be started. All filter modes that are tested are initialised using the estimated target state, the original noisy measurement from the true target and servicer satellite states, the active sensors and the update time difference. Based on these inputs at time t, each filter mode separately determines a new state estimate for time t+1. This state estimate is passed to a logging function for each filter. Furthermore, the time it took the filter mode to compute the new state estimate is logged, as well


Determine input conditions
Initialise filter modes
Determine measurement update for t+1
Perform state estimation
Propagate true states
Target state estimate Covariance
Log state estimate and true state
Log covariance
Log update time
Figure 4.3: Update structure of filters in the simulation test

as the covariance matrix at t+1. Then, the true states are propagated using the same orbit propagation model that is used in the filters. At the new time step, it is checked whether a new measurement should be passed to the filter modes, based on the measurement and filter update interval time. Then the iteration is repeated and the new results are logged.
All tests are run for 3000 iterations.

4.4. Setup of analytical assessment

The outputs of the performance test are logged in csv files, one for the each filters state estimate and covariance, respectively, and one file for the iteration time log. These csv files are then read into a Python tool that was specifically designed for this purpose.

The Python tool utilises the pandas library to read csv files and store the content in data frames. Thus, each logged parameter can be accessed.

4.4.1. Data read-out

For each logged parameter, the data is read into data frames and then analysed.

State estimates: The state estimate logger in the performance test logs both the state estimate and the true target state in the same file, meaning that for each filter both these states are logged side by side and can be read out in Python as such. This facilitates the analysis greatly since no matter whether test cases are changed or not, the corresponding true state to the state estimate is available and is read out at the same time.
After read-out, states are stored in data frames which allows to compare individual parameters or overall deviations of the respective estimate from its corresponding true state.

Covariances: Covariance matrices do not have a corresponding "true" covariance and are thus read out separately. The assessment is thus not possible with a true reference covariance but only becomes meaningful in comparison with the other filters.

Time log: The time log stores the amount of time a filter runs to determine a new state estimate at each iteration step. The log file contains the information for all filters for a specific test case. The read out of this file thus allows for a direct filter comparison.

4.4.2. Working principle of performance assessment

The performance assessment is broken down into several stages. A code sample is shown in Appendix A.

Read-in and data frame extension:

In this first stage, the filter output data is read from csv files into pandas data frames. Each filter output is read into a separate frame. Since this initial data read-out contains the estimated and true state parameters but not the deviation of the estimate from the true parameter the data is modified using the functions *posDataFrame(df)*, *velDataFrame(df)*, *quatDataFrame(df)* and *rateDataFrame(df)*. These functions all follow the same principle. They add new columns to the data frame to store the difference between the parameters, 3 columns for x, y and z position in *posDataFrame*, 3 columns for x, y and z velocity in *velDataFrame*, 4 columns for quarternions x, y, z and s in *quatDataFrame* and 3 columns for the spin rate differences in x, y and z in *rateDateFrame*. Equation 4.1 shows an example for the position error computation using the data frame structure. In addition, *posDataFrame* and *velDataFrame* add another column for the absolute position and velocity error (combination of error in x, y and z direction), respectively, as shown in Equation 4.2.

$$\begin{aligned} df[x-position-error] &= df[estimated-x-position] - df[true-x-position] \\ df[y-position-error] &= df[estimated-y-position] - df[true-y-position] \\ df[z-position-error] &= df[estimated-z-position] - df[true-z-position] \end{aligned} \tag{4.1}$$

$$abs_{err} = \sqrt{x_{err}^2 + y_{err}^2 + z_{err}^2} \tag{4.2}$$

Since it was deemed that quarternions were not always suitable for visualisation purposes another function was implemented that adds 3 columns with the attitude error in Euler angles in x, y and z orientation, which are computed based on the quarternion error computed before.

The thus found errors in the filter parameter estimation can be used for filter performance assessment.

High level performance assessment:

For a high level performance assessment the total error and mean error by state parameter is determined for every filter. For an assessment of position and velocity the total accumulated error of the absolute error over the entire test run is determined. Based on the total error, a high level computation of the improvement of the new filter modes over the original EKF is performed, as in Equation 4.3, where *error* represents the error of a state parameter estimate, *i* is the filter test iteration number and i_{test} is the total number of iterations in a test run.

$$\%-improvement-to-EKF = \left(1 - \frac{\sum_{i=0}^{i_{test}} error_{filterMode}}{\sum_{i=0}^{i_{test}} error_{EKF}}\right) \cdot 100 \tag{4.3}$$

New testing approach for numerical comparison:

Filter performance assessment is usually discussed when new filters are developed. However, filter performance comparison seldom advances beyond a simple comparison of high level accuracy assessments and a comparison to an arbitrary baseline. Thus, a new filter assessment approach was developed to be able to compare filters fairly and using mathematically determined performance parameters.

This new performance assessment allows to judge the convergence of the filter numerically. It was observed that usually when a filter estimate converges towards the true state it doesn't do so smoothly but it fluctuates towards and around the true state. This fluctuation is reflected in a fluctuation of estimate errors as well, meaning that the estimate error increases and decreases, and thus, over the course of the test runs, forms local maxima (or highpoints). This can be used to determine mathematically when convergence is reached.
It was defined that for an estimate to be converged it needs to stop continuously approaching the true state but rather needs to have started fluctuating around the true state. The point where convergence is said to start thus splits the test run into a section before convergence, and a tail-section where the estimate is converged.

To determine the onset of convergence, the parameter error at each iteration step is determined, as was shown above. Next, it is assessed for each iteration step whether a there is a local error maximum. Each local maximum throughout the test run is saved in a new data frame. Out of the list of local maxima the first low local maximum is determined, meaning that both the local maximum before and after have higher error values. This first low local maximum is called the *boundary point*, the point at which the error is already converged. To identify the quality of the convergence, the maximum error occurring in the test run in the section after the boundary point is found. This value is then compared to the local maxima before the boundary point. The latest local maximum that is higher than this value occurring in the test run is set to be the *convergence point*. It marks the start of convergence.
Several parameters can be deduced from this approach: The time to convergence (time it takes to reach the convergence point), the quality of convergence (maximum error occurring after the boundary point) and the absolute error before and after convergence. In addition, for filter comparison, it can be assessed how long it takes a certain filter to converge to within the same convergence bounds as another filter. Lastly, if no high-points can be determined it can be concluded that either no convergence is reached or that the filter instantly converges smoothly towards the target, in which case the test duration needs to be increased.
The performance improvement of the error in the tail section, the time to convergence or any other parameter of the new filters to the original EKF can then be determined using Equation 4.3 by replacing the sums in the second part by the parameters under investigation.

4.5. Simulation performance test outcomes

This section shows the results from the simulation performance test. For most position estimation errors, the errors are shown for as the absolute errors (see Equation 4.2) in blue, for the error in x position estimation in yellow, for the y position in green and for the z position in red. The data is presented in metre deviation from the true state across time of the test run. The absolute error in blue is always positive. The velocity estimation error is presented in the same way as the position error, given in metres per second deviation from the target state.
The attitude error is shown for the quaternion components for the error in x estimation (blue), y estimation (yellow), z estimation (green) and s estimation (red). The spin rate is shown as radians per second deviation from target state, around the x axis of the target (blue), the y axis of the target (yellow) and the z axis of the target (green).

4.5.1. Comparison setup and baseline filter performance

In all 16 test cases the filter estimates converged towards the simulated true target state. However, the filter performance differs greatly by test case and by filter. For the following discussion, several test cases were eventually excluded:

- Test cases 1 and 4: It was observed that filters can (and in reality would) always run at around 10 Hz. Since the 1 Hz filter update test cases thus didn't seem representative anymore they are mitigated.
- Test cases 7 and 8: Despite the fact that all filters could recover from large outliers, the great offset in the estimate that they caused would not serve well for a comparison with the overall filter performance. Figure 4.4 illustrates this, where the response of the EKF to a large outlier can be seen. After discussion with filter experts it was deemed highly unlikely that such an outlier would ever occur. Even if this was the case the filters could recover. For a comparison this test case is not of interest, however.

Adjustment of tail section assessment

Throughout the data assessment it was found that the new testing approach for numerical assessment presented in 4.4.2 is not suitable for all filter test cases. This is due to the fact that, especially in test cases 9 through 14, there can be early fluctuations of highpoints which causes the algorithm to register an onset of convergence very early on in the test run when estimation errors are still high. While this would indicate a fast time to convergence, which would be favourable, it could also result in very high convergence errors being recorded, which would downgrade the performance. This was primarily observed for the new filters, which seem to not be able to deal with delayed measurements and withheld measurements as well as the original EKF. Thus, the new assessment approach explained above was primarily used to assess the convergence time and quality of the original EKF and then assess how long it takes the EKFS and UKF to reach and stay within

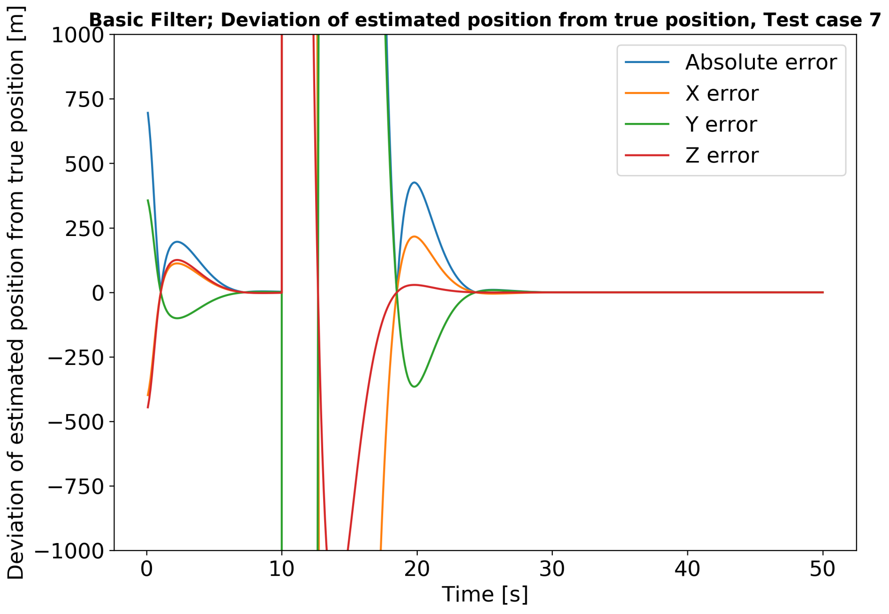


Figure 4.4: Position estimation error response of EKF to large outlier occurrence

the same convergence bounds.
In order to still be able to exclude the swing in period for all filters and fairly compare the performance once a filter estimate is close to the true state the first 10 seconds of every test run were excluded for part of the assessment. 10 seconds were chosen since it was observed that after 10 seconds all filters were swung in (exception: test case 7 and 8). Test results for which this was done are marked as "shortened" in the following.

Baseline filter performance

The EKF used by DLR at the start of the project is used as a baseline. Its position estimation (absolute position error) performance parameters for each test case are presented in Table 4.2, where an (*s*) in the header row stands for a shortened data set without swing-in.

Several insights can be gained from the data presented in Table 4.2. Firstly, as was mentioned before, the error in test cases 1, 4, 7 and 8 is considerably higher than in the other test cases. This is due to slower convergence due to slower updates (test case 1 and 4) and the large error offset due to measurement outliers (test case 7 and 8). Furthermore, it can be seen that the mean error after the 10 second swing-in period is dramatically reduced for all test cases except for the large outliers, since in test case 7 and 8 the large outliers occur right at 10 seconds into the test run, raising the average error in the tail section. In test cases 1 and 4 the error does not lower as much comparatively as in the other cases since the slower convergence means that the filter estimate is still swinging towards the true state.

Test cases 2 and 3, and 5 and 6 show very similar performance. The difference in the test cases lies in the distance between target and servicer satellite. The key performance change is the time it takes to reach convergence in test case 6 compared to test case 3. However, it was found that this is due to an anomaly where the algorithm registers an early lowpoint in the local maxima. Thus, this convergence point is reached early but still features a high error. Apart from this there does not seem to be a significant performance difference between far and close range test cases.

The difference between test cases 9, 10 and 11, and 12, 13 and 14, respectively, lies in the measurement delay time. As can be seen, the error in the former three test cases, where the measurement delay time is 0.5 seconds, is larger than in the latter 3 cases, where the delay time is 0.3 seconds. This makes sense since more recent delays should lead to better estimates. Furthermore, it can be seen that the relative error distribution between the three test cases in each of the two groups is roughly the same. Case 9 and 12 have the lowest relative error in their respective group, followed by cases 11 and 14, and lastly 10 and 13. This was to be expected since the main difference is the measurement update time. Thus it can be seen that longer measurement

Table 4.2: EKF position estimation performance in simulation test

Test case	Total error [m]	Mean error [m]	Mean error (s) [m]	Median error (s) [m]	Time to convergence [sec]
1	61656.6 m	20.552 m	10.757 m	0.175 m	162 s
2	9964.9 m	3.322 m	0.103 m	0.092 m	8.5 s
3	14989.7 m	4.997 m	0.321 m	0.283 m	8 s
4	61795.3 m	20.598 m	10.781 m	0.182 m	148 s
5	10162.3 m	3.387 m	0.104 m	0.092 m	8.5 s
6	14993.0 m	4.998 m	0.224 m	0.204 m	2.5 s
7	3851293.2 m	1283.764 m	1323.785 m	0.095 m	25.6 s
8	3851484.5 m	1283.828 m	1323.785 m	0.096 m	25.6 s
9	19880.3 m	6.64 m	0.176 m	0.108 m	12.5 s
10	26140.9 m	8.731 m	0.403 m	0.242 m	17 s
11	23015.3 m	7.687 m	0.279 m	0.176 m	3 s
12	14701.3 m	4.907 m	0.126 m	0.095 m	14.2 s
13	20349.4 m	6.792 m	0.284 m	0.223 m	12.5 s
14	17609.9 m	5.878 m	0.208 m	0.167 m	7.8 s
15	10149.2 m	3.383 m	0.101 m	0.087 m	8.5 s
16	10182.7 m	3.394 m	0.112 m	0.094 m	8.5 s

update times cause worse estimates, whether combined with delayed measurements or not.

Figure 4.5 shows the estimation error in the position estimate of the EKF in test case 9. It can be seen that the estimate smoothly converges towards the true state (0, since the error is plotted). However, since this is a highly zoomed out view, no details can be made out. A more detailed view of the entire test run (300 seconds, or 3000 iterations) of the position error response of test case 9 is shown in Figure 4.6. The position error from the EKF estimate of test case 10 is shown in Figure 4.7. The difference in average error is clearly visible between test case 9 and 10.

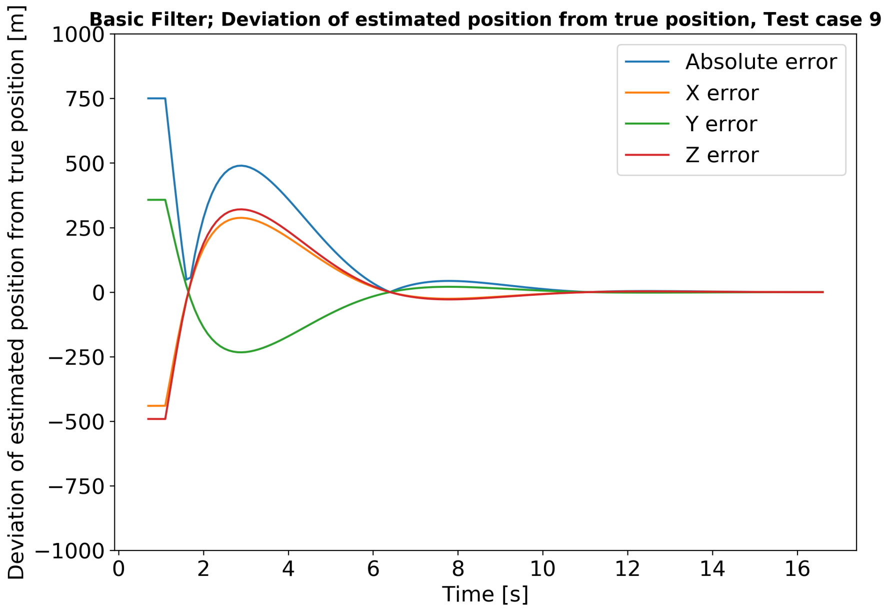


Figure 4.5: Position estimation error response of EKF with measurement delay time of 0.5 seconds, test case 9; view focused on the first 16 seconds of the test run

Again, there seems to be an anomaly of time to convergence in test case 11. However, observing the

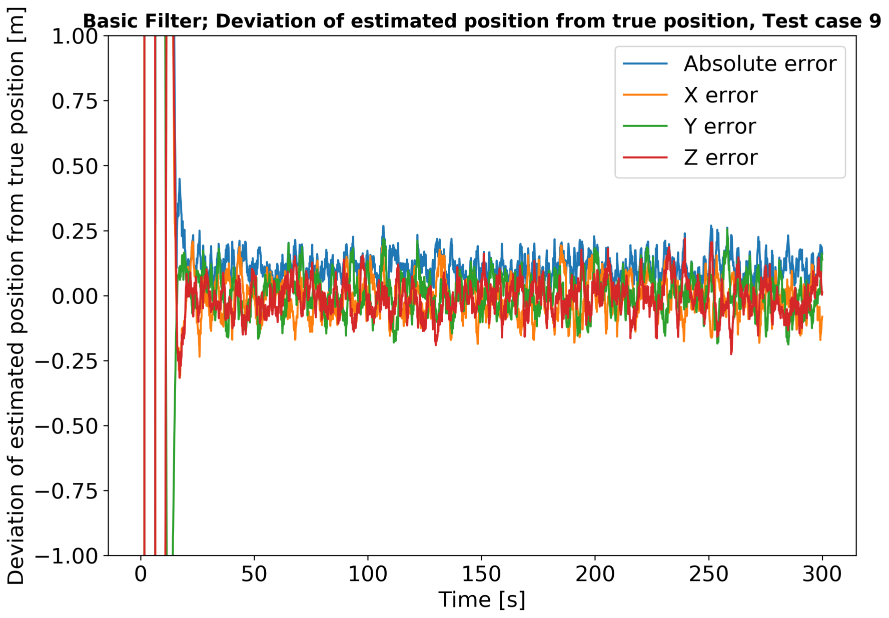


Figure 4.6: Position estimation error response of EKF with measurement delay time of 0.5 seconds, test case 9

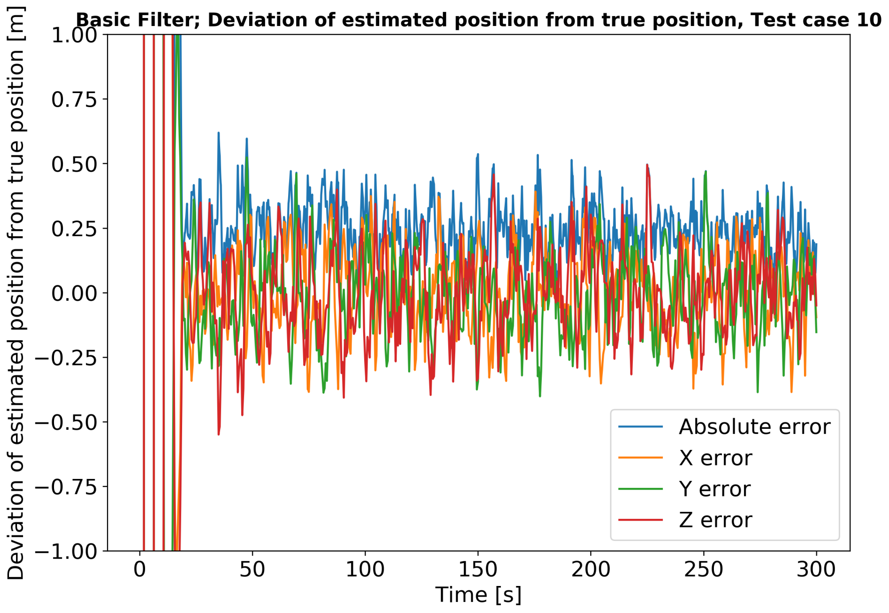


Figure 4.7: Position estimation error response of EKF with measurement delay time of 0.5 seconds, test case 10

position estimate error in Figure 4.8, no difference in the estimate error can be identified. It is assumed that the anomaly originates in the assessment algorithm determining an early convergence point and thus registering high tail section errors. Test case 11 has a measurement update time difference of 0.5 seconds. These anomalies seem to be more likely when the measurement update time is longer. This would make sense since the estimate is more likely to be thrown off by new measurements coming in after a couple of iterations of pure propagation. This would require further investigation.

From the performance of cases 15 and 16 it can be seen that the small measurement outliers have hardly any effect on the performance when comparing these cases with test case 5. The slight improvement in case 15 is attributed to the slightly smaller outlier (observe lack of error spike in Figure 4.9 as response to the outlier as compared to Figure 4.10 which shows the response in test case 16) as well as rounding. This shows that

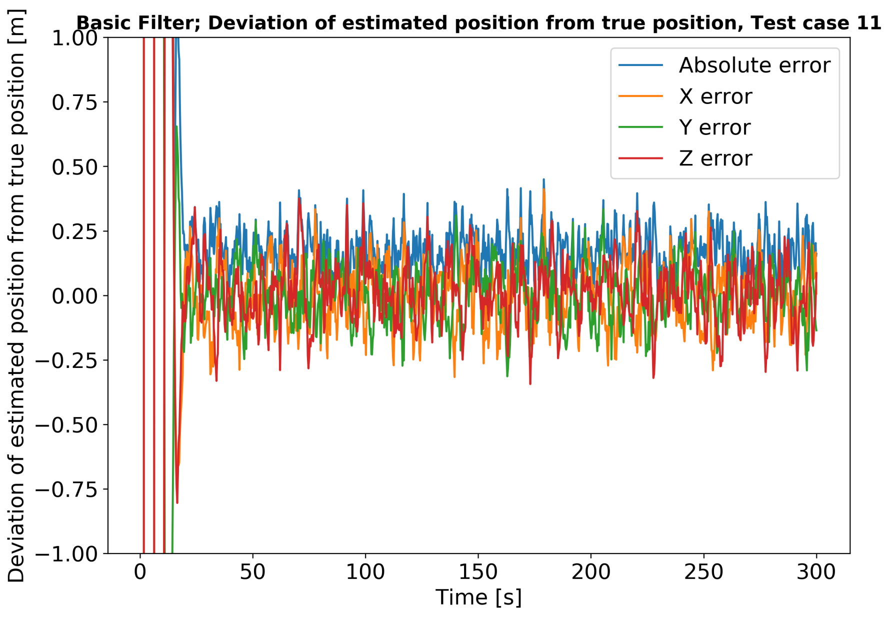


Figure 4.8: Position estimation error response of EKF with measurement delay time of 0.5 seconds, test case 11

a small measurement outlier of 1 or 10 metres has no discernible effect on the test run or the overall filter performance.

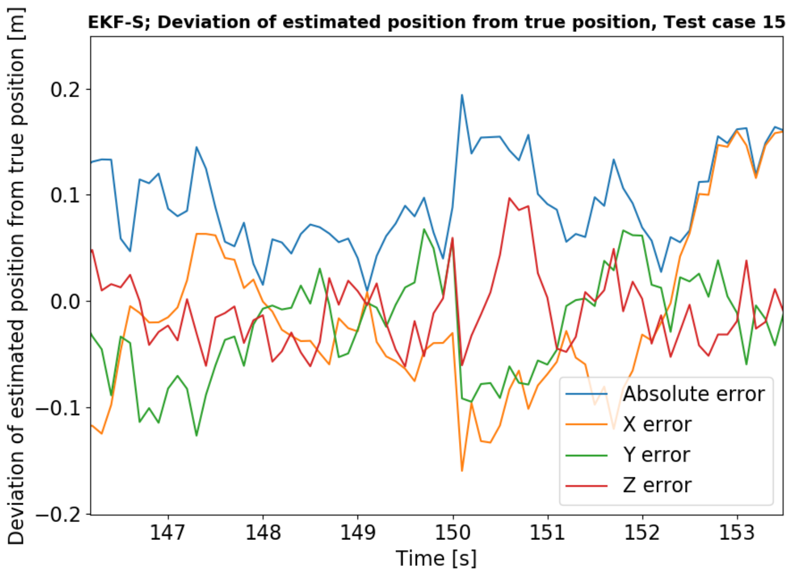


Figure 4.9: Position estimation error response of EKF to small outlier, test case 15; only a small deviation in the estimated error can be observed around 150 seconds into the test run

Remaining state parameter performance

The other state parameters are assessed the same way as the position estimate. Since the main focus lies on the position estimate, they are discussed more briefly here.

The average error of the estimated state parameters excluding the swing in period across all test cases (excluding 1, 4, 7 and 8) are shown in Table 4.3. It can be seen that the attitude angle estimation error and

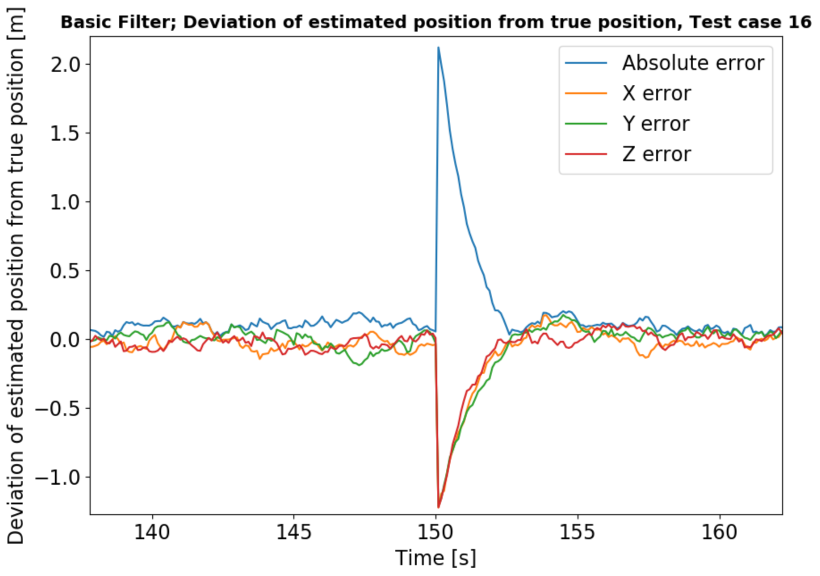


Figure 4.10: Position estimation error response of EKF to medium outlier, test case 16; a larger error spike can be observed at the occurrence of the outlier at 150 seconds into the test run

Table 4.3: EKF remaining state parameter performance, test cases excluding 1, 4, 7 and 8

Parameter	Mean error
Velocity	0.105 m/s
Attitude angle x	0.083 deg
Attitude angle y	0.083 deg
Attitude angle z	0.086 deg
Rate x axis	0.008 rad/s
Rate y axis	0.009 rad/s
Rate z axis	0.012 rad/s

the spin rate estimation error are very low. Contrary to the other filter modes (see below), the quality of performance of the EKF actually drops slightly when only focusing on test cases 9 through 14, the test cases that were deemed a more "realistic" representation of real-life conditions due to the implementation of measurement delays. This is discussed further in Chapter **??**.

The velocity estimation accuracy strongly resembles the position estimation accuracy in its fluctuation behaviour, which is not surprising due to the stepwise linear relation between the two parameters. From a comparison of Figure 4.11 with 4.5 the similarity between the two errors of the estimated parameters can be seen.

The quaternions that indicate the performance in the attitude estimation were converted to Euler angles in the performance assessment for more convenient representation in the comparison. Nevertheless, the quaternion error of test case 11 is shown in Figure 4.12.
While the quaternions almost immediately after the start of the test run settle and fluctuate around the target attitude, there are several outliers. It is postulated that phase shifts could offset the error of spin rate and attitude in such an instantaneous manor as can be observed in Figure 4.12. This occurs in all filter modes and was attributed to the data analysis approach rather than to errors in the filters. The instantaneous occurrences of such errors where corrected for the same way as outliers by neglecting them for further assessment.

The estimation spin rate error, shown in Figure 4.13 for test case 12, is overall very small and fluctuates about the target state without extraordinary outliers or deviations. This is representative of the other test

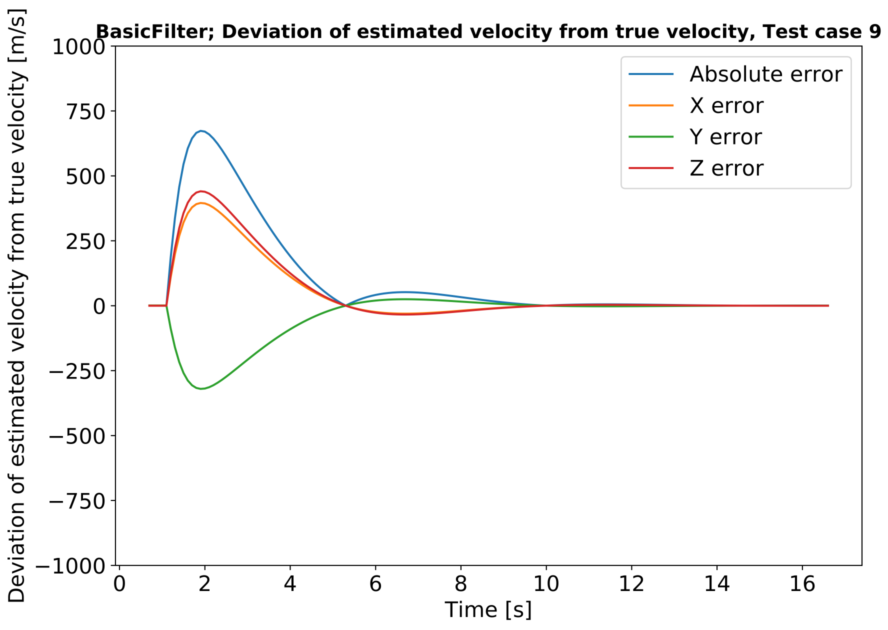


Figure 4.11: Velocity estimation error of EKF estimate in test case 9; very similar performance to position estimate

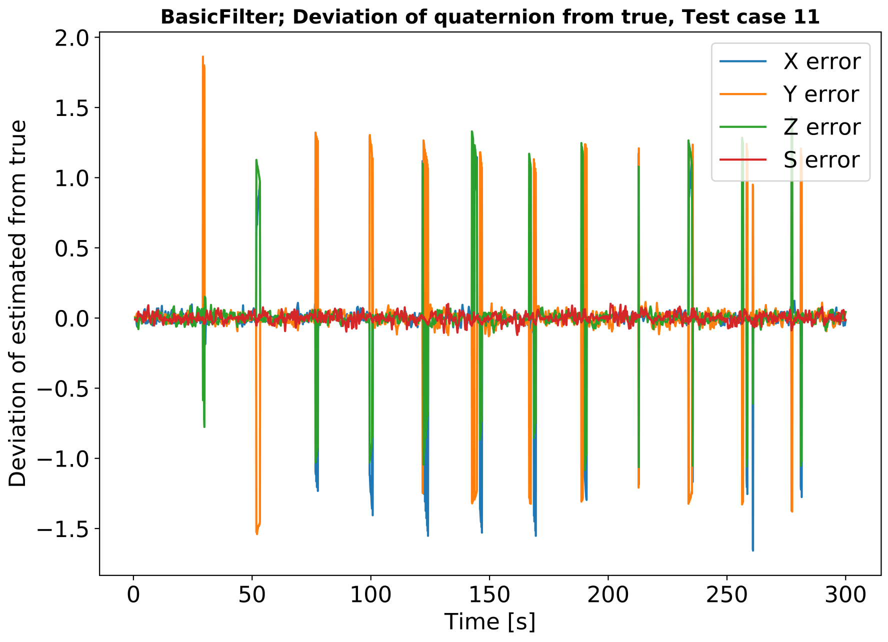


Figure 4.12: Quaternion estimation error of EKF, test case 11; occasional instantaneous outliers are partly attributed to phase shifts between actual and estimated state

cases.

4.5.2. Comparison of EKFS and UKF performance to EKF

This section presents the performance of EKFS and UKF in comparison with the original EKF. The individual performance values are not presented here, but rather the quantitative improvement or worsening of the filter performance with respect to the EKF.

EKFS performance compared to EKF

The position estimation performance of the EKFS with respect to the EKF is shown in Table 4.4. The total

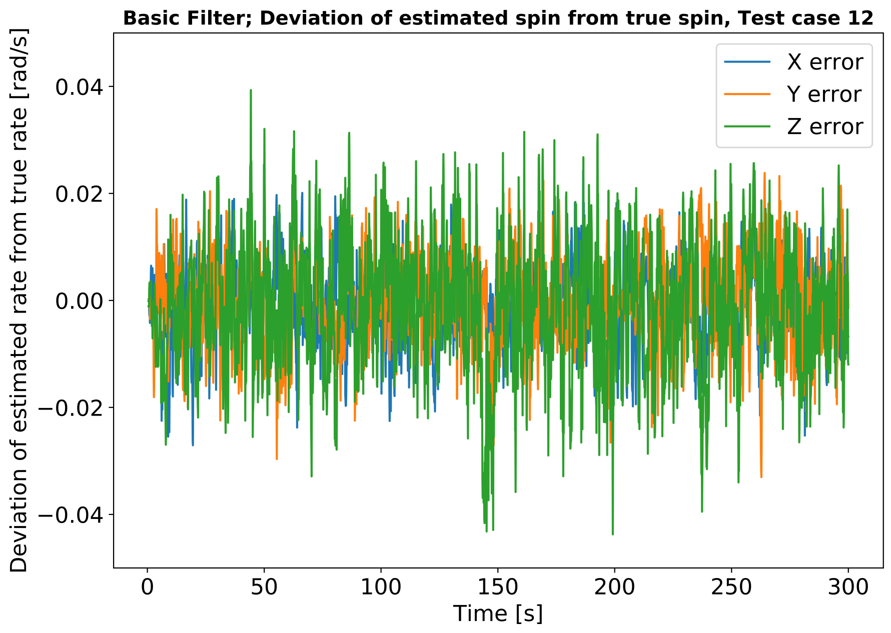


Figure 4.13: Spin rate estimation error of EKF, test case 12; small error fluctuating about the target state

error column is not needed since the % improvement would be the same as for the mean error. The table shows the parameter improvement (positive value) or worsening (negative value). Several insights can be gained. Firstly, the EKFS outperforms the EKF in position estimation by an average of 4.6% excluding test cases 1, 4, 7 and 8 for the reasons mentioned before. This increases to an average improvement of 6% in the shortened case without swing in period, showing that the converged estimate of the EKFS is more accurate than the EKF, also in comparison to its own swing in period. However, the median does not on average drop closer to the true state (worsens by 2%), meaning that while the average of the tail section error distribution lowers, this does not represent all points in the distribution.
The biggest improvement can be seen in the shortening of the time to reach the same convergence limit as the EKF. The EKFS on average converges 16.6% faster to within the same bounds as the EKF, meaning that if the same accuracy as for the EKF should be achieved faster the EKFS would be superior.

Figure 4.14 shows the absolute position error of all three filter modes in test case 9 for the first 30 seconds of the test run. It can clearly be seen that the EKFS and UKF error in position estimation is lower than for the EKF. The convergence range of the EKF is reached faster as a result.

The other state parameters are assessed the same way as the position. The average improvement relative to the EKF are shown in Table 4.5. From this data assessment it is clear that the EKFS only outperforms the EKF in the estimation of position and velocity, not in the estimation of attitude and spin rate. This is not only the case in a few select test cases but across the entirety of the tested spectrum. The possible theoretical origins of this observation, which are rooted in the approach to taking averages of quaternions, are discussed further down.
Nevertheless, it is necessary to mention that the position and velocity estimation improve when using the EKFS, as was expected based on the documentation of the EKFS and the theoretical background.

The poorer performance may also be due to the propagation method in the simulation test matching the propagation in the EKF. Thus, the intermediate smoothing step in the EKFS may actually introduce errors in the simulation test while it would improve the performance in real-life conditions.

The absolute velocity error for all three filter modes is shown in Figure 4.15. Compared to Filter 4.14 it can be observed that the relative distribution of the velocity error matches the position error, which was expected since the position linearly relates to the velocity, which, especially when the filter estimates have not converged yet, means that the trend in both graphs is the same. Once filters have converged, however,

Table 4.4: EKFS simulation test position estimation performance compared to EKF

Test case	Mean error improvement [%]	Mean error improvement (s) [%]	Median error improvement (s) [%]	Time to conv. improvement [%]
1	50.1 %	73.3 %	12.7 %	48.1 %
2	0 %	0 %	0 %	-1.2 %
3	6.0 %	0. %	-1.2 %	26.3 %
4	50.1 %	73.3 %	13.0 %	42.6 %
5	0 %	0 %	0 %	-1.2 %
6	6.1 %	1.7 %	-1.3 %	68.0 %
7	0 %	0 %	0 %	-0.4 %
8	0 %	0 %	0 %	-0.4 %
9	7.8 %	17.9 %	-2.1 %	18.4 %
10	4.4 %	14.6 %	-3.7 %	18.8 %
11	6.8 %	15.2 %	-3.1 %	6.7 %
12	9.7 %	8.4 %	-5.2 %	21.1 %
13	7.1 %	8.1 %	-3.3 %	20.0 %
14	8.4 %	6.7 %	-4.2 %	24.4 %
15	0 %	0 %	0 %	-1.2 %
16	0 %	0 %	0 %	-1.2 %

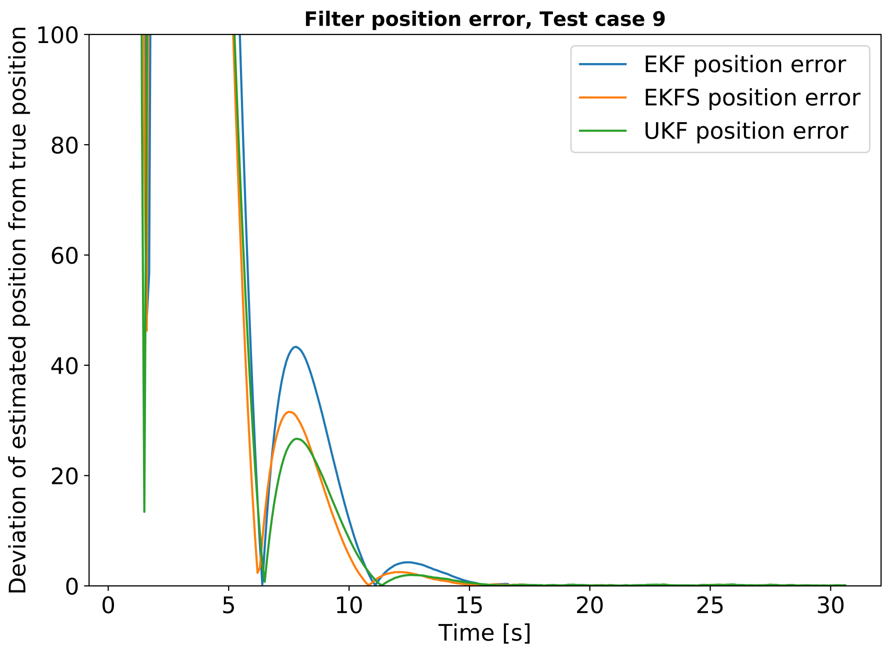


Figure 4.14: Absolute position estimation error of EKF, EKFS and UKF, test case 9

Table 4.5: EKFS remaining state parameter performance compared to EKF

Parameter	Mean error improvement [%]
Velocity	9.7 %
Attitude angle x	-16.2 %
Attitude angle y	-14.7 %
Attitude angle z	-16.8 %
Rate x axis	-19.3 %
Rate y axis	-21.7 %
Rate z axis	-31.7 %

this will not be the same anymore, since different filters will estimate velocities around the true target state differently and thus cause more or fewer fluctuations of the position estimate error, depending on how far off the velocity estimate is.

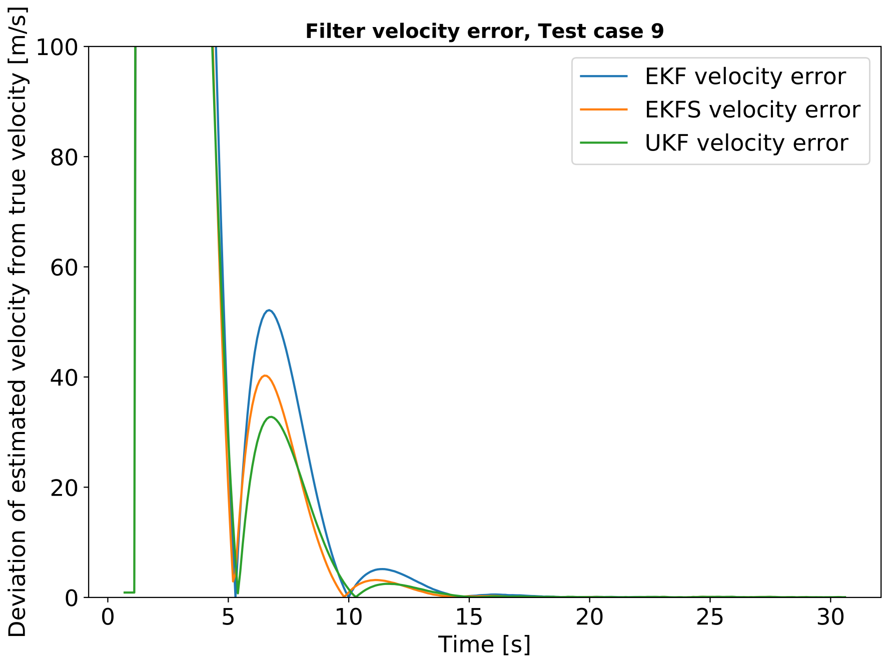


Figure 4.15: Absolute velocity estimation error of EKF, EKFS and UKF, test case 9; the relative error distribution is the same as for the position, as expected

Figure 4.16 shows the estimation error of the quaternion from the EKFS for test case 11, the same test case as in Figure 4.12 where the EKF performance is shown. As can be seen, the error is again small apart from obvious instantaneous error spikes. The distribution of these spikes is similar, suggesting that the filters estimate a wrong quaternion at similar intervals or that the simulation test delivers wrong true states to which the quaternion estimates are compared. Apart from these spikes, the EKFS estimates the quaternions on average slightly worse than the EKF. Again, this is discussed in more detail below.

The spin rate estimation performed by the EKFS, show representatively for test case 12 in Figure 4.17, was analysed to be slightly worse than for the EKF. Comparing the figure with Figure 4.13 it can be seen that the EKFS estimate fluctuates about the target the same way as the estimate of the EKF, with the difference that the fluctuation arches have higher maxima throughout the entire test run. This explains the poorer performance.

UKF performance compared to EKF

The UKFs performance improvement to the EKF is shown in Table 4.6. From the data presented above it can be seen that the UKF performs better than the EKF and even the EKFS in the position estimate error, in particular in the test cases where delayed measurements are used (9 through 14), as was already seen in Figure 4.14. However, the relative improvement to the tail section is not as great as in the case of the EKFS, meaning that the improvement is more evenly split over the swing-in period and the tail section.
This general behaviour matches the expectation for the UKF. Since the propagation of individual sigma vectors should theoretically result in more accurate state estimations in a non-linear system, it is not surprising that the estimation is better and the convergence faster, since it is expected that at each iteration step a more accurate result can be achieved than for the other filters. Thus, faster convergence is expected.

Furthermore, the median error does not improve significantly over all test cases and even worsens in some. Lastly, the time to convergence to within the same convergence bounds as in case of the EKF improves

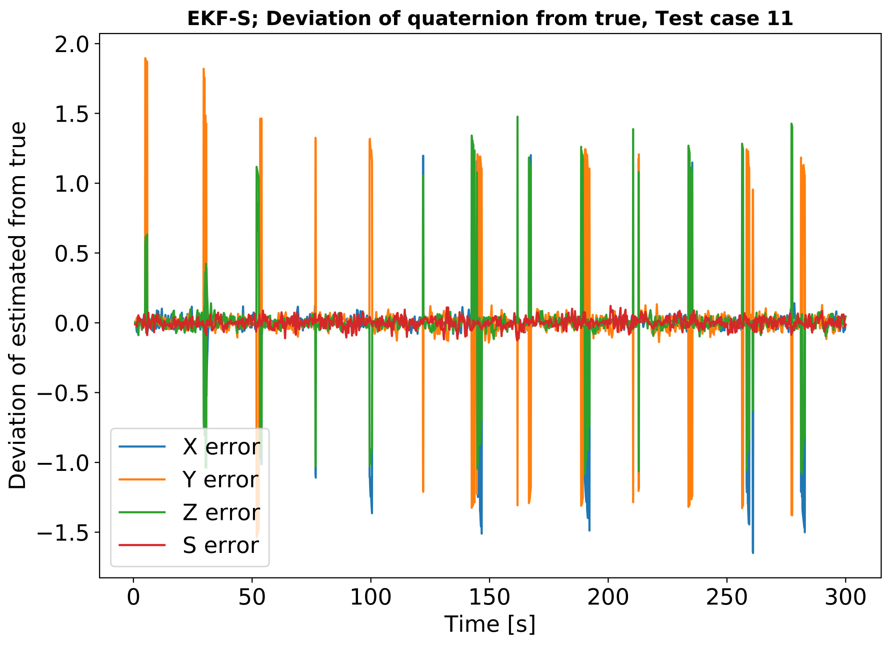


Figure 4.16: Quaternion estimation error of EKFS, test case 11; as with the EKF, outliers or phase shifts cause error spikes to an otherwise accurate estimate

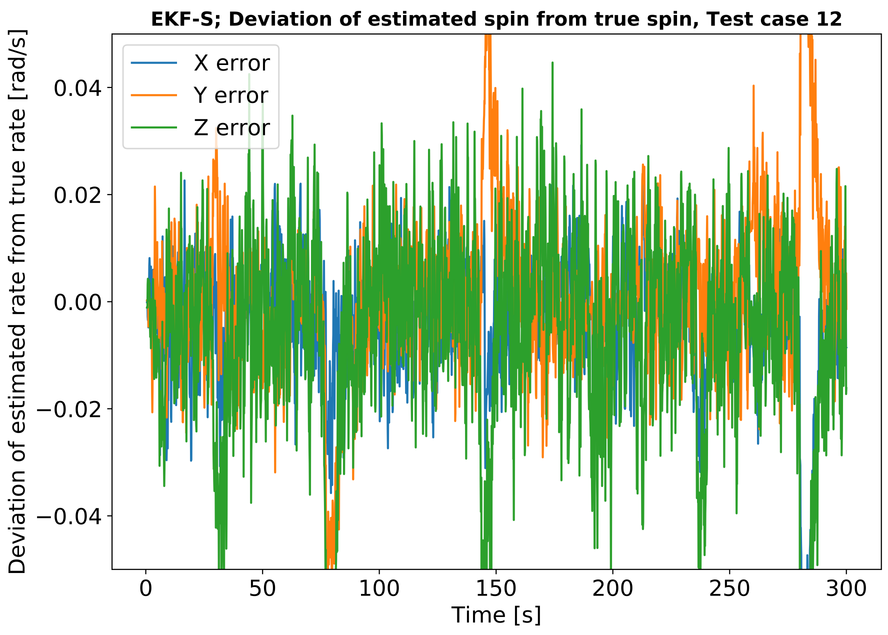


Figure 4.17: Spin rate estimation error of EKFS, test case 12; larger fluctuations than for the EKF estimate are observed and cause a larger error on average throughout the test run

over the more complex test cases with delayed measurements but significantly worsens in the test cases with the same filter and measurement update frequencies. This is primarily attributed to a couple of outliers in these test cases, since the median in all of these test cases is significantly lower than the mean position estimate error. Registering outliers in the UKF position estimate error that are beyond the position error of the converged section in the EKF would thus delay the time at which the UKF estimate is considered to be converged within the same bounds as the EKF. An example can be seen in test case 5, shown in Figure 4.18, illustrates this. A late outlier can be seen after 260 seconds into the run, delaying the registered time to converge to within the EKF's convergence limit substantially.

Table 4.6: UKF simulation test performance compared to EKF

Test case	Mean error improvement [%]	Mean error improvement (s) [%]	Median error improvement (s) [%]	Time to conv. improvement [%]
1	48.0 %	55.9 %	7.4 %	NA %
2	-3.9 %	-131.0 %	-25.0 %	-2687.1 %
3	2.6 %	-15.0 %	-1.0 %	22.5 %
4	40.3 %	41.2 %	-14.8 %	-1912.8 %
5	-3.1 %	-107.2 %	-14.2 %	-3031.8 %
6	3.6 %	-1.0 %	0.1 %	68.0 %
7	0 %	0 %	-33.2 %	6.3 %
8	0 %	0 %	-15.5 %	-867.2 %
9	19.4 %	16.0 %	1.3 %	15.2 %
10	19.2 %	19.1 %	0.2 %	15.3 %
11	20.1 %	18.5 %	4.7 %	56.7 %
12	13.5 %	5.5 %	-1.2 %	17.6 %
13	13.7 %	9.9 %	4.3 %	16.0 %
14	13.3 %	6.7 %	2.5 %	21.8 %
15	-1.7 %	-62.2 %	-18.9 %	18.8 %
16	-4.4 %	-140.8 %	-11.8 %	-2901.2 %

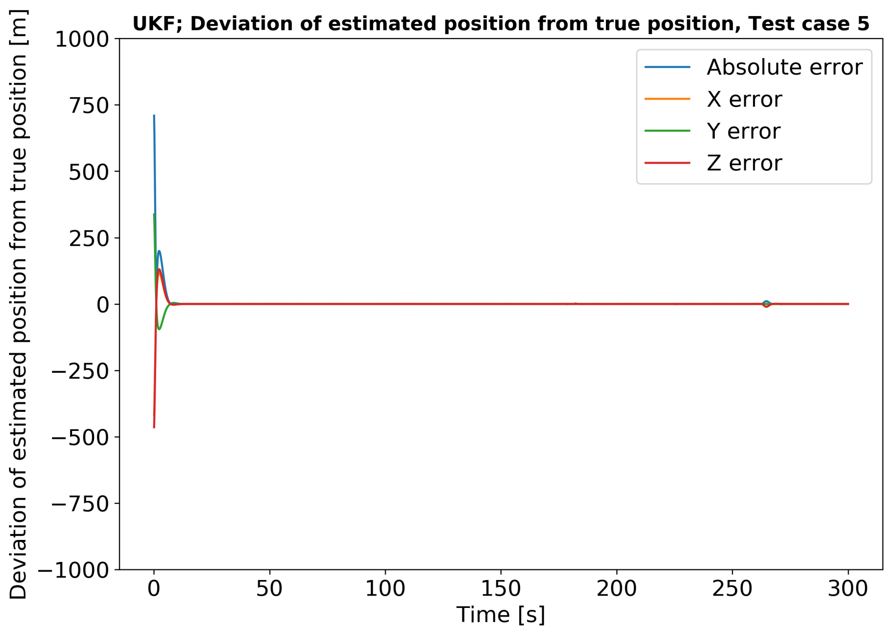


Figure 4.18: Position estimation error of the UKF in test case 5 shows a small position outlier late in the test run, delaying the perceived time to convergence

For the relevant test cases presented earlier (excluding 1, 4, 7 and 8) the average position estimation improves by 7.7%. In the tail section alone, however, this estimation worsens by 31.8% in comparison to the EKF, which, however, is only due to the test cases with the same filter and measurement update times (see the example of test case 5 above, where outliers disturb the estimate). Only focusing on the cases where measurement and filter update times are different the UKF outperforms the EKF by 12.6% in the tail section. The median error in the tail section lies 4.9% further away from the true target position than in the case of the EKF. Only excluding test cases 1, 4, 7 and 8 from the assessment the time to converge to the same limits as the EKF is on average 6.97 times as long as for the EKF. However, only focusing on test case 9 through 14, the UKF outperforms the EKF in this time by 23.8%. The example of test case 12, shown in Figure 4.19, illustrates this, where no obvious late outlier can be observed. This performs both the overall error performance as well as

Table 4.7: UKF remaining state parameter performance compared to EKF

Parameter	Mean error improvement [%]
Velocity	-20.8 %
Attitude angle x	-1114.2 %
Attitude angle y	-1646.0 %
Attitude angle z	-1041.7 %
Rate x axis	14.4 %
Rate y axis	-24.5 %
Rate z axis	-27.1 %

the time to convergence with respect to the EKF convergence limit.

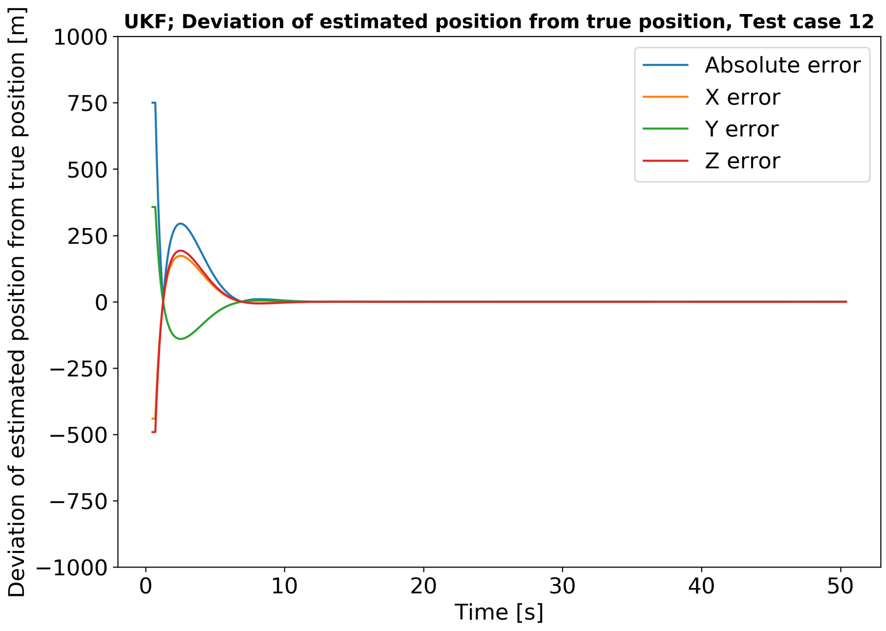


Figure 4.19: Position estimation error of the UKF in test case 12 does not show the small position outlier observed in test case 5, improving the performance compared to the EKF

This leads to two conclusions about the UKF. It can estimate the position of the target more accurately on average than the EKF and the EKFS, but it features occasional outliers and has a wider spread around the target than the EKF. Also, the UKF performs considerably better when filter update times and measurement update times differ.

Since the UKF is the only tested filter that propagates multiple possible points and since the simulation test only propagates the average of a possible state distribution, their propagation methods differ considerably more than for the EKF and the simulation performance test. Theoretically, this can explain the performance difference, especially in cases where measurement disturbances are not as great as when measurements only occur every couple of iterations and cause considerable estimation changes.
It is thus concluded that a different propagation method is needed for the simulation performance test than the one that is used in any of the filters.

Again the remaining state parameters are compared to the EKF performance as before. The performance comparison can be seen in Table 4.7.
Similar to the EKFS, the UKF cannot match the performance of the EKF in the estimation of the attitude and spin rate. In addition, the velocity estimation is also slightly worse. However, this is again purely due to the test cases where the filter and measurement update times match. In the other test cases the UKF performs 23.6% better than the EKF. An example is shown in Figure 4.20

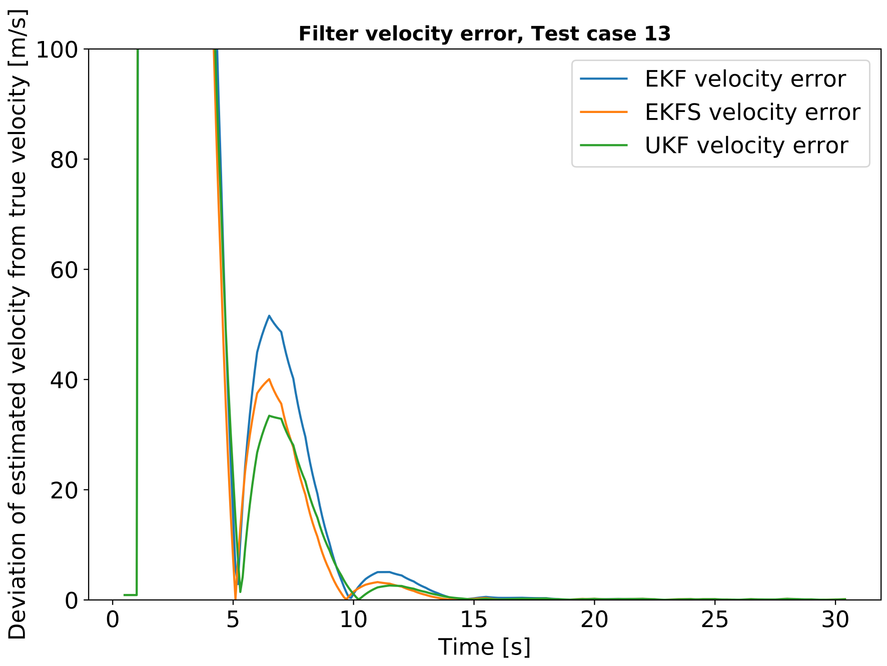


Figure 4.20: Comparative velocity estimation error of all filter modes for test case 13; the UKF outperforms both other filter modes in test cases 9 through 14

The most obvious difference in the performance of EKF and UKF is the decrease in UKF attitude estimation accuracy. This has several reasons: the EKF estimates the attitude very accurately (average error of 0.08 deg in x and y and 0.09 deg in z). Comparatively, a fine but not as good performance by the UKF is regarded as very poor (1 deg average error in x, 1.5 deg error in y and 1 deg error in z).
Additionally, several test cases show that after several seconds into the test run the UKF attitude estimation diverges from the simulated target attitude and estimated target spin rates. This is where the more complicated behaviour of quaternions may have an effect on the performance. The UKF propagation is based on weighted state averages. Since quaternions show their own behaviour and cannot simple be assessed by taking an average of all the individual quaternions, the UKF approach may not work with the kind of state under investigation. Further research is required to assess this behaviour since it was identified too late in the project to address it properly.
This behaviour is showcased for test case 11 and 12 in Figures 4.21 and 4.22, respectively. For test case 11, it can be seen that the quaternions seem to fluctuate around the target quickly after the start of the test run, which registers increased average errors across the test run. Test case 12 shows a behaviour similar to the one observed for EKF and EKFS (compare with Figures 4.12 and 4.16). This proves that in this case, this is not an issue related to delayed measurements but it occurs across test cases from all categories.
This was not expected. However, it was found out after the research phase that taking the weighted average of quaternions may not work the way it was intended.

The problem of computing average quaternions:
The problem that arises in the UKF (and potentially also in the EKFS when the smoothing state is computed from the average of two states) can be explained theoretically. Since this was only discovered after the testing phase ended it is not implemented in this project. The intention of the UKF is not to take the average quaternion but to compute an average attitude. The quaternion used for showing an attitude can be imagined to contain both information on the orthogonal rotation as well as on the orientation of the axis of rotation. Taking the average of this information for multiple quaternions does not maintain the orthogonality of one quaternion and the result does not represent the average attitude information. Thus, as described in [21], different approaches are required to determine the quaternion that represents the average quaternion. Therefore, the approach taken for the UKF of simply taking the average of a state vector does not truly represent the average attitude of the different state vectors and a different approach needs to be taken. This problem could be fixed in future problems by consulting [21] for example.

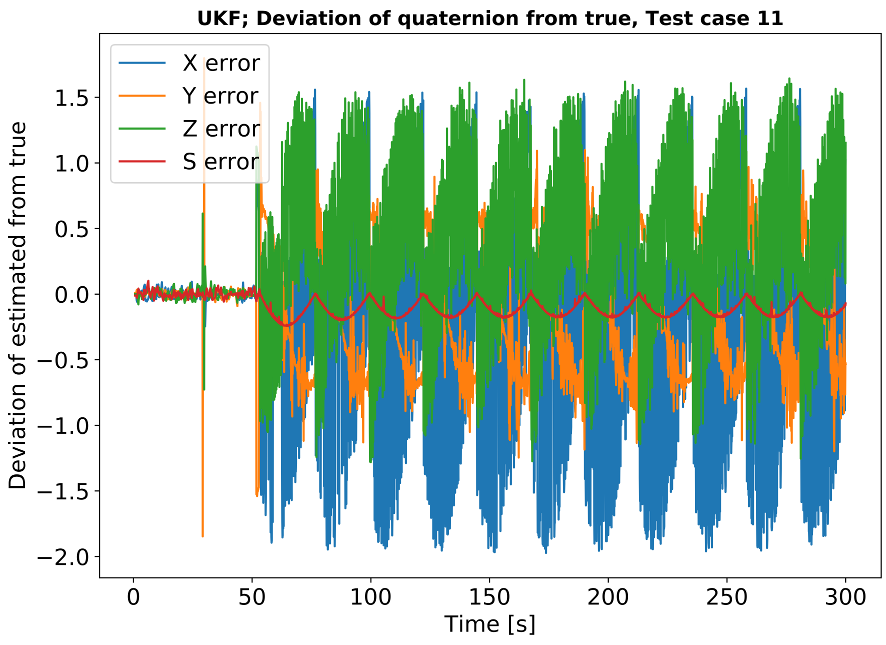


Figure 4.21: Quaternion error estimate of UKF, test case 11; large error fluctuations start briefly after the beginning of the test run

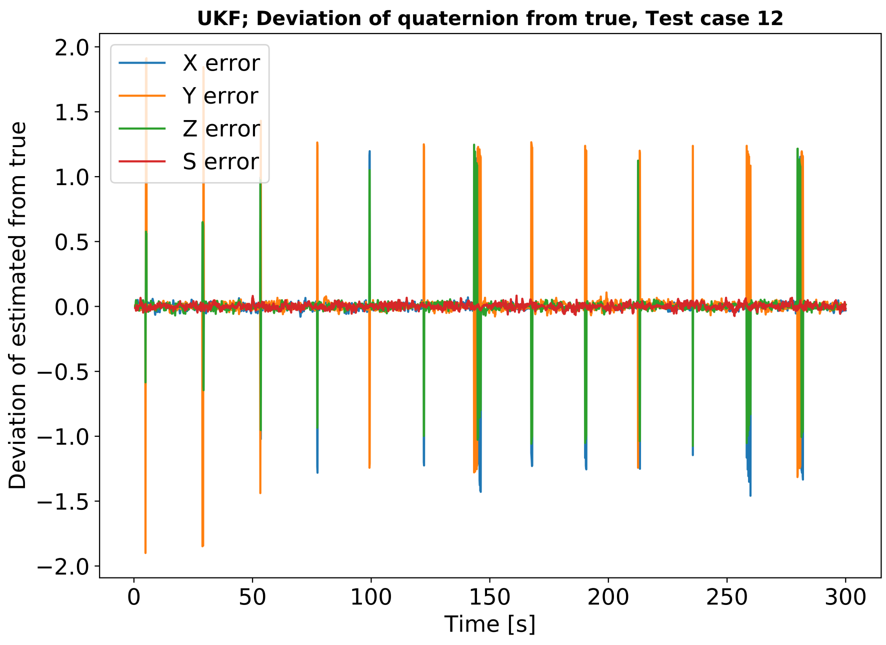


Figure 4.22: Quaternion error estimate of UKF, test case 12; similar performance as for EKF and UKF, no large error fluctuations

Since the EKFS also includes a step where the average of two state vectors is computed (to find the smoothing step), this may also be a problem here. However, it is not yet determined why the problem does not seem to be as grave.
It remains to be investigated as well why these problems do not occur consistently across all test cases.

Shortcomings of the unit test approach for verification and validation:
It can be said that the performance of the position and velocity estimation works well for all filters and is verified and validated, as expected based on the performance of the filters in their individual unit tests. Within the bounds of the simulation test, the performance for these parameters is also validated, as it could be shown that the position and velocity estimation performs as required when estimating a simulated target satellite

state. However, this verification and validation based on the simulation performance test is not possible for the attitude estimation of the UKF. This highlights a wider problem with the verification and validation method: the verification is primarily performed based on the performance of the individual filter functions in their respective unit tests (where all filter functionalities work and are thus verified), and the unit tests that test the interaction of the individual functionalities to check that the states are computed correctly using specific inputs. All these tests are passed for all state parameters, which suggest that the filters are fully verified. The simulation performance test was used as another verification to show that the state estimation works given propagating inputs, and as a validation for the filter performance. However, as was seen, the UKF attitude estimation does not work when performing filter updates over a multitude of subsequent iterations and can thus not be validated. A more stringent approach for verification in the unit tests, however, could have captured this earlier since the problem of non-orthogonality of the result of averaged quaternions and longer testing over multiple test runs could have captured this in the verification step already.

4.5.3. Assessment of computational resources

The simulation performance test is run step wise, updating the filter time and the measurement time by 0.1 seconds at every iteration step and passing the same input information to each filter mode. This enables to assess the computational requirements of each filter via the time it takes to run a filter iteration when executing all filters on the same system (as was done). The average time to execute a filter for all test cases is presented below. Furthermore, the runtime increase is presented in Table 4.8. All filters can be executed easily

Table 4.8: Execution time of filters in simulation test

Parameter	EKF	EKFS	UKF
Avg execution time	6.99e-5	8.89e-5	22.2e-9
% of EKF execution time	-	127 %	318 %

at 10 Hz on a conventional desktop computer. As can be seen, the execution time of the EKFS only increases by 27% compared to the EKF, while the UKF time increases by 218%. This, however, is not surprising since the UKF needs to propagate all sigma points and thus computation time is expected to increase. As long as filters can be run at 10 Hz using the relevant system this shouldn't be a concern. Nevertheless, this insight also shows that fewer resources could potentially be freed up for other applications in a system using the new filters, in case that other applications or capabilities should be added to the system. This needs to be kept in mind throughout a system design process.

4.6. Judging simulated filter performance against performance criteria

Having collected performance parameters from the different test cases, the performance of the individual filters can be judged against the quantitative criteria presented in Section 1.9.10.

It was seen for the UKF that the performance in the cases with delayed measurements is generally better. Furthermore, it was determined that the performance in such test cases is more representative of real-world scenarios. Past tests and real-world applications have shown that such test cases are more realistic since image processing and other intermediate steps delay measurement input to the filter. This is why the comparison of the performances in test cases 9 through 14 is considered when judging the performance of the filters against the criteria. For these test cases, the comparative performance of the EKFS and the UKF from the simulation test with respect to the EKF is shown in Table 4.9.

Table 4.9: Filter alternatives - performance in test cases with measurement delays

Parameter	EKFS	UKF
Position mean error	11.8 %	12.6 %
Velocity mean error	14.4 %	23.6 %
Attitude error x	-20.7 %	-757 %
Attitude error y	-18.4 %	-1176 %
Attitude error z	-21.8 %	-727 %

Following the performance criteria rating defined in Section 4.1, each filter is rated whether it performs

Table 4.10: Rating of filters against performance criteria based on simulation test results

Criterion	Weight	EKF score	EKFS score	UKF score
Accuracy	5	0	+1	+1
Time to settle	3	0	+1	+2
Stability	4	0	0	0
Bias	2	0	0	0
Complexity	3	0	-2	-2
Overall	NA	0	+2	+5

better or worse than the EKF. The underlying data is the test results data from the simulation performance test using the representative test case results shown in Table 4.9.

The filter modes are rated in each category as follows:

Accuracy: The position estimate, velocity estimate and the average of the attitude estimates are each rated. The rounded average of the three ratings is taken for the accuracy parameter.

Time to settle: For the simulation performance test the average percentage time improvement of the new filters to arrive within the same convergence boundary as the EKF is rated.

Stability: No differences could be observed in the simulation performance test cases, including the test cases featuring outliers.

Bias: No bias was observed in the filters.

Complexity: A clear difference in run time could be seen in the simulation performance test, where run times for the filter alternatives went up as expected.

The scoring of the filters is shown in Table 4.10.

From the filter performance scoring it can be seen that the UKF slightly outperforms both the EKF and the EKFS. The positive rating in estimation accuracy despite the quaternion problems observed during the test data analysis stems from the positive rating in both position and velocity assessment. It is seen that the rating criteria may thus not yield a fully fair and representative judging of the filters since a medium improvement in two parameters can outweigh the very poor performance in another parameter.

4.7. Conclusions from analytical performance assessment

From the results presented in this chapter several conclusions can be drawn.

A simulation performance test was designed that aims to reflect real-life testing conditions by propagating simulated true satellite states and supplying navigation filters with information to estimate the state of a target position. The purpose of this test was twofold: to demonstrate that the filters estimate the state of the target satellite and to show the quality of the state estimation of a filter in comparison with other filters.
Thus, the simulation performance test can be used to verify the filter working (it shows that a state is estimated that has the expected outcomes and it can incorporate the relevant inputs) and to validate the performance in a simple simulated mission scenario. The verification for the individual filter functionalities is concluded since all filters deliver the expected output parameters and the separate filter functionalities have been shown to work. Nevertheless, the deviation of the UKF attitude estimation highlights a problem in the wider verification approach: the verification methods applied throughout development cannot be used to assess the proper working of the filter functionalities over a large number of iterations. This problem was identified too late in the research phase of the project to address it and correct for it.
The validation could not entirely be concluded since the UKF shows problems in the estimation of the attitude parameters, which means that it cannot be used for missions that rely on the accurate estimation of the attitude. These problems could be explained theoretically through the fact that it is not possible to simply take weighted averages of a quaternion, as was done in the UKF for many state vectors and in an intermediate step in the EKFS. This was only realised after the research phase was finished and is thus not implemented in

this project. Nevertheless, both newly implemented filters are taken forward into the hardware-in-the-loop test, which is used as a further validation test under more realistic mission testing conditions. This is done since it should yield more insight on the quality of the estimation on the other state parameters. Also, this should deliver further insights on filter performance when using real-life sensor inputs and verify the simulation testing performance. Furthermore, it will show whether the UKF quaternion estimation deviation is a filter problem or a simulation testing problem.

In addition to these conclusions on the verification and validation of the filters, several conclusions of the performance in the simulation test can be drawn.
The simulation performance test itself was shown to work well and 16 test cases were simulated of which 12 were used for data assessment. The EKFS and UKF outperform the original EKF in the position estimate. The EKFS also performs better in the velocity estimation. Both new filter modes perform poorly in the estimation of the attitude and spin rate. Especially the UKF lacks behind the other two filters.

The UKF performs considerably better than the EKF in test cases where the measurement is delayed or withheld. The performance drops comparatively when the update frequencies match. This was concluded not to be a problem since in real-life missions measurements would always be delayed when passed to the filters due to image processing steps taking time. Thus, these test cases were also used as primary sources of data for the eventual judging of filter performance.

The newly designed assessment method where the convergence is determined mathematically works well for certain test cases but could not be applied across all test cases in a meaningful way. However, when determining the quality of convergence for the EKF and when this convergence was achieved in a test run, it could be seen that the new filter modes reach the convergence bounds of the EKF considerably faster. The only exception were the test cases of the UKF where update frequencies of filter and measurements match. It was thus concluded that the new filter modes have faster performance to within more accurate bounds. This assessment was primarily performed for the position estimation since it was deemed the most critical parameter for close proximity operation without docking.

Since both the EKFS and the UKF use slightly different propagation methods than the EKF, and since the EKF shares its propagation method with the simulation performance test it was concluded that a different method than in any of the filters should be applied in the simulation performance test. This would allows for a fairer comparison and may yield more representative data. However, developing a new orbital propagation method was beyond the scope of this project and is left for future research.

5 Hardware-in-the-loop test and results

Having tested that the filter modes return sensible results and converge when given simulated inputs that represent real-life conditions they can be implemented in a hardware-in-the-loop testing facility to confirm their performance and investigate how they perform with real sensor inputs. The facility used is the EPOS 2.0 facility at DLR Oberpfaffenhofen.

This chapter explains the need for hardware-in-the-loop testing and the EPOS 2.0 testing facility (5.1). Thereafter, the filter implementation and test setup is outlined in 5.2. The test execution and results are explained in 5.3 and 5.4. Having found the test results, the filters are judged against the performance criteria in a similar way to the results from the simulation test, shown in 5.5. At this point, when both tests have been discussed and judged, the differences in testing and possible improvements in future projects are presented in 5.6, followed by a more detailed discussion of the observations and correlations, and lack thereof, of the results from both tests in 5.7. The insights from hardware testing are briefly summarised in 5.8, after which the research questions are addressed in 5.9. Finally, the value of the results of the project is discussed in 5.10.

5.1. Background for EPOS 2.0 simulator testing

The simulation test is performed an all 3 filter modes throughout their development and implementation phases. It is performed before the hardware-in-the-loop test. The need for a simulation performance test has been discussed in Section 1.9.6.

The European Proximity Operations Simulator (EPOS) 2.0 facility is a ground based robotic testing facility. It was designed to perform close proximity (0 - 25 metre approach) late stage approach simulation for two spacecraft, which are represented by two KUKA robots. The facility is primarily used for testing in rendezvous and docking scenarios.
The robots (KUKA KR100HA and KR240) are mounted on a rail facing each other. The KR100HA (servicer) can move in a straight line on the rail towards the other robot or away from it. A satellite mock-up on the stationary KR240 (target) can be moved up, down and sideways and spin around its axis, which simulates movement in plane, pitch and yaw, and, from the perspective of the servicer represents approach conditions and fly-arounds. These scenarios are captured by sensors mounted on a board on the servicer robot, such as cameras. The facility is seen in Figures 5.1 and 5.2, where the view from the servicer to the target, and from the target to the servicer can be seen, respectively.

The great advantage of the EPOS 2.0 facility is that simulations can be combined with real-world sensor inputs. While it is impossible to represent orbital dynamics fully realistic on Earth, these can be simulated with satisfying accuracy and passed to the robots which can then move relative to each other based on the orbital parameters that they are given. This is based on a simulation model of orbital parameters. The orbital model is a classic two body model and the satellites are simulated with unperturbed Kepler orbits. Influences from sun, moon, solar radiation pressure and other factors were not modelled in the EPOS 2.0 test series performed as part of this project. Nevertheless, the facility has the capability to run more complex models including further factors.

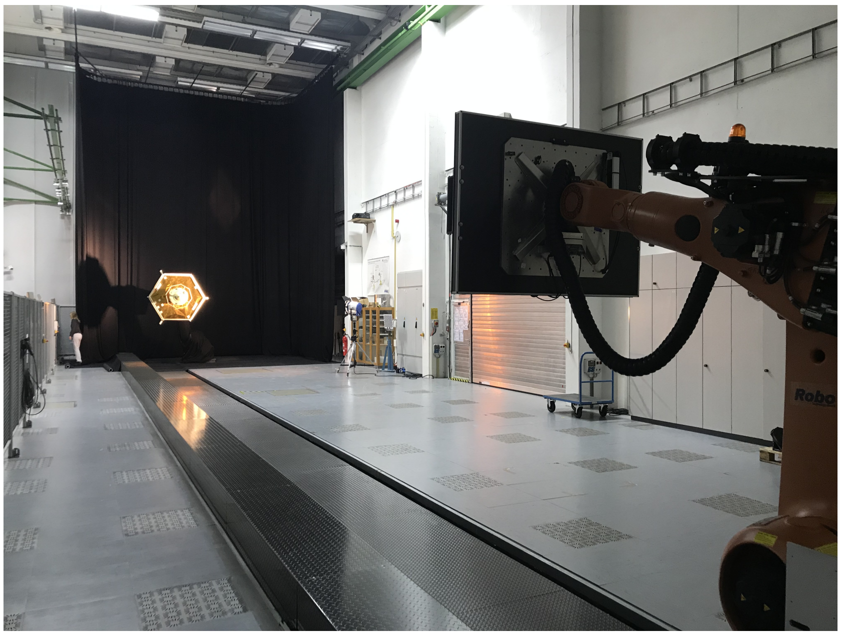


Figure 5.1: EPOS 2.0 facility, view from the servicer towards the target satellite

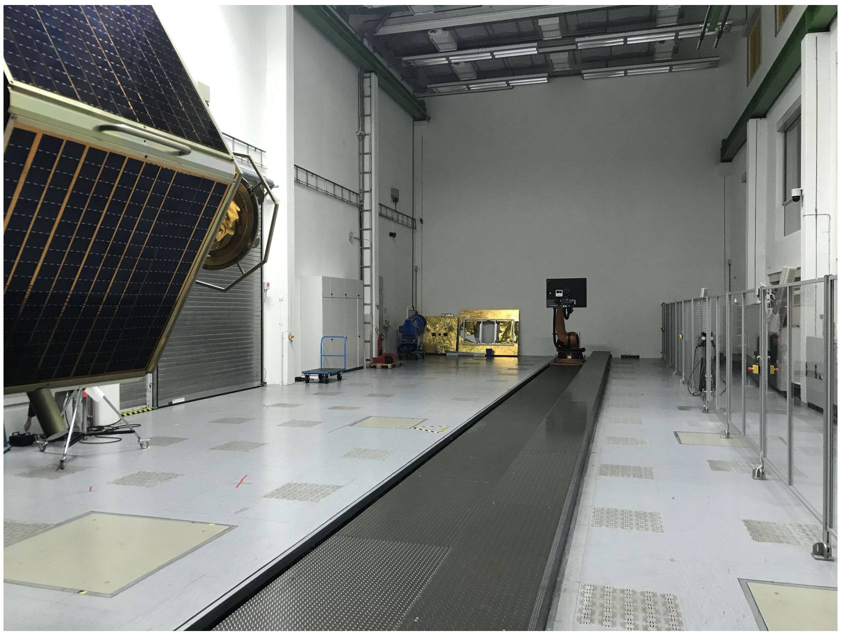


Figure 5.2: EPOS 2.0 facility, view from the target towards the servicer

On the other side, sensory inputs that are gained in satellite operation are very difficult to simulate accurately. However, since real sensors can be equipped in the facility this information can be gathered from actual sensors and can be combined with the orbital simulation model.

Sensor information is not passed directly to the robots but, in order to represent satellite operation in space more accurately, the telemetry is passed to a sensor interface and data acquisition system from where it is passed into the GNC system and image processing system. This represents the ground system in a normal satellite operation. It is visualised in the image from Benninghoff et al. (Source:[4]) in Figure 5.3. Image processing is performed (which takes time and causes measurement delays, as was mentioned before) and the information is passed to the navigation filters. The output from the orbital model and the GNC outputs are then passed on to the EPOS control system where facility monitoring and the dynamic motion simulation take place. The ground system can send commands to the servicer satellite to perform movements, otherwise the robots operate idly based on the inputs they get from the EPOS control system.

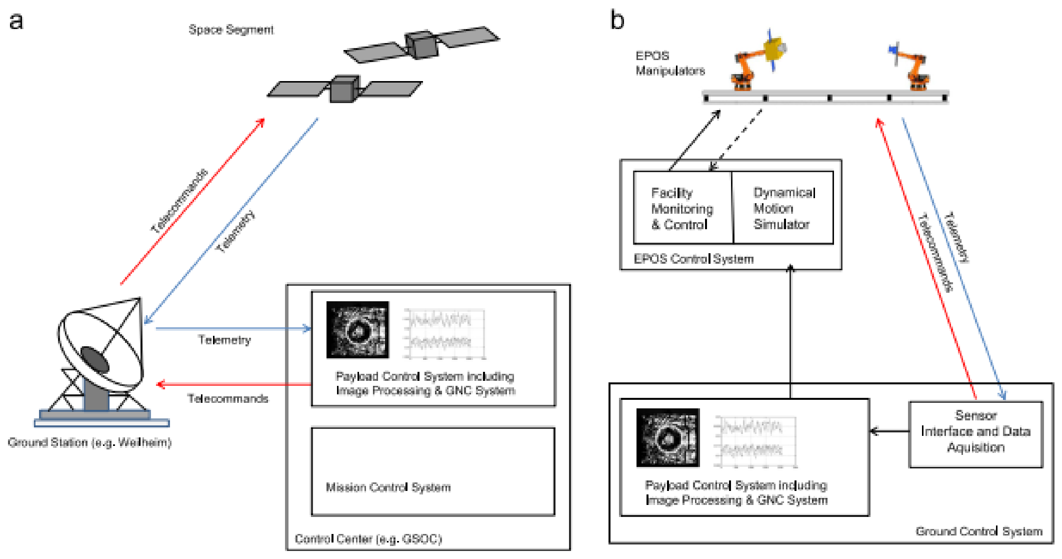

a
Space Segment
Telecommands
Telemetry
Telemetry
Telecommands
Payload Control System including Image Processing & GNC System
Mission Control System
Control Center (e.g. GSOC)
Ground Station (e.g. Weilheim)
b
EPOS Manipulators
Facility Monitoring & Control
Dynamical Motion Simulator
EPOS Control System
Telemetry
Telecommands
Sensor Interface and Data Aquisition
Payload Control System including Image Processing & GNC System
Ground Control System

Figure 5.3: Representation of system architecture; left: actual satellite communication concept; right: representation of system in EPOS 2.0 facility; Source: [4]

Using the hardware-in-the-loop facility for verification and validation:
As was discussed before, the EPOS 2.0 facility tests should be used to validate the working of the filters, ideally in a similar way as the simulation test does in Chapter 4. However, upon closer inspection it becomes obvious that there are several essential differences between the facility and the simulation performance test that cannot be easily overcome. The EPOS 2.0 facility uses a different orbital model and a different initialisation approach than the simulation performance test. Furthermore, it uses real measurement inputs which cannot be fed to the simulation performance test since the measurements are linked to the actual satellite positions. While the actual satellite positions could be extracted and fed to the filter, the satellites are, when fully operational, controlled by the filter that is currently running as well as by a human controller. This requires two things for future use of the simulation performance test: The extraction of the real life data and force inputs from the EPOS 2.0 facility, as well as a re-structuring of the simulation performance test to not only adjust the propagation model used but to also access the real-life data passed from the hardware facility. This is beyond the scope of the project at hand.
Nevertheless, the EPOS 2.0 facility test results should indicate similar trends as were observed in the simulation performance test. They may not deliver exactly the same numerical results but can be used to confirm general behaviour of the filters observed earlier.

5.2. Test setup for filter testing

This section describes the EPOS 2.0 test run initialisation and the scenarios tested in the facility. Furthermore, the data handling and test execution plans are explained.

5.2.1. Facility setup and filter implementation

The EPOS 2.0 facility is initialised by setting up the robots and the orbital model.[3] The GNC system is started using a configuration file used to initialise the relevant parameters. For the purpose of testing several filters the type of filter mode was included in this configuration file.

The filter hierarchy in the GNC system is structured as such that there is an overarching filter class. This class has access to all the different filter modes and can pass informations such as the active sensors, the measurement, the time and the servicer satellite state to the different filter modes. In return, it can extract the filter outputs from each filter separately.

Thus, when a filter mode is specified in the GNC configuration file this filter mode information is passed down to the filter class which in return passes the GNC system information to the relevant filter mode and extracts the required output information from the filter mode.
The system was set up in such a way since it allows to keep the addition of new filter modes fully decoupled from the wider GNC system. This means that when a new filter mode is added the only thing that needs to be adjusted is the filter class so that it can pass and extract information. All other unit tests and GNC functionalities remain unchanged.

The disadvantage of this approach is that currently filter modes cannot be changed mid-test-run. This is due to the fact that all filters would need to run in parallel and perform their own state estimation or be initialised at every iteration step with the full available estimation information from the currently running filter mode to allow for a smooth transition. This is not practical for it would highly increase the computational effort and since it was decided that it is highly unlikely in this early stage of testing that filter modes would be changed in the middle of a test run. However, this is a functionality that may become more relevant in the future as more specialised filter modes are implemented.

5.2.2. Data flow to the filter in EPOS 2.0

The primary difference between the simulation performance test and the hardware-in-the-loop test is that the filter is supplied with real sensory inputs in the latter. Figure 5.4 shows the data flow in the GNC system in more detail than before. The filter is supplied with the pose of the target satellite (position and attitude in quaternions) from either a PMD camera (3D imagery) or one of three 2D cameras. The imagery from these cameras is first assessed in an image processing step. The relevant information on position and attitude of the target is extracted and fed to the filter using a measurement vector in the chaser (servicer satellite) frame. The measurement vector is a 7D vector of which the first 3 entries are the position parameters in x, y and z, and the latter 4 parameters are the quaternions.

The filter performs the state estimation and, as was explained before, passes the state to the guidance system and the controller. In addition to the cameras, the EPOS 2.0 facility has the ability to feed the filter with simulated measurement data from a sensor dummy. This sensor dummy generates the same type of information as is extracted from the images taken by real sensors and passes it directly to the filter.

An example of the images collected by the sensors is shown in Figure 5.5. The target is seen in white from multiple distances. From one image, two parameters can be extracted currently by image processing algorithms: the position of the target and the attitude of the target. The state estimation from the filter is overlayed with the target in purple. The resolution of the camera as well as the shape and reflections on the target highlight could affect the accuracy of the extracted information at different distances. The information is extracted using a model-based visual tracking algorithm, meaning that the system uses the knowledge of the geometrical parameters of the target satellite model to identify edges. This was done since it was identified that this approach works best under the challenging lighting conditions both in space and in the EPOS 2.0 facility.[5]

5.2.3. Scenarios tested for filter performance assessment

Since the EPOS facility is controlled by people and features real sensor noise, and since not all filters are run in parallel, each test run in the facility will be different. Thus, each filter output will differ and include errors and variations. To account for this, a similar test approach was followed each time. Nevertheless, there are still substantial differences in the way tests were performed which may have affected test results and thus perceived performance.

Each test is started by moving the robots (satellites) to a relative distance of 15 metres. At this first hold point, the filters are initialised and it is waited for them to stabilise, based on information from the sensor dummy. The filter input is then switched to use real visual sensor information for state estimation. Thereafter, the servicer satellite is moved to a hold point at 6 metres from the target with a velocity of 0.02 metres per second in closed loop. After holding and observing the filter performance at 6 metres the servicer is moved to a distance of 4 metres at a velocity of 0.01 metres per second.

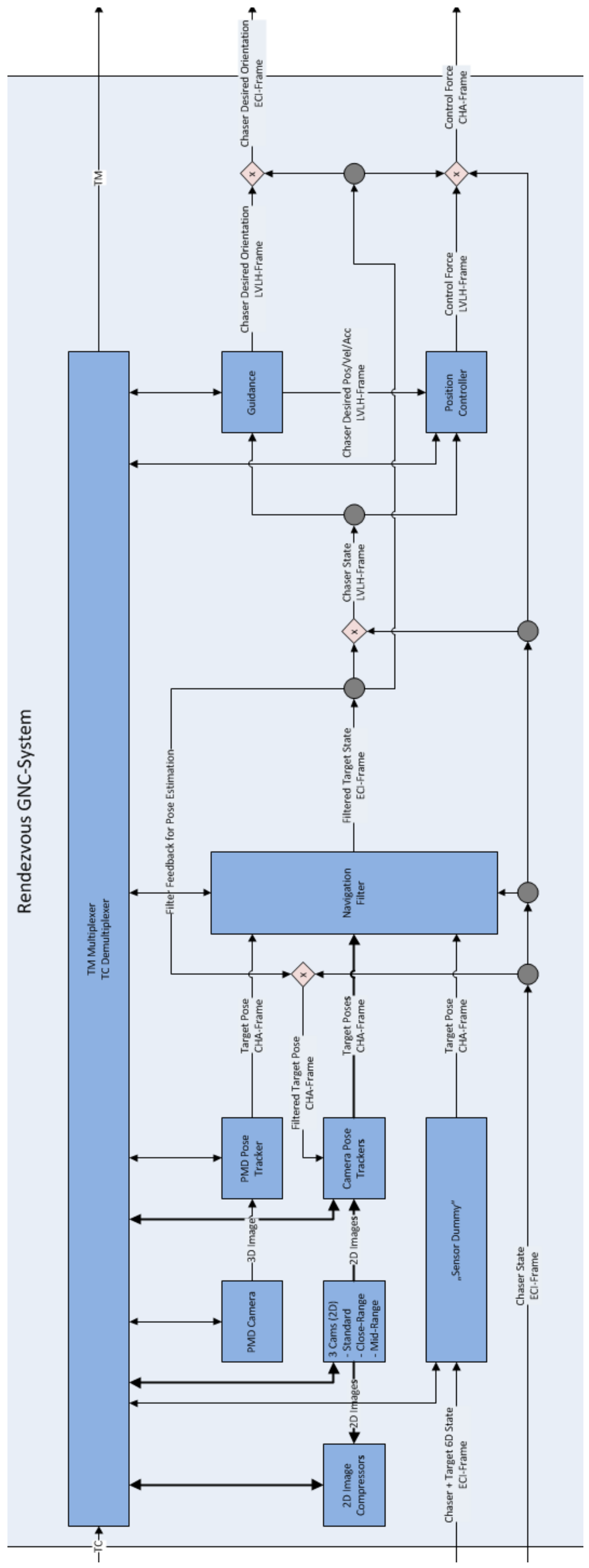


Figure 5.4: Representation of the information flow within the gnc system as it works in the EPOS 2.0 facility, provided by DLR

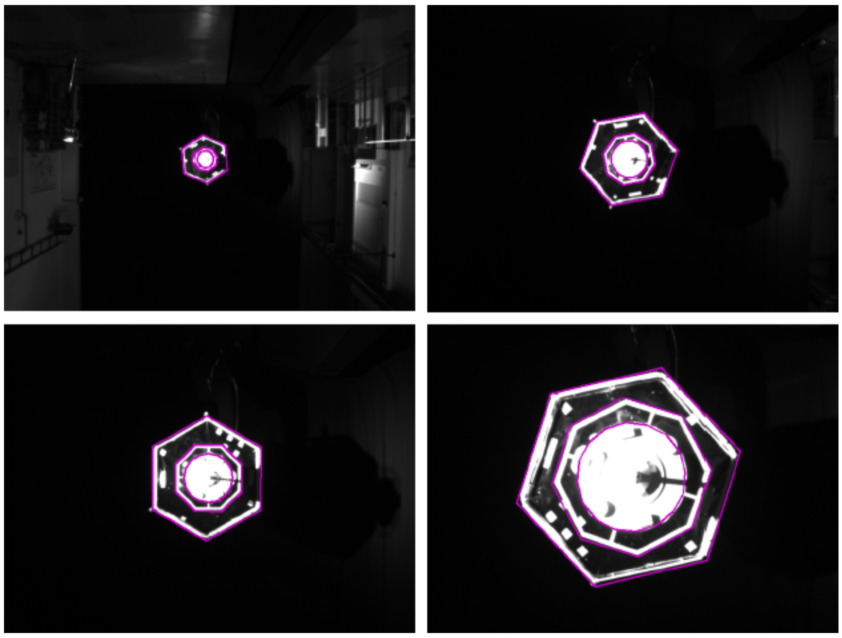


Figure 5.5: Black and white image of target satellite, overlayed by filter state estimate (purple), provided by DLR, DLR (CC BY3.0)

5.2.4. Data handling

In order to judge the performance of filter modes, their performance throughout the test is observed and anomalies are noted. Additionally, performance data is collected to more accurately and numerically judge them.

Data collected during testing

The criteria by which the performance of a filter is judged were explained before in 1.9.10. In order to assess the performance of a filter the following information needs to be collected:

- Estimated state: The estimated state shows the filters estimate of the target state. It is the primary filter output that is passed to the GNC system.
- True target state: The true target state needs to be logged to judge how correct the estimation is. The difference between estimated target state and true target state yields the estimation error.
- Measurement: The measurement collected by the sensors on the servicer is collected to judge whether the information that is passed to the filter improves or worsens the estimate.

Other parameters such as the covariance could not be logged. This functionality would need to be added in the future. However, the state estimation accuracy is the most important aspect for assessment, which is why the other filter performance information is considered secondary and can be omitted for fulfilling the scope of this project.

Data storage and management

All data is logged in sql files and saved in the EPOS 2.0 control system. A separate collection of data files is generated for each test run. They are transferred via memory stick for further assessment and are later converted into csv files for read-out. The relevant files, their structure and content is explained below.

Navigation Output Logfile

The file saved as navOutLog contains the filter state estimate information and the true servicer satellite information for every filter iteration step. They are ordered by filter time, the logged time of iteration execution. Since the filter runs at 10 Hz there is a new state estimate available at every 0.1 seconds throughout the run. All information is given in an ECI reference frame.

Sensor Dummy Input Logfile

The file saved as sdInLog contains the input information to the sensor dummy. This is the true position of the target satellite and the servicer satellite (chaser) in the orbital model. The information is given in an ECI reference frame. The states are not logged regularly as was the case with the filters but they are logged following their own timing. Thus, the filter log-time and the true log-time may not match and require interpolation. Furthermore, the true spin rate of the target is not logged (it is set to 0 as a default). This was only realised after testing was finished and could not be compensated for anymore, which is why the following analysis only assesses the quality of the estimation of the other state parameters.

Camera Logfile

The file campeOutlog saves camera data collects whenever the camera is running. Sensor output information is collected whenever it is available and thus does not match the time stamps of the filter iterations. From the visual sensors, only information on position and attitude is available. All information is given in a body reference frame centred at the servicer satellite. Position information is given in the x, y and z coordinates of the body frame. The attitude is given in quaternions.

The camera logfile needs to be stored since it can be used to identify potential camera - target misalignments. Such offsets can be a source of error in the state estimation, if, for example, a state prediction actually gets worsened in its update step due to a faulty measurement.

5.2.5. Data assessment

Data assessment is performed in a similar way as in the simulation performance test by reading the data from csv files into data frames in a dedicated python script. The following section describes the data manipulation.

Correlation of true state to filter time

As was mentioned before, the true position of the target time is not logged at the same intervals as the filter estimates. To account for this, the two data frames containing the estimation information (navOutLog) and the true state (sdInLog) are read into an extrapolation function. The data frame containing the estimation information is appended columns to include all true state parameters. The function then loops through the entirety of the estimation data frame (df_{est}). At every iteration step, the filter time information is extracted (t_{flt}). Parallelly, the algorithm propagates through the data frame with the true states (df_{true}), starting at the first time step, and selects two consecutive time steps ($t_{target-1}$ and $t_{target-2}$). The algorithm then checks whether the filter time is larger than the higher target time, and, if so, selects the next consecutive target times, as can be seen in Equation 5.1.

$$\begin{aligned} \text{while } t_{target-2} &<= t_{flt}: \\ i_{target} &+= 1 \\ t_{target-1} &= df_{true}[i_{target}, t] \\ t_{target-2} &= df_{true}[i_{target}+1, t] \end{aligned} \tag{5.1}$$

Once $t_{target-2}$ is larger than the filter time it is checked whether the filter time lies between the two target times (see Equation 5.2). This is necessary to assess since target time iteration steps may be small enough to not contain a filter update at all.

$$\text{if } t_{target-1} < t_{flt} \text{ and } t_{target-2} > t_{flt} \tag{5.2}$$

Once it is confirmed that a filter update step lies between the two true target state update steps, the true position is extrapolated to find a corresponding true position at each filter update step. This is done by first extracting the true state parameter at $t_{target-1}$ (p_{low}) and at $t_{target-2}$ (p_{high}). Then the corresponding extrapolated true state parameter p_{true} at t_{flt} is found using Equation 5.3 and is added to the estimation data frame.

$$p_{true} = p_{low} + \frac{t_{flt} - t_{target-1}}{t_{target-2} - t_{target-1}} * (p_{high} - p_{low}) \tag{5.3}$$

Thus, a corresponding true state at every filter iteration step can be found.

Estimation error assessment

In the same way as in the simulation performance test the estimation error can be assessed for every state parameter now that the true state and the filter estimate are available at every time step. The error is simply computed from the difference of the estimate and the true state. For the position and the velocity estimates the absolute error is computed as well.
The true state parameters for the spin rate of the target are not logged and are thus excluded from the assessment.
The quaternion error is translated into Euler angles for easier visualisation later on.

Correlation of measurement data to filter time

As was mentioned previously, the interval of the logged measurement data does not match the filter estimation interval either. Therefore, the approach as for the correlation of the true state to the filter time is followed to determine a measurement entry at every filter iteration step at which the cameras were running.

As was discussed before, the measurements are only provided in a servicer centred body reference frame. Therefore, the measurement information needs to be translated into ECI to be comparable with the true state information. Firstly, the reference frame is established with

$$q_{cha-ECI} = [q_{x-cha-ECI}, q_{y-cha-ECI}, q_{z-cha-ECI}, q_{s-cha-ECI}]$$

From this, the $3x3$ reference matrix $R_{ref2body}$ is found using Equation 5.4 where $q = q_{cha-ECI}$.

$$\begin{aligned}
R_{ref2body}[0,0] &= q[0]*q[0]+q[3]*q[3]-q[1]*q[1]-q[2]*q[2]\\
R_{ref2body}[0,1] &= 2*(q[0]*q[1]+q[2]*q[3])\\
R_{ref2body}[0,2] &= 2*(q[0]*q[2]-q[1]*q[3])\\
R_{ref2body}[1,0] &= 2*(q[0]*q[1]-q[2]*q[3])\\
R_{ref2body}[1,1] &= -q[0]*q[0]+q[3]*q[3]+q[1]*q[1]-q[2]*q[2]\\
R_{ref2body}[1,2] &= 2*(q[1]*q[2]+q[0]*q[3])\\
R_{ref2body}[2,0] &= 2*(q[0]*q[2]+q[1]*q[3])\\
R_{ref2body}[2,1] &= 2*(q[1]*q[2]-q[0]*q[3])\\
R_{ref2body}[2,2] &= -q[0]*q[0]+q[3]*q[3]-q[1]*q[1]+q[2]*q[2]
\end{aligned} \tag{5.4}$$

The transpose of this matrix is the reference matrix of the body centred reference frame to the ECI reference frame, $R_{body2ECI} = R^T_{ref2body}$. The transformation matrix $T_{body2ref}$ from the body reference frame to the ECI frame is then found from the position of the chaser satellite (which takes the images) in ECI, $p_{cha-ECI}$, and $R_{body2ECI}$, as shown in Equation 5.5, with $R = R_{body2ECI}$ and $p = p_{cha-ECI}$.

$$T_{body2ref} = \begin{bmatrix} R[0,0] & R[0,1] & R[0,2] & p[0] \\ R[1,0] & R[1,1] & R[1,2] & p[1] \\ R[2,0] & R[2,1] & R[2,2] & p[2] \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5.5}$$

The measured position in the body reference frame $p_{meas-body}$ is then transformed to the ECI frame using the transformation matrix to find the 3-dimensional vector $p_{meas-ECI}$, as in Equation 5.6. The 4th value ($misc$) in the output is ignored.

$$\begin{bmatrix} p_{meas-ECI} \\ misc \end{bmatrix} = T_{body2ref} \cdot \begin{bmatrix} p_{meas-body}[0] \\ p_{meas-body}[1] \\ p_{meas-body}[2] \\ 1.0 \end{bmatrix} \tag{5.6}$$

The quaternions in the ECI frame are found using the transformation matrix $T_{body2ref}$ and the measured quaternions $q_{meas-body}$. First, a reference matrix R_{ref} is formed using the same Equation as in 5.4 but with $q_{meas-body}$ as q. Next, the reference matrix is regained by extracting it from the top left $3x3$ entries of $T_{body2ref}$, yielding again $R_{body2ECI}$. Next, the reference matrix between the quaternions and the ECI frame is formed using Equation 5.7.

$$R_{quat2ECI} = R_{body2ECI} \cdot R_{ref} \tag{5.7}$$

The transform of matrix $R_{quat2ECI}$ is then finally used as input R for the algorithm presented in Equation 5.8.

$$
\begin{aligned}
\text{a} &= R[0,0] + R[1,1] + R[2,2] \\
\text{if } a > 0.0: & \\
b &= \sqrt{1+a} \\
c &= 2.\cdot b \\
q_x &= (R[1,2] - R[2,1])/c \\
q_y &= (R[2,0] - R[0,2])/c \\
q_z &= (R[0,1] - R[1,0])/c \\
q_s &= 0.5 * b \\
\text{else:} & \\
\text{if } R[0,0] > R[1,1] \text{ and } R[0,0] > R[2,2]: & \\
b &= \sqrt{1 + (R[0,0] - (R[1,1] + R[2,2]))} \\
c &= 2 \cdot b \\
q_x &= 0.5 \cdot b \\
q_y &= (R[1,0] + R[0,1])/c \\
q_z &= (R[2,0] + R[0,2])/c \\
q_s &= (R[1,2] - R[2,1])/c \\
\text{elif } R[1,1] > R[2,2]: & \\
b &= \sqrt{1 + (R[1,1] - (R[2,2] + R[0,0]))} \\
c &= 2 \cdot b \\
q_x &= (R[1,0] + R[0,1])/c \\
q_y &= 0.5 \cdot b \\
q_z &= (R[1,2] + R[2,1])/c \\
q_s &= (R[2,0] - R[0,2])/c \\
\text{else:} & \\
b &= \sqrt{1 + (R[2,2] - (R[0,0] + R[1,1]))} \\
c &= 2 \cdot b \\
q_x &= (R[2,0] + R[0,2])/c \\
q_y &= (R[1,2] + R[2,1])/c \\
q_z &= 0.5 \cdot b \\
q_s &= (R[0,1] - R[1,0])/c
\end{aligned}
\tag{5.8}
$$

Finally, the vector q determined in Equation 5.8 is normalised by dividing it by the sum of its entries. Thereafter, the sign of the first entry is checked. If $q[0]$ is negative, the vector is multiplied by -1. The resulting vector q is the measured quaternion vector in ECI. The accuracy of the measurement data is computed as for all other state parameters by determining the difference between the measured state and the true state.

Hold-points and approaches

Since the EPOS 2.0 tests feature actual approaches it is expected that there are differences in state estimation accuracy between the approaches and the times when the satellites are held at constant distances (hold-points). These are thus considered separately. In order to identify hold-points the times during a test run were noted at which satellites moved and kept position relative to each other.

5.2.6. Test planning

Tests were set up as described in 5.2.1 and 5.2.3. The test plans thus all featured the same test procedure. The following parameters were recorded in addition to the logged data in each test:

- Time of start of test, to identify the saved logfiles based on their creation time
- Time of end of test, to determine length of test

- Time and type of input commands given to facility to correlate potential filter deviations to commands
- Irregularities and exceptional observations throughout test

5.3. Performing hardware-in-the-loop tests

This section describes how the tests were performed, what was observed and what deviations occurred from the test plans, what lessons were learned and the adjustments that were made.

5.3.1. Observations during testing

Several observations were made throughout testing, especially on the time of test runs and on the accuracy of filter estimates throughout close proximity operation at a distance below 5 metres using the two newly implemented filter modes.

The total time of a test run was not deemed too crucial for the accuracy of an estimate at the beginning of the test sequence, which is why the total time of the original EKF test was longer than for the other filters. All filters were given time to settle and converge towards the target ad additional time was given to assess the quality of the filter performance. However, it was realised later that, in order to judge the filters across the entirety of the test runs, the intervals of movements and hold-points should be roughly the same since otherwise the average accuracy may be distorted. This is a potential error source in the assessment that needs to be kept in mind.

It was observed that the EKFS and the UKF showed erratic filter behaviour in close proximity operation. The EKFS was accidentally started when the satellites were at a relative distance of 5 metres in an early test run. The filter estimate almost immediately diverged and the test had to be stopped. When the filter was restarted with the satellites at a 15 metre distance it worked well (this is the test case assessed in the following sections) and the normal test approach was performed. Even below a relative distance of 5 metres the filter performed fine. It could not be determined what caused the initial filter behaviour, but the EKFS is not considered reliable for close proximity operation in its current state.
The UKF faced a similar problem throughout testing. It was initiated at 15 metres as planned and performed the approach nominally. However, during the approach from 6 to 4 metres the filter estimate diverged between 5 and 4 metres distance. The test had to be stopped to avoid harm to the robots. The behaviour could not be reproduced reliably.

Thus, both new filters are not entirely stable in their performance at close proximity operation. Further stability assessments are still ongoing at DLR at the time of writing of this report. A potential error source that was identified by assessing differences between far distance and close distance operation is the increased image processing time at close proximity operation. This may cause the filters to use outdated measurements, which may offset the EKFS and the UKF. However, it is unclear why this would only be a problem for these new filters.

5.3.2. Lessons learned and test adjustments

Purely based on the observations during conducting the tests, several conclusions and recommendations can be given.

- The test times need to be standardised in a better way to more easily control filter performance
- Ideally, the filters should be able to run parallelly given the same inputs; however, facility limitations and filters yielding non-representative results when fed with outputs from a different filter mode may hinder this in the future
- More stability assessments at close proximity operation are required to further develop the EKFS and the UKF

5.4. EPOS 2.0 test results

All filter data was assessed the same way, in a raw format, where all data entries were considered for analysis, and a cut-off version, excluding outliers. Data entries were considered outliers when they exceeded a certain

Table 5.1: Thresholds for data cut off, used for outlier identification in results

Absolute position	Absolute velocity	Attitude angles
0.5 m	0.5 m/s	0.4 rad

Table 5.2: Error in state estimate using EKF, hardware-in-the-loop test

Parameter	error
Position mean error	0.085 m
Velocity mean error	0.005 m
Attitude error x	0.009984 rad
Attitude error y	0.007797 rad
Attitude error z	0.007197 rad
Measurement mean position error	0.125191682

threshold, which was determined from assessing the plotted data. Table 5.1 shows the value above which a data entry was considered an outlier by parameter. Excluding outliers rigorously is important since at the beginning and end of each test run the filter is not initially set to the actual target state but assumes a default zero state which is then set to orbital parameters once the filter is synced to the target. Thus, great offsets are possible.

For the figures in the following sections, the error in estimation is presented in the same way as for the simulation performance test (see Section 4.5). The colour coding is the same.

5.4.1. EKF performance

The performances of all filters are presented in the format excluding outliers and excluding error values that are exactly 0, since the algorithm logs a zero error when no data is available (for example at time steps where no measurement is recorded). The only state parameter not assessed here is the spin rate. As in the simulation performance test, the EKF performance across the test run is presented below. In addition, the measurement accuracy is determined. The mean errors across the test run are shown in Table 5.2. As can be seen, all errors are very low for the EKF. No error stands out particularly. The low velocity estimation error translates into a low position error, which, however, seems to have a slightly higher error. This is not surprising since the filter corrects its estimation towards the true state over time, reducing the (initially large) position estimation error using a smaller velocity estimation error. Figures 5.6 and 5.7 show the error of the EKF estimate in position and velocity across the duration of the test run. The deviations in the beginning and in the end are attributed to the filter initialisation and shut-down, when default values are assumed. These areas are excluded from analysis and assessment, but are later discussed for the other filter modes. As can be seen, the position estimate shows larger error values across the entire test and never achieves the accuracy of the velocity estimation, as was assumed from the numerical assessment.

As was mentioned before, the same assessment is performed for hold-points and approaches separately. The data collection is shown in Tables 5.3 and 5.4, respectively.

The tables above show that the EKF performs slightly worse in the state estimation when the relative distance between the satellites is kept constant, as showcased by the larger error values in Table 5.3 than in Table 5.4. This is also supported by the graphical assessment of the errors.

The attitude performance across the test duration is shown in Figure 5.8. It can be seen that the attitude

Table 5.3: Performance of EKF in state estimation at hold points

Parameter	error
Position mean error	0.115 m
Velocity mean error	0.007 m
Attitude error x	0.010894 rad
Attitude error y	0.008897 rad
Attitude error z	0.007286 rad

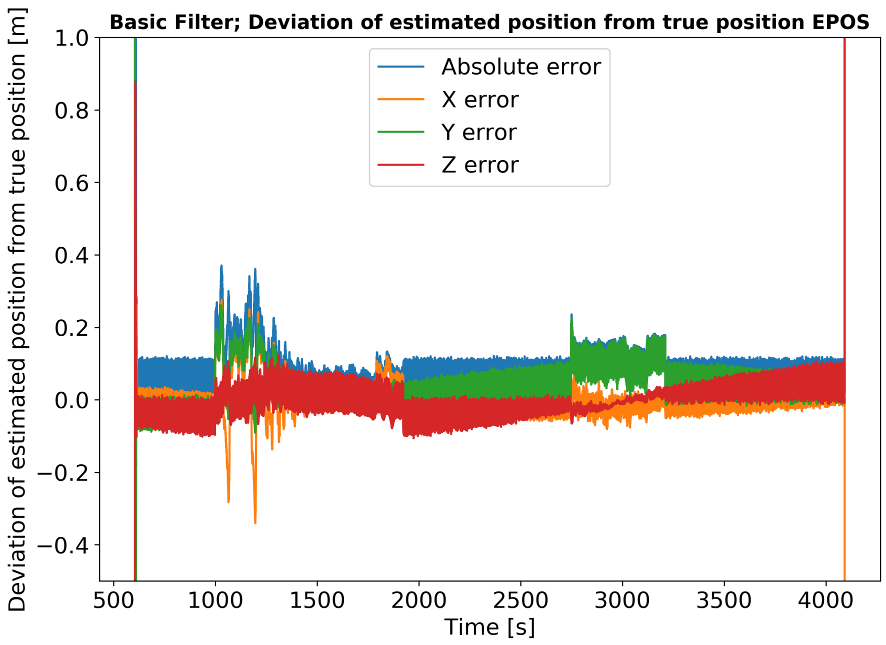


Figure 5.6: Position estimation error of EKF state estimate in EPOS 2.0 facility test

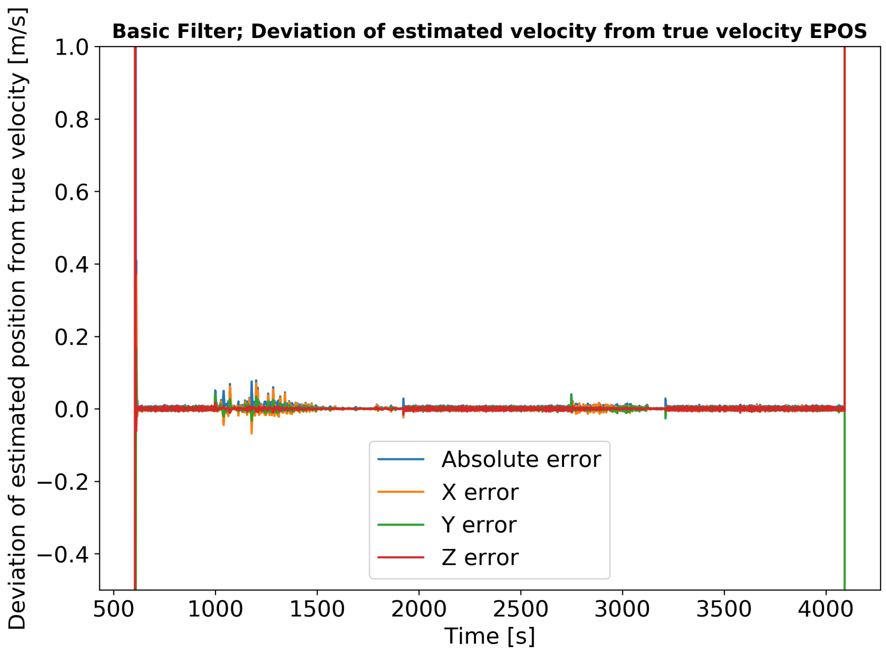


Figure 5.7: Velocity estimation error of EKF state estimate in EPOS 2.0 facility test

estimation is working better in cases where the position and velocity estimation is performing worse. Since there should not be a correlation between these factors (the respective blocks in the covariance matrix are 0), this means that the attitude estimation works better in hold points than during approaches. This could be due to the fact that changes of the imagery of the target satellite during the approach caused by the decrease of the relative distance could be misinterpreted by the image processing as attitude changes, decreasing the quality of the perceived measurement that is passed to the filter.

Table 5.4: Performance of EKF in state estimation during approach

Parameter	error
Position mean error	0.076 m
Velocity mean error	0.007 m
Attitude error x	0.006280 rad
Attitude error y	0.003835 rad
Attitude error z	0.002493 rad

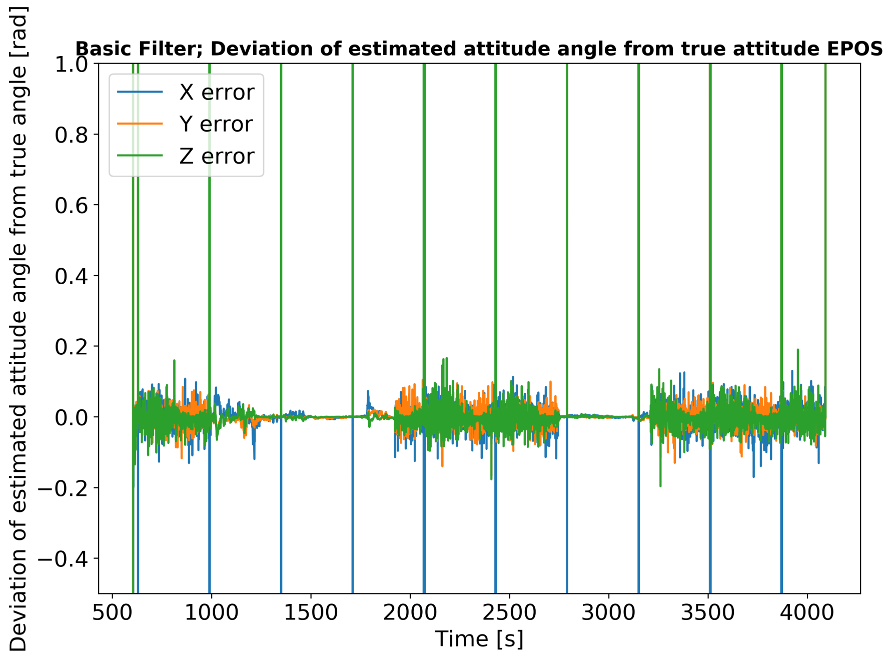


Figure 5.8: Attitude estimation error of EKF state estimate in EPOS 2.0 facility test

5.4.2. EKFS performance

The EKFS performance relative to the original EKF is assessed using the total test run data excluding outliers. The data for the total test runs as well as a comparison of the respective hold points and approaches is presented here. Table 5.5 shows the overall performance comparison.

Overall, the performance of the EKFS compared to the EKF worsens. However, it can also be seen that the average measurement error of the measurement used by the EKFS excluding outliers was worse throughout the test run.

Figures 5.9 and 5.10 show the position estimation error and the velocity estimation error from the EKFS test, respectively. As can be seen, the EKFS position estimate develops an offset from the true state at around 5300 seconds into the test run (the high test time originated from an initially failed run after which the system was set up again but the internal clock kept running). This time roughly coincides with the start of the approach from 15 metres. As shown below, this matches the EKFS performance in approaches, which is worse

Table 5.5: Performance of EKFS compared to EKF in state estimation, hardware-in-the-loop test

Parameter	error
Position mean error	-45.2 %
Velocity mean error	-8.9 %
Attitude error x	-9 %
Attitude error y	-7 %
Attitude error z	-9.5 %
Measurement mean position error	-52.4 %

Table 5.6: Comparison of EKFS to EKF performance at hold point state estimation

Parameter	error
Position mean error	-20 %
Velocity mean error	-8.7 %
Attitude error x	0.3 %
Attitude error y	-7.9 %
Attitude error z	-0.5 %

Table 5.7: Comparison of EKFS to EKF performance during approach

Parameter	error
Position mean error	-76.6 %
Velocity mean error	-31.8 %
Attitude error x	-40.2 %
Attitude error y	-21 %
Attitude error z	-53.8 %

than at relative satellite hold points. Furthermore, from assessing the measurement quality around the onset of the approach, it can be seen that the measurement also features an offset, thus causing the EKFS to perform worse with respect to the real target estimation. Figure 5.11 shows this. Large measurement error spikes in the beginning of the test run can be compensated for by the filter, but the offset of the measurement disturbs the overall performance in the approach. Further investigation shows that the filter estimation reflects the measurement offset well, meaning that no conclusive performance comparison is possible for the different filters when not using the same measurement input quality.

This may also be a reason why the EKFS performs worse than the EKF overall, as the test was dominated by approaches and the filter follows the measurements.

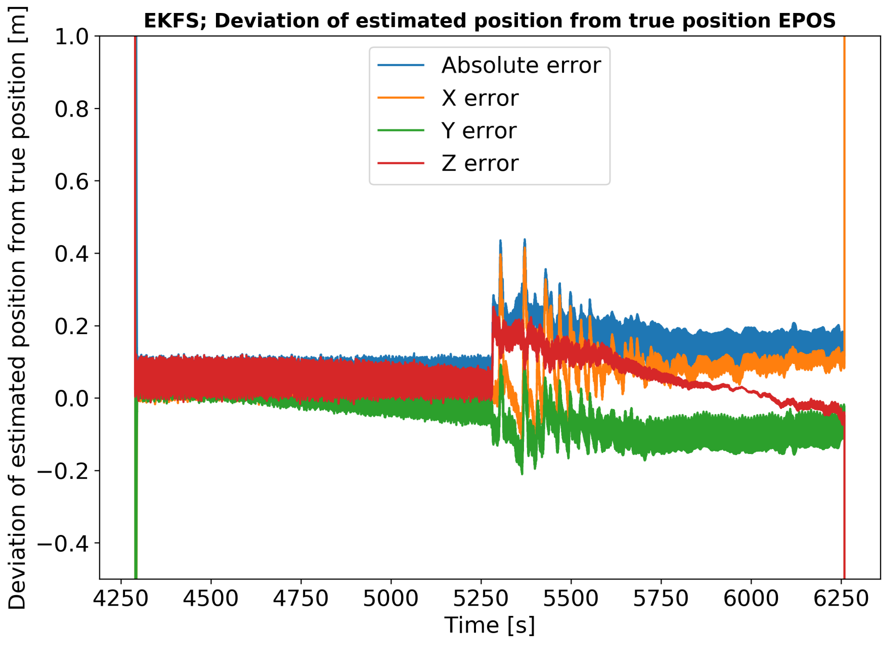


Figure 5.9: Position estimation error of EKFS state estimate in EPOS 2.0 facility test; an estimate offset can be observed

Table 5.6 shows the improvement and worsening of the EKFS performance at hold-points compared to the hold-point performance of the EKF.

The comparative approach performance is shown in Table 5.7. As can be seen, the EKFS performs better in the cases where the satellites are held at constant distances from each other. This is not surprising since

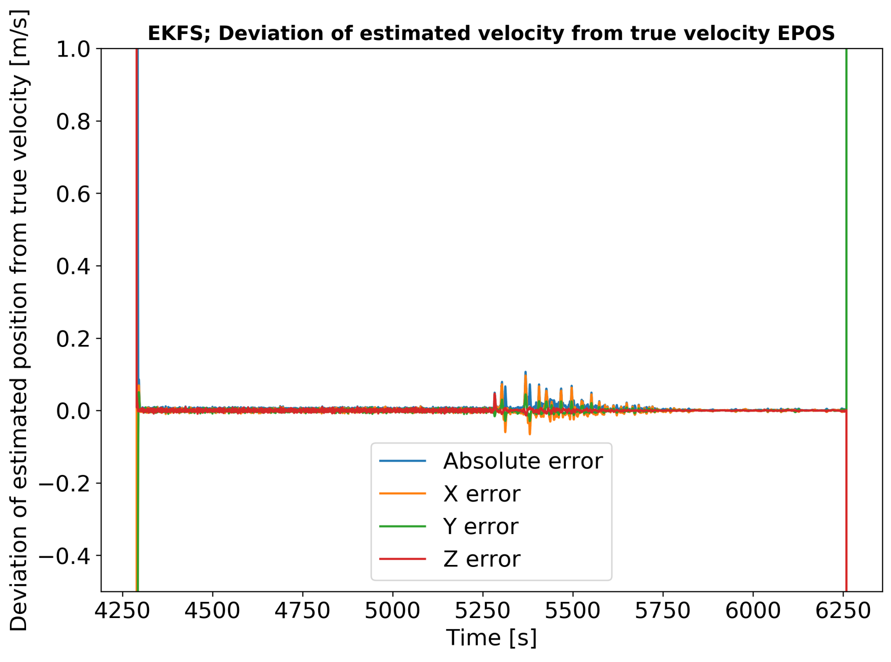


Figure 5.10: Velocity estimation error of EKFS state estimate in EPOS 2.0 facility test

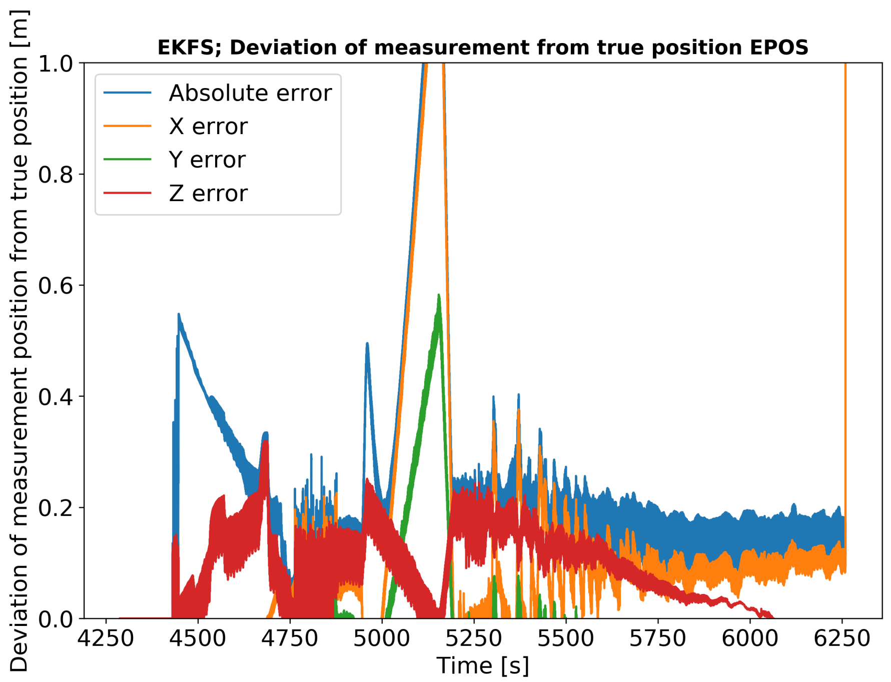


Figure 5.11: Accuracy of the position measurement of the measurement used in the EKFS test run

the EKF performance is better in the cases where the position is changed, making it more difficult for the EKFS to outperform the original filter in this case. It is, however, interesting to note that the position estimate is always inferior to the EKF estimate, partially also due to the offset discussed above.

Interestingly, the attitude estimation seems to improve with the onset of the approach. The performance throughout the test is shown in Figure 5.12, where it can be seen that the attitude estimation quality again improves when the position estimate gets worse. However, in this case it seems like this coincides with the onset of the approach.

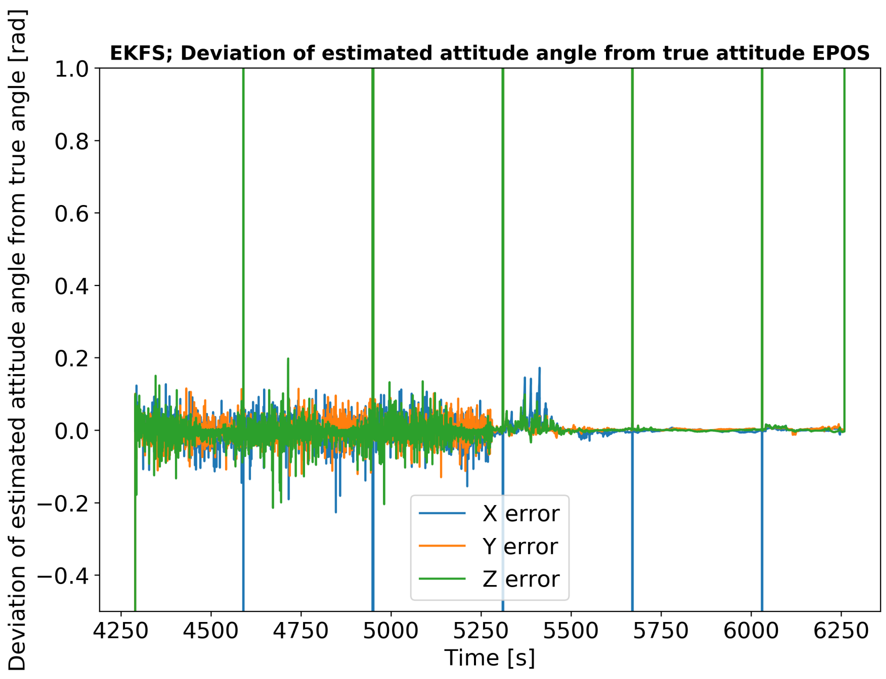


Figure 5.12: Attitude estimation error of EKFS state estimate in EPOS 2.0 facility test

Table 5.8: Performance of UKF compared to EKF in state estimation, hardware-in-the-loop test

Parameter	error
Position mean error	10.2 %
Velocity mean error	-98.7 %
Attitude error x	47.8 %
Attitude error y	21.8 %
Attitude error z	34.2 %
Measurement mean position error	50.4 %

Conclusively, the EKFS performs worse than the EKF. However, it also uses a far worse measurement input than the original filter. Additional tests have to show whether the EKFS performance can be improved and why erratic divergence occurred at close-proximity application. As for the simulation performance test, the hardware-in-the-loop test verifies and validates the performance of the EKFS for the test case at hand. However, due to the poor quality of the measurement and state estimation fluctuations when initialising the filter at close-proximity it is recommended to perform further tests to make sure that the validation is conclusive.

5.4.3. UKF performance

The UKF is assessed comparatively the same way as the EKFS. The overall compared performance data is shown in Table 5.8. The UKF outperforms the EKF in all parameters except for the velocity estimate. This indicates that the position estimate may be more jumpy, but not wrong on average. However, this could cause concern during close proximity approaches where a very smooth and accurate estimate is required and may even lead to divergence, as was observed in the real life testing. Figures 5.13 and 5.14 show the UKF performance in position estimate and velocity estimate. Again, the position estimate is offset by the start of the approach but it quickly resettles towards the actual target state. The velocity error is smaller but follows the same pattern. As shown in Figure 5.15, the measurement fed to the UKF is far more accurate and less disturbed by outliers than the measurement used for the EKFS. This helps the UKF to achieve a better performance in the test. However, as can be seen, both parameters diverge at the end of the test run. This has nothing to do with the system shut-down but an actual filter divergence was observed during testing at the approach between 5 and 4 metres distance of the servicer satellite to the target. A detailed assessment of the position estimate in this section of the test run, shown in Figure 5.16, shows that the position diverges quickly.

The velocity estimate shows the same behaviour. It has not been conclusively determined what caused this behaviour.

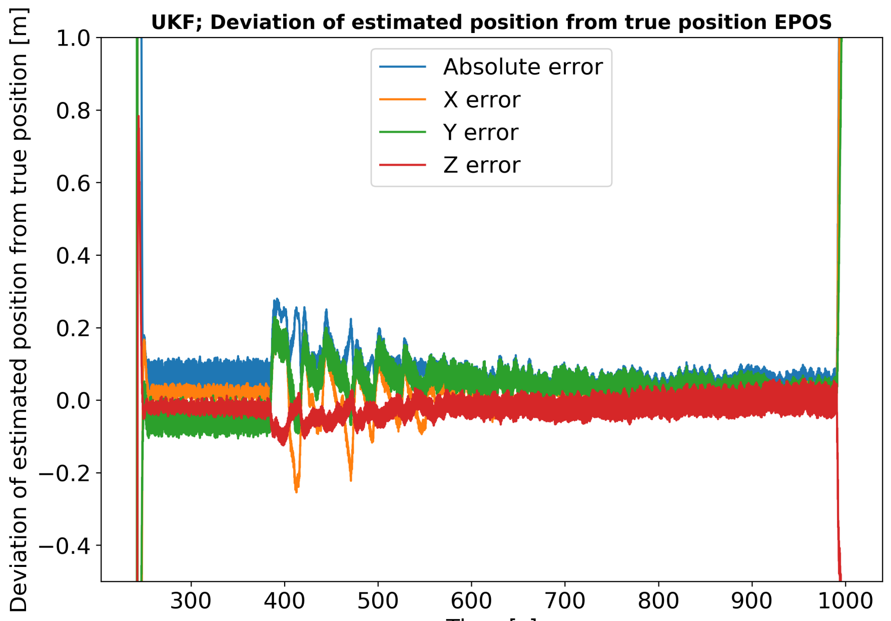


Figure 5.13: Position estimation error of UKF state estimate in EPOS 2.0 facility test

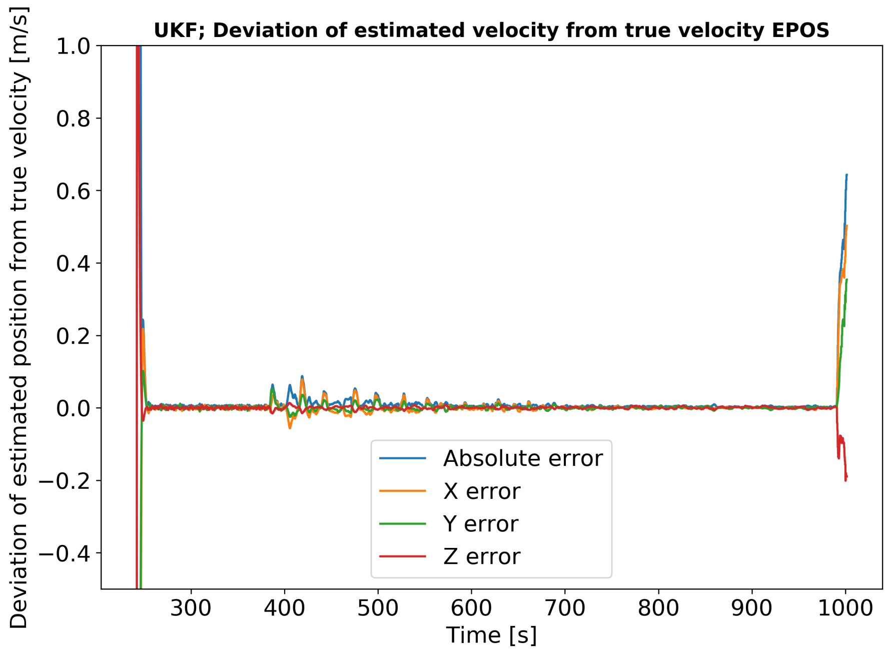


Figure 5.14: Velocity estimation error of UKF state estimate in EPOS 2.0 facility test

The attitude estimate is far more accurate than in the case of the simulation performance test. This could either indicate that the simulation performance test does not accurately represent reality (for example through its propagation model, as was discussed before) or that the UKF performs far better in test cases where measurements are delayed and where the filter and measurement update rates do not match. This latter point was partially observed in the simulation performance test as well. However, as was shown for test case 11 in Figure 4.21, the attitude estimation of the UKF also showed faulty behaviour in such test cases.

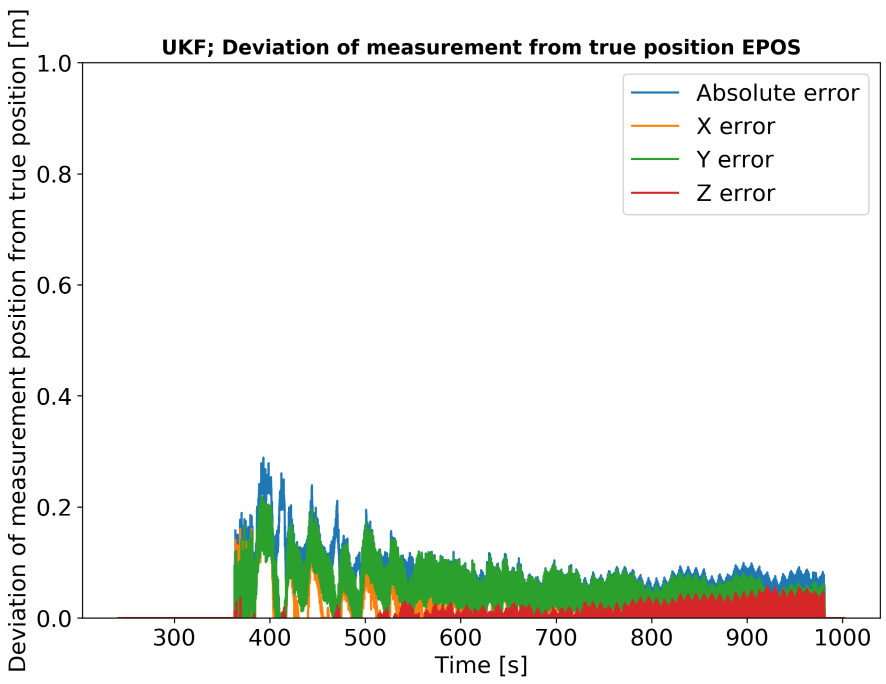


Figure 5.15: Accuracy of the position measurement of the measurement used in the UKF test run

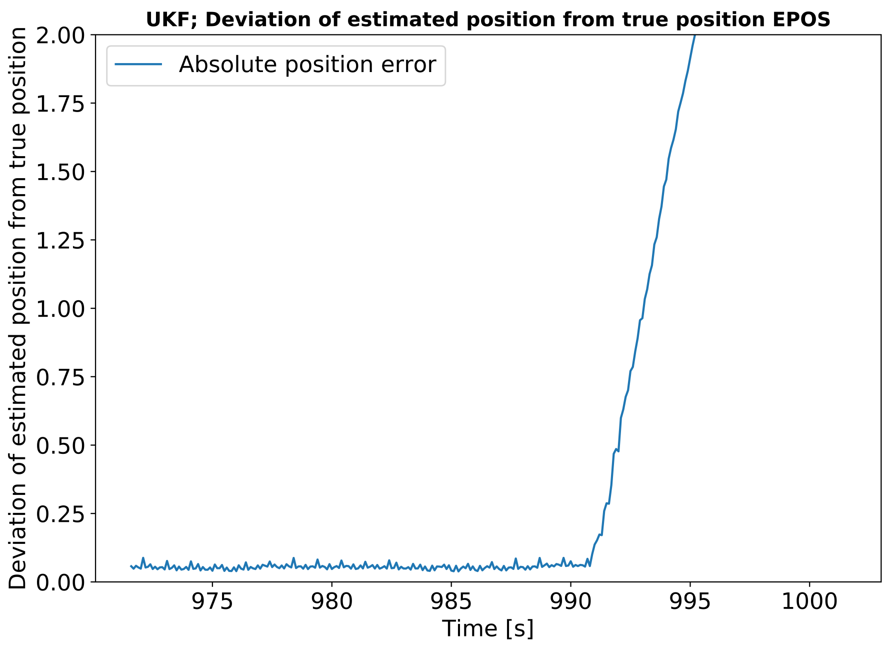


Figure 5.16: Accuracy of the position measurement in the end of the test run; divergence is observed

Furthermore, this observation does not match with the insight that the implementation of taking weighted averages of quaternions is faulty. It is not yet explained why attitude estimation problems only occurred very late in the UKF test run. However, as was seen in some test cases in the simulation performance test where problems also only occurred very late or hardly at all, this was also the case in the simulation test. Thus, it is seen again that the UKF implementation of quaternion calculation would need to be adjusted and then tested again. This is seen as a confirmation of the observations from the simulation test.

Throughout the simulation performance test it was observed multiple times that the UKF performs well in position estimation and on average better in general for test cases 9 through 14 and comparative performance decreases are primarily due to the test cases where filter and measurement update intervals match. The EPOS 2.0 test partly verifies this general behaviour.

The attitude performance of the UKF is shown in Figure 5.17. The performance resembles the other filters. However, as for the position estimate and the velocity estimate, the final iterations where the estimate diverged are dominated by an error increase, as was eluded to above. A zoomed in version of the quaternion state estimation of this section is shown in Figure 5.18, where it can be observed that the quaternions deviate away from the true target state in a manor similar to the one observed in multiple test cases in the simulation test (see for example Figure 4.21). Quaternions are shown here to highlight the resemblance with the simulation performance test. Again, the potentially faulty implementation of the UKF approach, taking weighted averages of the state parameters to determine a state estimate, may cause the UKF quaternion estimation to perform in this way and result in divergence. Furthermore, comparing Figures 5.16 and 5.18 the onset of the quaternion divergence seems to occur slightly earlier than the onset of the position deviation. Thus, it is postulated that the error source lies in the quaternion behaviour, which would match with the assessment of the implementation problem of the quaternions in this kind of UKF. However, the quaternion and position estimation should not be closely related (the covariance is near zero), which is why this requires further investigation. The general take away is, nevertheless, that these observations are in line with the conclusion from the simulation performance test, in that the EPOS 2.0 test can neither be used to fully verify nor validate the performance of the attitude estimation for the UKF. Again, it is concluded that the unit tests are not enough to conclusively verify the filter performance since after many iterations errors occur, and the hardware-in-the-loop test can as a result not be used to validate the filter performance of the UKF.

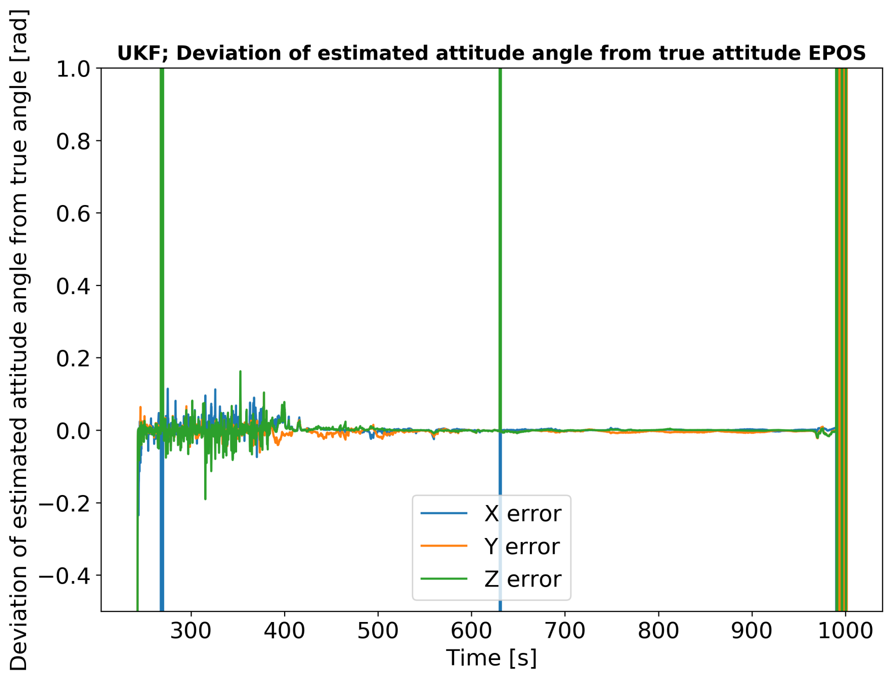


Figure 5.17: Attitude estimation error of UKF state estimate in EPOS 2.0 facility test

The UKF hold-point and approach performances are shown in Tables 5.9 and 5.10, respectively.

As can be seen, the improved performance in the UKF tests primarily came from the approach assessment. It has to be stated that the approach period was longer then the hold-point period in the test that was performed. Still, it is noteworthy that the velocity estimation is off in both the hold-point case and the approach case. The UKF is thus also not yet an alternative for the original EKF that should be applied without further improvement. Particularly the strong estimation deviation from the target state below a distance of 5 metres is threatening to a real-life system.

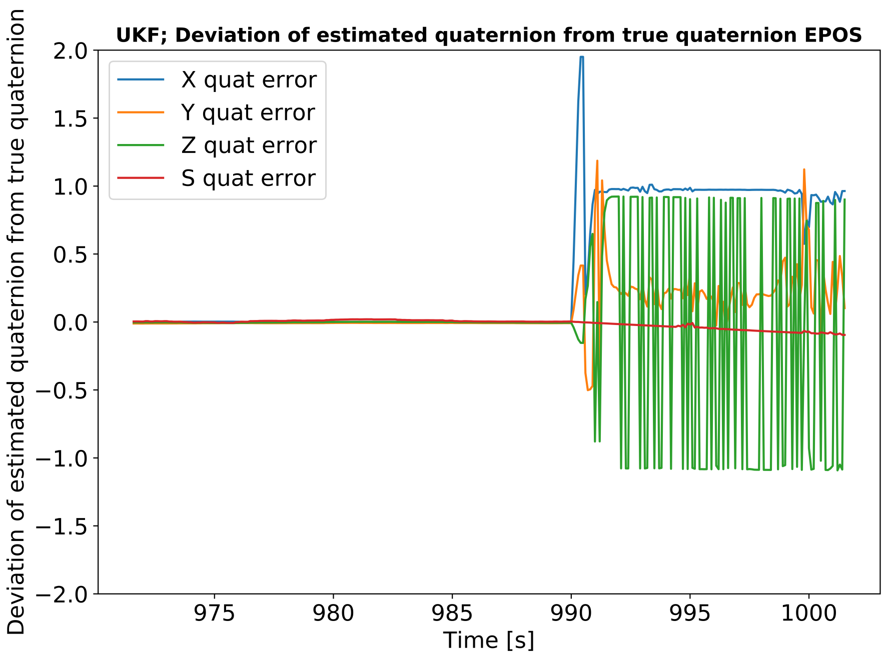


Figure 5.18: Attitude quaternion estimation error of UKF at the end of the test run; error deviation resembles behaviour observed in simulation test

Table 5.9: Comparison of UKF to EKF performance at hold point state estimation

Parameter	error
Position mean error	-2.6 %
Velocity mean error	-158 %
Attitude error x	-2 %
Attitude error y	0.3 %
Attitude error z	-18.4 %

5.5. Judging hardware test performance against performance criteria

Again, as for the simulation test (see Section 4.6), the performance of the different filters is judged against the performance criteria presented earlier. As can be seen in the comparison of the two test approaches and subsequent results, the hardware-in-the-loop test did not confirm the results of the simulation test.Nevertheless, some observations can be related.

As the simulation performance test suggested the UKF performs far better than the EKF in cases where the measurements are delayed. The UKF also has a better position estimate than the EKFS and the EKF in the EPOS 2.0 test, as was already indicated by the simulation performance test. The EKFS does, however, not show the improved position and velocity estimation from the simulation test in the hardware test. Only the attitude estimation performance in case of the EKFS is similar in both tests, where the filter performs poorly compared to the EKF.

Table 5.10: Comparison of UKF to EKF state estimation performance during approach

Parameter	error
Position mean error	14.4 %
Velocity mean error	-53 %
Attitude error x	69.5 %
Attitude error y	9.5 %
Attitude error z	54.6 %

Table 5.11: Rating of filters against performance criteria based on EPOS 2.0 test results

Criterion	Weight	EKF score	EKFS score	UKF score
Accuracy	5	0	-1	+1
Time to settle	3	0	0	0
Stability	4	0	-2	-2
Bias	2	0	-1	0
Complexity	3	0	0	0
Overall	NA	0	-15	-3

The filter performance is judged similarly as in the simulation test data rating:

Accuracy: Again, the accuracy in position estimate, velocity estimate and the average of the attitude estimates is rated.

Time to settle: In the hardware-in-the-loop test this parameter could not be observed satisfactorily, but no substantial differences could be observed. Thus, all filter modes are scored equally.

Stability: Since the EPOS tests showed divergence for both new filters as well as strong quaternion fluctuations late in the test run (primarily for the UKF) they are rated as performing very poorly.

Bias: For the EKFS, a slight offset could be made out in the position estimate.

Complexity: No difference could be made out in the EPOS 2.0 test. All filters ran smoothly and no obvious changes in runtime could be observed. However, this parameter could not be observed conclusively from the test data.

The results from the filter performance rating is shown in Table 5.11.

Having judged the different results from the simulation test and the hardware-in-the-loop test against the filter performance criteria, it can be seen that there are substantial differences between the different test methods and the way they affect filter quality perception (see Tables 4.10 and 5.11). Both filter alternatives perform better in the controlled environment of the simulation performance test. No filter estimate divergence or other unusual estimate behaviour can be observed and both accuracy and time to settle improve. The UKF slightly outperforms the the EKFS due to a faster time to settle. Both filters have a substantially longer runtime, but since it remained under 0.1 seconds of filter execution time using a conventional desktop PC this is not considered a substantial problem.

However, these results weren't confirmed in the EPOS 2.0 tests and the state estimation could not be conclusively be validated. Here, both filter alternatives perform poorer than the original EKF when judged against the performance criteria. In particular the EKFS is downgraded due to its poorer accuracy in estimating the state parameters. Despite it being difficult to rate the stability parameter numerically, both filter alternatives are rated as very poorly due to divergence at close proximity operation. Complexity and time to settle were not rated due to data not being available. However, they are expected to perform in a similar way as in the simulation performance test, thus cancelling out in the overall rating.
From the overall result it can be concluded that neither the EKFS nor the UKF are currently viable alternatives for the EKF, at least not in the way they are currently implemented. Nevertheless, the UKF shows the greatest potential to eventually replace the EKF, since it performed better than the other filters in the simulation performance test and only ranked slightly worse in the EPOS 2.0 test. The problem of estimate divergence needs to be addressed first, though.

In addition, it can be said from the assessment of hold-points and approach stages from the hardware-in-the-loop tests that the EKFS performs worse in both stages, whereas the UKF seems to be particularly suited for approach stage state estimation. This should be confirmed through additional testing.

The differences in ratings from the results of simulation test and hardware test highlight another important conclusion. While the simulation test results on their own suggest that the implementation of the filter alternatives would bring a substantial performance improvement the reality observed in the EPOS 2.0 tests is a different one. This means that simulation tests only give a suggestion of performance but that hardware-in-the-loop tests are crucial to judge a filter and are far more important for filter assessments. While a more accurate model in a simulation test may yield results closer to reality the real-life facility test cannot be neglected. This may also explain the differences between observed filter performance and filter performance found in documentation as presented in [29], since most filters are assessed theoretically or through methods similar to the simulation performance test.

In line with this, the verification method through unit tests needs to be reconsidered, since problems in the state estimation, especially in the attitude estimation of the UKF, were only caught after multiple iterations. This could not be captured by the verification method and was only highlighted through validation in the simulation performance test and the hardware-in-the-loop test. It should thus be reconsidered whether the verification method should be adjusted to include a method to verify the working of the filter across a multitude of iterations to catch such problems earlier in the development phase.

5.6. Explanations and improvement of the testing approach

Several explanations for the difference in performance in the different tests have been given before.

For the UKF, the calculation method of taking the weighted average of state vectors to find propagated states may not work for the quaternions and thus cause divergence. It remains to be explained why this does not seem to be a problem in the hardware-in-the-loop test up to the point of divergence at the end of the test run and why some test cases in the simulation performance test do not show as strong quaternion error divergence as others. In the future, it can be tested whether this causes a problem by re-implementing the computation of quaternions via a different averaging approach (see [21]).

Another observation in the tests is the comparatively poor performance of the EKFS and the UKF (the latter shows a worsening of the velocity estimation) in the hardware test. This may have to do with the far better performance of the EKF in the hardware-in-the-loop test than in the simulation test. This can be explained theoretically by the fact that the EKF test was run longer than the other tests, which caused the EKF to have more time to settle around the target state. Thus, in comparison with the other test runs, a higher percentage of states are estimated that are closer to the target state, which means that the mean performance is better than for the EKFS and the UKF, where tests were run shorter. This could be improved in future projects by either further standardising testing to follow the same timeline, inputs and structure, or by reliable excluding swing-in-periods. Further options for improving the comparability of the tests is shown below in Section 5.7

Another difference in the tests was observed in the overall performance of state estimation being different in the simulation performance test. This could be due to the inconsistencies in the hardware tests and the occurrence of real errors. However, the comparatively different performance may also have to do with the matching propagation methods of the filters and the simulation test. In the simulation performance test, the EKF may have an advantage over the other filters. The propagation of the simulated true servicer and target satellite is performed using the same orbital propagation algorithm as is applied in all filter propagation steps. This means that the propagation of the simulation test perfectly matches the EKF's propagation method, but not the propagation method of the other two filters, since the EKFS and UKF use the same propagation method as a basis but both perform adjustment steps to it, such as the smoothing steps and the determination of sigma vectors. Thus, theory predicts that the EKF would perform better in the simulation test comparatively. This could be tested by using a different propagation method in either the simulation performance test or the filters.

5.7. Comparison of hardware test and simulation test

When comparing the performance of the filters in the hardware test with the performance in the simulation test cases with delayed measurements (which are the most representative compared to the EPOS 2.0 test conditions and are shown in Table 4.9), the poor performance by the UKF in attitude estimation compared

to the EKF in the simulation performance test is standing out and does not compare to the actually better performance in attitude estimation of the UKF in the hardware test. Neither does the UKFs simulated good performance in velocity estimation correlate to the poorer estimation in the hardware test. However, the position estimation accuracy of the UKF matches closely to the position estimation improvement seen by this filter in the hardware test.
Again, when looking at the EKFS's performance in test cases 9 through 14 of the simulation test, they do not match the hardware test performance. The EKFS performs worse than the EKF in the hardware test in all areas, while the simulation test suggested an improvement over the EKF in the estimation of position and velocity.

Furthermore, both new filters showed problems at close-proximity operations. While this wasn't observed in the simulation performance test, where this close distance was not tested either, it highlights the wider issue that neither of the newly implemented filter modes are yet a viable alternative for the existing EKF. This is due to the fact that unforeseen state estimation deviation could lead to catastrophic failure, especially since it occurs in close proximity operation. However, it has to be stated as well, as was seen in the simulation performance test, that both new filters have the potential, and in case of the UKF have shown, that they can outperform the EKF in real life conditions.

This also relates to a great mismatch between simulation testing and hardware testing. The simulation test did not test the approach and hold point scenarios in the same way as the hardware test. This is due to the lack of some of the inputs (mainly the force input to move the servicer satellite relative to the target satellite in the example under investigation) and slight differences in the setup of the satellite models. It can thus be concluded that both the setup and the system inputs into a system need to be modelled the same way as are tested in the hardware test.

It was seen in the hardware test that the measurement delay fluctuated and dramatically increased the closer the servicer satellite moved towards the target satellite. This fluctuation and the higher measurement delays in excess of 0.5 seconds was not modelled in the simulation performance test since it was not foreseen. The occurrence in the hardware test can, however, be explained theoretically from the fact that the target satellite occupies a larger area of the image taken by the sensor the closer the two satellites are. Thus, the image processing algorithm needs to perform increasingly more feature recognition tasks as the servicer approaches the target. This delays the sensor information input to the navigation filter, which in turn may lead to bigger state estimation "jumps" between different iteration steps. This shows that in a simulation test, the inputs to a system or filter can be perfectly controlled but in a hardware test they cannot and might differ dramatically, resulting in different outcomes.

In line with the previous point, the perfect controllability of inputs in simulation testing, another challenge for the transition from simulation to hardware testing was observed to be the consistency across test runs in hardware testing. As was seen, all filters could be fed with perfectly matching input data, initialisation conditions and test scenario sequences in simulation testing. This was not the case in hardware testing, where filters had to have their own dedicated test runs. This meant that both initialisation and input as well as test sequences differed as they were partly human-controlled and not necessarily fully consistent. Thus, while simulation test results may be less representative of real-world performance, they are more suited for direct comparison of filter performance. However, when moving into hardware testing, this also explains why results observed in simulation testing may not be reproduced and may rather give an indication of the real-world performance under the circumstances given in the relevant test runs. Thus, in order to perform performance comparison based on results from hardware testing, the consistency or precise awareness of circumstances in the test runs is far more crucial than for simulation testing.
This also means that in order to compare the performances of systems in simulation tests and hardware tests, not only a general assessment of the differences between the simulation and the hardware facility testing approach needs to be conducted (as was done in this project), but a precise assessment of the differences between the simulation and each individual test run in the hardware test is necessary.
An alternative for more consistent hardware testing would be to only track the real life behaviour, measurements and filter inputs throughout a test run and not use filter feedback to steer the satellites. Then, saving all these parameters and externalising them, the filters can be fed with these inputs and the state estimation from each filter based on the same inputs can be compared to the true state collected before. Thus, the fil-

ters could be tested using the hardware facilities advantages but also using the same inputs for comparability.

As was shown throughout this report, the transition from simulation testing to hardware testing and the definition of expected results from this comparison is also affected by the selection of appropriate test cases. For the project at hand, several test cases were used in the simulation test which were known to not be fully relatable to the hardware tests. Thus, a separate comparison was conducted for the test cases that were deemed more appropriate. The knowledge of such test cases was taken from experts knowledge of test runs. However, simulation testing for future research projects, and the transition from simulation to hardware in particular, can be improved by iteratively adjusting the simulation testing approach using observations from hardware testing. This will make the two test approaches more comparable and the definition of expected outcomes from hardware testing can be facilitated, while at the same time the simulation models are improved.

Several differences between the performance of filters in simulation and hardware testing also arise from erratic behaviour and random errors in the test systems. This can, for example, explain fluctuations in the accuracy of the state estimation of navigation filters. An example of such a behaviour can be random noise on measurements. In the simulation test, noise is randomly generated within certain bounds. However, this random behaviour may be different in real-world applications and test facilities. A detailed assessment of the origin, the frequency and the magnitude of such erratic effects should be conducted to either adjust the simulation model or to exclude the effects from the performance assessment and the comparison between simulation and hardware tests.

The section above discusses the mismatches between simulation testing and hardware testing, and which steps can be taken to either correct for, overcome or understand such mismatches. Since the hardware test is used to capture behaviour that is not observed in simulation testing, the approaches shown above can be used to either improve the simulation model or to better understand the origin of unrealistic behaviour in the simulation model. This in turn can be used to improve the theory of whatever system is tested to make it better suited for real-world applications.

5.8. Conclusion and recommendations for future tests

The analysis performed on the test results from the hardware in the loop tests have shown several insights. These are reiterated in the following.

Neither the EKFS nor the UKF could consistently outperform the EKF in a real-life hardware in the loop test. The UKF showed better performance except for the performance in the estimation of the velocity of the target satellite, and confirmed the observation from the simulation performance test that the UKF performs better when measurements fed to the filter are delayed or withheld (as is the case in a real-lief test requiring image processing). However, the UKF showed sudden estimation divergence in close proximity operation that could not be recreated and conclusively explained yet. The EKFS also showed such behaviour at an initial start up of the system but never afterwards. For both cases this error has thus not been understood or mitigated.

The EKFS shows poorer performance than the other two filter modes in the state estimation (exception: velocity estimation). This is partially explained through an estimation offset and a worse quality measurement that was fed to the EKFS. However, it also highlights another issue that needs to be addressed in future tests: A standardisation of the test parameters and inputs.
Since the facility is operated by human operators there are always differences between individual test runs testing different navigation filters. This means that filter run times can vary and that different measurements are fed to the filters, which in turn affects the filter estimate and, comparatively, the performance. For future test runs, a system needs to be developed that allows for filters to be compared under more similar circumstances. Additionally, more test runs under similar and different test conditions need to be performed, particularly at close-proximity operation. These should be used to confirm the test results gathered so far and can be used to help improve the performance of the new filters.

The implemented filter modes showed differences in performance between test phases where the satellites were held at a constant relative position and where the servicer satellite approached the target. These were investigated separately. It was found that the EKF performs very well in the case where satellites are moving, however, in this condition, the UKF performs even better (up to the point of divergence at close proximity at the end of the test run). The EKFS performs almost as well as the EKF in the case where the satellites are held at hold-points but far worse when satellites are moving.

Neither the EKFS performance nor the UKF performance could conclusively be validated through the hardware-in-the-loop test. This has several reasons. The EKFS performs according to the expectations, however, the measurements fed to the EKFS were so poor that further tests are suggested to validate the performance conclusively under better test conditions. Furthermore, state estimation deviations were observed once when initialising the filter at close proximity, which has not be explained yet. The UKF showed state estimation deviation for all parameters late in its test run. This was similar as the attitude deviation observed in the simulation test but affected the position estimate as well, meaning that the filter should not be used in real mission applications yet. This means it is not validated yet and further filter testing and improvement is required. This again highlights the conclusion drawn from the simulation test that while the unit tests could be used to verify the working of the filter state estimation and the individual filter functionalities, they were not extensive enough to assess potential problems building up over a multitude of iterations. This is what the simulation test and the hardware test were used for and the results shows that the new filter options are not yet fully ready for implementation in real-life applications.

Furthermore, based on the results from both the simulation test and the hardware-in-the-loop test, several conclusions were drawn on how to explain the differences in test performance theoretically, and how future projects can approach testing to validate the explanations. In addition, general approaches for improving the tests and the compatibility of the test results were presented.

5.9. Addressing the research questions

As part of the research project, the research questions posed in Chapter 1.7 need to be answered. Throughout the project all of them were addressed. However, minor changes presented in the previous chapters affect the answers and the way the questions were tackled. In particular, the sub-sub-questions of sub-question 1 needed to be altered slightly since it was identified that a full implementation of all 6 identified potential filter alternatives for the EKF in the simulation performance test would have exceeded the scope of the project. This is mainly due to the fact that the implementation in the simulation performance test and the EPOS 2.0 facility hardly differs, meaning that the full filter would need to be developed and unit tested in either case. Thus, the narrowing down of filter options to 2 potential filter alternatives was not done quantitatively, as originally planned, but qualitatively. Table 5.12 gives an overview over the Sections in this report where individual research questions were answered and briefly explains the answers themselves.

5.10. Value of results

The value of the research project lies in both the assessment of the viability of the newly implemented filters to serve as alternatives for the exiting EKF, and the identification of mismatches between the simulation and hardware tests and challenges in the transition from simulation to hardware testing. The latter can be used by future research projects to improve their test approaches.

As was shown, the newly implemented filters cannot yet be used in real-life applications as alternatives to the EKF since they both showed problems in close-proximity operations in the hardware-in-the-loop test. These problems need to be resolved first.

Nevertheless, the research performed over the course of this project shows the potential of both alternative filter modes to outperform the EKF in close proximity operation. The state parameter estimation was better especially for the UKF for most test cases. Thus, it is recommended that the development of this filter method is intensified.

In addition, the research presents a comprehensive approach for comparative satellite navigation filter assessment. By presenting performance criteria for close proximity satellite operation and judging them according to their importance to mission success, as well as by presenting several approaches for collecting performance data this work can be used as a baseline study for future comparative filter performance assess-

Table 5.12: Addressing the research questions

Research question:	Answered in section:	Answer
Question 1.1	3.1	EKFS, UKF, EnKF, RBPF, MKF
Question 1.2	2.1	The EKF takes visual sensor inputs and initial state estimates to update state estimations; it can take delayed measurements and assess whether a measurement is appropriate for use or not; it was chosen since it is a well established navigation filter
Question 1.3	2.1	DLR has an existing orbital propagation model that is used in all filters and in the simulation performance test
Question 1.4	1.9.8, 3.2	Readiness to use in real-time, Ease of implementation, use in space applications, novelty of filter, readiness for visual navigation systems
Question 1.5	2.1, 2.4	Filters should be able to run at 10 Hz and output the updated state estimate; all filters should have the same functionalities (accepted inputs, delivered outputs) as the originally used EKF
Question 1.6	4.5	Time constraints required to limit the assessment to the EKFS and the UKF; both EKFS and UKF perform deliver more accurate estimates of the state position and converge to the true state faster; the attitude estimate of the UKF is considerably worse than that of the EKF; both filters show potential to outperform the EKF
Question 1.7	3.2	EKFS and UKF; due to time constraints the trade-off was performed purely based on qualitative criteria
Question 2.1	5.1	The EPOS 2.0 facility is operated through a control room and is run using the DLR GNC system; filters are tested by implementing them in the GNC system using configuration files
Question 2.2	1.9.5, 1.9.8	Accuracy, time to settle, stability of estimates, estimate bias, computational complexity
Question 2.3	4.4, 5.2	The performance of filters against the criteria is assessed using a simulation test and a hardware in the loop test; the relevant parameters are deduced from the state estimates and the true target states as well as filter execution times and measurement accuracy
Question 2.4	5.1	EPOS 2.0 test results cannot currently match the simulation test results since they are generated using different approaches. However, the test results should show the same trends and confirm the overall performance behaviour observed in the simulation test.
Question 2.5	5.2, 5.3	The original filter structure is rewritten as such that an overarching filter class has access to all filter modes; configuration files used to initialise the GNC system are used to access and execute different filter modes using the given input parameters
Question 2.6	1.9.10	The same performance parameters as in Question 2.2 are used
Question 2.7	4.7, 5.8, **??**, **??**	These differ greatly by test case; both filter alternatives show the potential to outperform the EKF in estimate accuracy and execution speed but currently diverge in close-proximity operation in the hardware test
Question 2.8	**??**, **??**	The EKFS performs worse in the EPOS 2.0 test, the UKF performs well in the position estimation in the EPOS 2.0 test but neither of the filters is stable at close-proximity operation
Question 2.9	**??**	From the assessment of hold-point and approach performance it seems that the UKF is particularly suited for approach operations

ments.

Furthermore, the research highlighted a wider problem in satellite navigation filter assessment, which is an over-reliance on simulation results over hardware test results. As was shown, the simulation test method was too simplified in comparison to the EPOS 2.0 test and could not reflect many of the more complex state estimation conditions occurring in the approach and arising from using real-world sensor inputs.

Another problem that the difference between the different test approaches highlights that can be considered for future work on navigation filters in general is the approach taken to hardware testing. In simulation testing, the comparability of tests is ensured since different filters can be initialised and tested using exactly the same input conditions, whereas this is not always possible in real-world applications, especially when using complex systems including satellite models. This needs to be considered before starting testing and test approaches and expected outcomes should be adjusted.

It was shown that the verification method of purely using unit tests on filter functionalities and instantaneous state estimation assessments is not extensive enough to account for problems occurring when performing multiple subsequent iterations. This needs to be accounted for in any filter development at earlier verification steps, rather than in the filter validation through testing.

Several challenges in transitioning from simulation to hardware testing were identified that can be considered by future projects: the need to identify the complete set of input conditions that may be present in hardware testing and the assessment of their effect on the outcome of the test campaign in comparison with results from simulation testing; Ongoing changes in the timing and quality of inputs throughout a hardware test which are simulated as constants or are simplified in simulation testing; The mismatch between different test runs in hardware testing that leads to results not being fully comparable and the resulting need for consistency across test runs.
Throughout this report, several examples of such challenges are collected and potential approaches to validate and tackle them are presented. The research report can thus serve as a blueprint for identifying and approaching challenges in simulation and hardware testing in a research project.

6 Recommendations and conclusion

Based on the insights gained over the course of this project, several recommendations are given in this chapter. In section 6.1, general recommendations are given how the research results can be used to improve future projects and what lessons have been learned. In section 6.2, recommendations on future filter developments are given. Section 6.3 presents the next steps in the process of determining viable filter alternatives to the EKF and section 6.4 finishes the report with conclusions from the project.

6.1. Overall observations based on project outcomes

Several issues occurred throughout conduction the research project at hand. This section summarises insights that were gained specifically around the methods of filter assessment and how the problems that occurred can be avoided in the future. Furthermore, it is reiterated how the insights from the research project can be used to improve the transition from simulation model testing to testing in a hardware facility.

Revision of performance criteria is required throughout project:
Since the performance criteria that are used to judge the filters are defined in the beginning of the project when the methods of filter assessment are outlined, they should be revised once the practicalities of the test methods are clearer. As was seen, the EPOS 2.0 test did not yield clear insights on time to settle and computational complexity since the filters could not easily be executed parallelly and using the exact same test conditions. This was not known upfront but affected the filter comparison presented in Section **??**. A revision and adjustment of filter performance criteria based on the practicalities of gaining filter test data would have been more sensible.

Detailed assessment of compatibility of different assessment method results is necessary:
The research project is based on the assessment of filter performance based on two methods, the simulation test and the hardware-in-the-loop test in the EPOS 2.0 facility. While the simulation performance test was designed and developed in such a way that it can simulate test cases encountered in the EPOS 2.0 facility, it is not advanced enough to yield fully comparable results. Furthermore, the EPOS 2.0 facility can also not fully reproduce the test cases defined for the simulation test. Thus, verification of simulation test results is very difficult using the hardware facility, which means that while performance parameters can be compared based on the individual test cases from the different test methods, there is no direct correlation between the tested scenarios. The simulation performance test thus primarily serves as a performance indicator of what is theoretically possible, whereas the hardware test yields more reliable results on real-life performance. The two methods are not fully compatible.

Purely simulation test based results are unreliable for real-life performance:
Following from the previous point, purely assessing the performance of a filter (or other new systems for that matter) based on a simulation model yields a clear idea of the performance, but, especially for poorer models, should always be followed up with more reliable testing methods, such as hardware tests. The research presented in this paper would have concluded substantially different results on the applicability of the filters in real-life scenarios if the hardware tests hadn't been performed. This is also observed in other publications

that rely heavily on theoretical performance of filters in simulations. In addition, the project detailed in this report would have benefited greatly from more stringent validation efforts which may have improved the quality of the simulation test outputs.
In summary, simulation models and tests should be followed by hardware tests to identify problems that the simulation model may not have captured.

When comparing simulation test results and hardware test results, the selected test cases are crucial:
As was discussed before, if one wants to compare test cases and results, the selection of representative test cases and the reiteration based on hardware test insights is crucial for the simulation testing phase.

6.2. Recommendation for future filter development and testing

The previous chapters outlined the problems that were highlighted through the filter test results. It was seen that both the EKFS and the UKF show a potential to outperform the EKF, but this performance could not be confirmed in all test cases and in the EPOS 2.0 test. This section aims to explain what could be improved in the filter development to avoid problems in the test process.
Furthermore, an important aspect of this research project is the identification of concrete mismatches and challenges when transitioning from simulation to hardware testing.

6.2.1. Lessons learned during filter development

Several recommendations can be given based on the lessons learned during the filter development: **Identification of key performance functionalities and their implementation:**
The original EKF has several key functionalities that were identified early on in the filter alternative development process. These include the inputs and output as well as all returns that need to be passed to the GNC system (see Section 2.4 for details). In addition, the original filter features functionalities in excess of a conventional EKF that were developed specifically for DLR needs and needed to be transferred into the new filter alternatives, such as the ability to include delayed measurements and neglect measurements that have been used in previous filter iterations. While there were identified as important early on their implementation wasn't specifically considered when implementing the new filter alternatives. Thus, it wasn't realised that the implementation of the delay function in the EKFS and the UKF actually required a deviation from the documented EKFS and UKF. Not realising this early on caused test errors and development delays. It is thus recommended for future filter developments to carefully adjust filters to include additional functionalities and to check how the documented filter may need to change.

Start hardware testing earlier in development process:
As was mentioned before, the simulation performance test was developed as a quick way to test filter capabilities and determine whether a filter could potentially be used in hardware facilities. While this worked well in theory and, especially early in the development process, errors in the filters could be identified, a great reliance on the simulation test as an actual performance indicator representative of the real-life test conditions pushed back the actual hardware test to a very late stage of the process. This meant that filters were adjusted based on simulation test results and not based on hardware test results since they came too late in the project. Thus, the divergence of the EKFS and UKF state estimation observed in the EPOS 2.0 facility tests could not be addressed through extensive testing and filter adjustments any more. It is thus recommended that in a filter implementation activity, hardware tests should ideally be performed earlier in the project and it should not primarily be relied on simulation tests.

6.2.2. Recommendations based on test performance

Based on the way the tests were conducted and from the assessment of the results, several recommendations can be given in general and for the transition between the different tests:

Make test runs comparable:
Despite the efforts to recreate similar test conditions in the EPOS 2.0 facility tests there were obvious differences in the filter tests that make comparative data assessment difficult. The simulation test on the other hand delivered comparable results since the same inputs were fed to the filter modes in parallel. While this was not possible for the hardware-in-the-loop test, a more strictly controlled test procedure or the capability

to run filters in parallel would benefit the comparability of filter performance and the accuracy of a comparable data assessment. Another approach would be to collect test run data and inputs from the EPOS 2.0 facility first and then run the filters all using this data.
Furthermore, longer and more filter tests in the hardware facility would reduce the impact of outliers or faulty data and thus yield a more reliable data assessment. Time limitations in the project at hand didn't allow for more tests, but future development projects should incorporate more time. This again roots back to an over-reliance on the accuracy of the simulation test results since time needed to address errors in the hardware test was not allocated.

Ensure compatibility of filter method with state parameters:
This is a problem that is postulated to have affected the UKF performance greatly. It was identified that the propagation method of the UKF, which relies on taking weighted averages of propagated sigma vectors, may not be compatible with the quaternions used in the satellite state to indicate the attitude. Mean quaternions cannot be calculated using weighted averages but have their own computational rules. However, this was realised too late for correction. It is postulated that this affected the divergence observed in the EPOS 2.0 facility tests when satellites operated in close distance to each other, as well as the performance in some of the simulation performance test cases. It should thus be checked earlier in the filter development whether the filter being implemented may need adjustments based on the produced output (or state). A specific revision of computational rules for all state parameters is recommended. Again, more stringent verification and earlier validation could help to avoid such problems as well.

Identify all input conditions and their potential effect that may be neglected in simulation testing but are present in hardware testing:
As was mentioned earlier, the simulation test is often a simplified version of a real-world application. It may thus neglect some of the inputs or environmental conditions that have an effect on the system in the hardware test. An assessment of such inputs (for example force inputs, tumbling) and the expected effect on the system response is required to perform a better comparison of test results and understand mismatches between the test outcomes.

Identify constant conditions modelled in the simulation test that are not truly constant in hardware testing:
As was seen, several inputs in the simulation test, such as the frequency with which measurements were passed to the filters, were modelled as constants, while it was observed in the hardware tests that these were not truly constant. Furthermore, the EPOS 2.0 facility, being more complex than the simulation test, involves other factors such as differences in sensor information errors, which divert the performance in the hardware test from that in the software test.

6.3. Next steps

Summarising the lessons learned and the recommendations given above, it can be said that the viability of the selected filter alternatives is not shown conclusively. The next steps in the process of determining viable filter alternatives for the EKF are shown here. These can also be used by future research projects to improve their own theory and implementation in software and hardware tests.

Improving the simulation performance test:
The simulation performance test should be adjusted to match the real-life conditions of filter testing more closely. One poignant example that was identified is the change in propagation algorithm in the simulation to not match the filter propagation method and to be more accurate than the current method.

Perform stability tests for all filters:
DLR has developed a stability check for the original EKF which tests whether filters access the correct stored state to compute measurement predictions when measurements are delayed. Since stored states do not necessarily match the time stamp of the delayed incoming measurement, past states need to be interpolated. It was observed that some interpolation cases can cause estimate divergence under specific conditions. DLR is currently working on a paper discussing this issue and has been able to fix it for the EKF.

Further research in how to stabilise the EKFS and UKF is required, and may lead to a better understanding of why the filters diverge at close-proximity operation.

Further hardware-in-the-loop tests:
As was discussed, the EPOS 2.0 tests were not all performed in such a way that they were easily comparable. In addition, only one test run per filter was fully performed under real-operating conditions (earlier tests used a different, faulty sensor feedback approach and were neglected for filter quality assessment), which means that more test runs are required to expand the data set and confirm the current results.

Average quaternion prediction adjustment:
It was previously explained that the part of the satellite state using quaternions for attitude determination in the UKF (and potentially the EKFS as well) may require a different method for drawing the mean weight estimate than is the case for the other state parameters. This is due to the quaternions not working with conventional weighted averages. A separate approach needs to be implemented. This may correct the quaternion deviation, and, consequently also improve the close-proximity operation of the state estimate overall.

Re-evaluate filter performance scoring:
The initial performance criteria were determined and rated based on purely theoretical knowledge at the start of the project to allow for an unbiased evaluation of the performance of the different filters. However, as was seen in the test process and the subsequent performance assessment the scoring may not always be representative of the actual usefulness of the filter in an application. As was seen with both the EKFS and the UKF, the stability of a filter and the range of application has to be rated higher and rating criteria have to be defined more clearly. This allows for excluding filter options that do not perform up to minimum requirements or at least scores them as considerably worse than the existing option. Furthermore, by assessing the performance requirements on the existing filter in existing mission specifications more thoroughly, performance criteria can be defined more clearly, in particular for numerical filter performance assessment.

Re-perform tests according to revised test approach and performance assessment:
Finally, the tests that have already been performed need to be repeated and added to. Thereafter, the revised performance assessment can be undertaken in order to achieve a more accurate filter viability comparison.

6.4. Conclusion

This report details a research project that was conducted as part of a master thesis at TU Delft at the faculty of Aerospace Engineering in the space engineering department. The purpose of the project was to determine the viability of 2-3 potential navigation filter alternatives to the Extended Kalman Filter in close-proximity satellite operation. This was done by qualitatively selecting two filters, an extended kalman filter with an intermediate smoothing step and an unscented kalman filter, for implementation and testing within the GNC system framework at DLR Oberpfaffenhofen. A simulation test was built to assess the theoretical performance of the filters and numerically compare them. 16 test cases were performed using the same test conditions and inputs for all filters. Furthermore, the filters were assessed through test data collected from approach scenarios performed in the hardware-in-the-loop facility EPOS 2.0. In addition, the purpose of the research was to uncover mismatches between simulation and hardware testing. While the transition from simulation to hardware testing may seem straightforward, it was found that there are several general problems that can be encountered. This report can thus serve as a blueprint to improve this transition for future projects.

Neither of the two newly implemented filters are, at this time and under the current implementation, viable for replacing the currently used EKF in real-world applications since their correct working could not be conclusively validated in the hardware tests. They both showed state estimation divergence in close-proximity operation below a relative distance of 5 metres between the target satellite and the servicer satellite, which was deemed a critical criterion for exclusion in real-world applications since it could lead to catastrophic failure, particularly in close proximity operation.
Nevertheless, both newly implemented filters showed the potential to outperform the original EKF in the estimation of the state parameters in both the simulation performance test and the hardware-in-the-loop test prior to divergence, in particular in the estimation of position and velocity of the target satellite. For the simulation performance test, it was decided to focus the performance assessment on test cases with delayed

measurements of which the update frequency did not necessarily match the update frequency of the filters. This was done since it was a realistic test scenario observed in real-world applications. Both the EKFS and the UKF were able to outperform the EKF in the estimation of the position and velocity by about 12% (position) and 14% and 24% (velocity, respectively).
In the hardware-in-the-loop-test neither filter could repeat this performance improvement. This had several observed reasons. The EKFS performed worse overall in the state estimation. All estimated state parameters did not match the true state of the estimated target satellite as well as in the case of using the original EKF. This was partly due to a poorer quality of the measurement that was fed to the EKFS. Due to the nature of the testing approach in the EPOS 2.0 facility it was not possible to run filters in parallel with exactly the same measurements (and thus measurement quality) which is why this inconsistency could not be corrected for. The UKF improved in the mean position estimation (10% improvement) and the attitude estimation compared to the EKF but greatly decreased in the accuracy of the velocity estimate. This had both to do with the EKF performing very well in this parameter estimation (thus resulting in a strong baseline which the new filters were compared to) as well as other inconsistencies in the testing approach. For example, it was found that the duration of test runs and corresponding longer settling times of filters in longer test runs would decrease the average estimation error. Since the EKF was tested the longest this could have affected the perceived performance in comparison by the other filters. It was concluded that hardware-in-the-loop tests need to be standardised better in the future to allow for fairer comparison (see below).

Not only the quality of the state estimation was assessed during the project. From the simulation test it could be seen that both the EKFS and the UKF were able to converge faster than the EKF to within the same convergence boundaries, meaning that they achieve the same accuracy quicker, which is important in close proximity operation. From the hardware-in-the-loop test it was seen that the UKF outperformed both the EKF and the EKFS prior to divergence. Fast state estimation convergence is critical for close proximity operation since it is crucial to achieve a precise estimate fast and for it to remain precise when satellites operate closely to each other.

Based on its potential for superior performance it was concluded that the UKF is the most viable alternative for the EKF. However, before it can be used as an alternative in real-life applications the divergence problems at close-proximity operation needs to be rectified and a hardware-in-the-loop facility filter validation needs to take place. The primary areas that need to be investigated are the implementation of the quaternion estimation from the weighted average that is drawn in the UKF to determine the state estimate as it was found after the testing that this approach may not work with the quaternions. Nevertheless, the UKF showed the most potential based on its test results. It was thus concluded that the UKF was the most promising alternative to the EKF once the divergence problems are mitigated, since it showed better accuracy and faster convergence, thus dominating two of the most important stated performance criteria. This was particularly observed in the approach phase in the hardware test, where more accurate state estimation results were achieved.

Throughout the implementation and testing process, several other conclusions were drawn on how to improve the approach to determine the viability of satellite navigation filters.

- While the simulation performance test can fairly assess the performance of the individual filters since it feeds the same information to all filters using the same initialisation for test cases, the hardware-in-the-loop test does not have this capability. Thus, the duration, approach scenario and the quality of measurement throughout the test of each individual filter differs. This was particularly observed in the EKFS test, where the measurement quality was very poor. The capability of the hardware test thus needs to be adjusted to be able to feed comparable inputs to the filters to allow for a fair assessment.

- The simulation performance test were not used to simulate tumbling or force inputs to the modelled satellites. Furthermore, no approach representative of the observed scenario in the EPOS 2.0 facility was performed. To make the results from both tests more comparable, these capabilities need to be added.

- The means of verification were concluded to not be sufficient for the project at hand. The primary method of verification of the filter functionalities was through unit tests that tested the individual functionalities of the filters as well as their interaction in determining a state estimate. This represents a single iteration of the filter, or, at most, a succession of only a few subsequent filter iterations. All these functionalities up to a system level could be verified. However, this verification could not account for accumulating errors across multiple iterations and thus either does not verify all relevant parameters throughout the state estimation or needs to be extended by other methods.

- It was concluded that an over-reliance on the simulation test results could negatively impact the legitimacy of the viability assessment. As was seen, the filters performed differently in the simulation and the hardware test. It was thus concluded for future navigation filter assessments that different verification strategies and simulation testing methods may be required to deliver more conclusive results.

It can be said conclusively that more tests need to be performed, in particular in the hardware facility, and that the newly implemented filters need to be assessed further in close proximity operations to make them viable alternatives to the EKF. For the moment, they cannot be used as viable alternatives for the EKF in real-world applications. The value of this research lies in showing that both the EKFS and the UKF show the potential to outperform the EKF but need further assessment regarding the implementation, and in highlighting the need for parallel hardware-in-the-loop testing of filters to judge filters fairly in addition to conventional simulation testing.

Throughout testing, several aspects were observed that differed between the simulation and hardware testing, causing mismatches. The purpose of the hardware test was to identify and capture conditions of performance that were not observed in the simulation tests. The conclusions drawn from this are presented here.

- The selection of comparable test cases is crucial when comparing the performance of a system in simulation and hardware testing. As was seen, not all test cases in the simulation test relate to the hardware test conditions and thus do not yield meaningful insight on a system under simulated real-world conditions. Thus, while such test cases may be useful to assess theoretical performance parameters, they are not useful for the specific application at hand when it can be shown that they do not relate to the real-world. This means that simulation test cases should be adjusted to, or reiterated, following real-world observations if the purpose is to yield meaningful insight on real-world behaviour in a simulation test.

- The identification of all possible inputs in the hardware test that may affect performance assessments but are neglected in the simulation test is important before and throughout hardware testing. This helps to assess the potential impact on the performance that is tested. An example is the fluctuation of errors and force inputs.

- Being aware of changing conditions throughout a hardware test run that are modelled as constants in the simulation test, or are simply initialised in the simulation and do not change dynamically, is critical to understand changing performance across a hardware facility test. Throughout this project, the outstanding parameter was the timing of the measurement information input to the filter, which was modelled as a constant in the simulation test but became drastically slower in the hardware test the closer the servicer satellite got to the target satellite. This, naturally, causes the state estimation to perform bigger jumps between update steps and may affect conversion quality. Being aware of this allows to adjust the simulation models of future projects.

A
Simulation test data analysis code

This appendix shows the code used to perform the simulation performance test data assessment. Code samples to plot graphs are excluded. Separate data frames were used to assess individual state parameters. The data assessment program is coded in Python.

The assessment is show for the EKF but is performed in the same way for the other filters. The extracted data is saved in a csv file separately.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
from openpyxl import load_workbook

## Define function for system assessment
def systemAssess(df_f):
## DETERMINE BOUNDARY POINTS FOR ASSESSING FLUCTUATION AND CONVERGENCE TIME
## determine high points
df_f['h_point_check'] = ""

for index, rows in df_f.iterrows():
if (index == 0):
df_f.iloc[0,-1] = 0
elif (index == df_f.index[-1]):
df_f.iloc[index,-1] = 0
else:
if (df_f.iloc[index,-2] > df_f.iloc[index - 1,-2] and df_f.iloc[index,-2] > df_f.iloc[index +
    1,-2]):
df_f.iloc[index,-1] = 1
else:
df_f.iloc[index,-1] = 0
df_f_hp = df_f.loc[df_f['h_point_check'] == 1]
df_f_hp = df_f_hp.reset_index()
df_f_hp['first_min'] = ""
abs_diff_loc = int(df_f_hp.columns.get_loc("abs_diff_pos"))

## determine lowpoints
for index, rows in df_f_hp.iterrows():
if (index == 0):
df_f_hp.iloc[index,-1] = 0
if (index == df_f_hp.index[-1]):
df_f_hp.iloc[index,-1] = 1
else:
```

```
if (df_f_hp.iloc[index,abs_diff_loc] < df_f_hp.iloc[index - 1,abs_diff_loc] and
    df_f_hp.iloc[index,abs_diff_loc] < df_f_hp.iloc[index + 1,abs_diff_loc] and
    df_f_hp['abs_diff_pos'].iloc[(index+1)::].max() <
    df_f_hp['abs_diff_pos'].iloc[::(index+1)].max()):
df_f_hp.iloc[index,-1] = 1
else:
df_f_hp.iloc[index,-1] = 0
if (len(df_f_hp.index) == 0):
conv_pnt_idx = 0
conv_pnt_time = "no conversion reached within time period"
bnd_pnt_time = "no boundary point reached within time"
bnd_pnt_err = "-"
max_tail = "-"
bnd_tail = "-"
else:
conv_pnt_idx = df_f_hp['first_min'].idxmax()
conv_pnt_time = df_f_hp.iloc[conv_pnt_idx,1]

######### find boundary point
# convergence point is highpoint after which higher highpoints follow
# boundary point is the point before conv point after which only smaller values
# first determine max point after conv point, then check last occurence for which total error
    higher
max_tail = df_f_hp['abs_diff_pos'].iloc[(conv_pnt_idx+1)::].max()
df_f_hp['first_max'] = ""
for index, rows in df_f_hp.iterrows():
if (index == df_f_hp.index[-1]):
df_f_hp.iloc[index,-1] = 1
else:
if (df_f_hp.iloc[index,abs_diff_loc] > max_tail and df_f_hp.iloc[index + 1,abs_diff_loc] <
    max_tail):
df_f_hp.iloc[index,-1] = 1
else:
df_f_hp.iloc[index,-1] = 0
bnd_pnt_idx = df_f_hp['first_max'].idxmax()
bnd_pnt_time = df_f_hp.iloc[bnd_pnt_idx,int(df_f_hp.columns.get_loc("est_time"))]
bnd_pnt_err = df_f_hp.iloc[bnd_pnt_idx, abs_diff_loc]
bnd_tail = df_f['abs_diff_pos'].iloc[int(bnd_pnt_time)::]
return (conv_pnt_time, bnd_pnt_time, bnd_pnt_err, max_tail, bnd_tail, df_f_hp);

######### determine accuracy of position estimate
def posDataFrame(df_f):
df_return = df_f
df_return['x_diff_pos'] = df_return['est_px'] - df_return['true_px']
df_return['y_diff_pos'] = df_return['est_py'] - df_return['true_py']
df_return['z_diff_pos'] = df_return['est_pz'] - df_return['true_pz']
df_return['abs_diff_pos'] = (df_return['x_diff_pos']**2 + df_return['y_diff_pos']**2 +
    df_return['z_diff_pos']**2)**(1./2.)
return df_return

######### determine accuracy of velocity estimate
def velDataFrame(df_f):
df_return = df_f
df_return['x_diff_vel'] = df_f['est_vx'] - df_f['true_vx']
df_return['y_diff_vel'] = df_f['est_vy'] - df_f['true_vy']
df_return['z_diff_vel'] = df_f['est_vz'] - df_f['true_vz']
df_return['abs_diff_vel'] = (df_return['x_diff_vel']**2 + df_return['y_diff_vel']**2 +
    df_return['z_diff_vel']**2)**(1./2.)
return df_return

######### determine accuracy of attitude estimate
```

```
def quatDataFrame(df_f):
df_return = df_f
df_return['x_diff_q'] = df_f['est_qx'] - df_f['true_qx']
df_return['y_diff_q'] = df_f['est_qy'] - df_f['true_qy']
df_return['z_diff_q'] = df_f['est_qz'] - df_f['true_qz']
df_return['s_diff_q'] = df_f['est_qs'] - df_f['true_qs']
return df_return

######### determine accuracy of attitude rate estimate
def rateDataFrame(df_f):
df_return = df_f
df_return['x_diff_w'] = df_f['est_wx'] - df_f['true_wx']
df_return['y_diff_w'] = df_f['est_wy'] - df_f['true_wy']
df_return['z_diff_w'] = df_f['est_wz'] - df_f['true_wz']
return df_return

######### determine time to stay within same position accuracy as original EKF
def timeToBfBndPnt(df_f_hp, bnd_pnt):
for index, rows in df_f_hp.iterrows():
if (index == 0 and df_f_hp.iloc[index,int(df_f_hp.columns.get_loc('abs_diff_pos'))] <
    bnd_pnt):
return df_f_hp.iloc[(index+1),int(df_f_hp.columns.get_loc('est_time'))]
break
elif (index == df_f_hp.index[-1]):
return ('No conversion to within comparative bounds')
break
elif (df_f_hp.iloc[index,int(df_f_hp.columns.get_loc('abs_diff_pos'))] > bnd_pnt and
    df_f_hp.iloc[index + 1,int(df_f_hp.columns.get_loc('abs_diff_pos'))] < bnd_pnt and
    df_f_hp['abs_diff_pos'].iloc[(index+1)::].max() < bnd_pnt):
return df_f_hp.iloc[(index+1),int(df_f_hp.columns.get_loc('est_time'))]
break
else:
continue

######### conversion from quaternions to Euler angles
def quatToEuler(x, y, z, s):
t0 = 2.0 * (s * x + y * z)
t1 = 1.0 - 2.0 * (x * x + y * y)
xangle = math.degrees(math.atan2(t0, t1))
t2 = 2.0 * (s * y - z * x)
t2 = 1.0 if t2 > 1.0 else t2
t2 = -1.0 if t2 < -1.0 else t2
yangle = math.degrees(math.asin(t2))
t3 = 2.0 * (s * y + x * y)
t4 = 1.0 - 2.0 * (y * y + z * z)
zangle = math.degrees(math.atan2(t3, t4))
return xangle, yangle, zangle

######### determine accuracy of Euler angles
def quatAngleDataFrame(df_f):
df_return = df_f
df_return['x_angle'] = ""
df_return['y_angle'] = ""
df_return['z_angle'] = ""
for index, rows in df_return.iterrows():
x, y, z =
    quatToEuler(df_return.iloc[index,int(df_return.columns.get_loc("x_diff_q"))],df_return.iloc[index,int(df_retur
df_return.iloc[index, int(df_return.columns.get_loc("x_angle"))] = x
df_return.iloc[index, int(df_return.columns.get_loc("y_angle"))] = y
df_return.iloc[index, int(df_return.columns.get_loc("z_angle"))] = z
return df_return
```

```
######### determine tail section after set swing-in time
def determineSwingTail(df, tailStart):
#determine filter dt
filter_dt = df.iloc[-1,int(df.columns.get_loc('est_time'))] -
    df.iloc[-2,int(df.columns.get_loc('est_time'))]
# delete all rows in sdInLog where time is smaller than filter_time_min
indexNames = df[df["est_time"] < (tailStart - filter_dt)].index
df_tail = df.drop(indexNames)
return df_tail

######### determine general error characteristics for a data frame
def errorAssessment(df_err):
sum_df_err = sum(df_err['abs_diff_pos'].abs())
print('Mean error in bounds ',round(df_err['abs_diff_pos'].mean(),5))
print('Median error in bounds ',round(df_err['abs_diff_pos'].median(),5))
return sum_df_err, round(df_err['abs_diff_pos'].abs().mean(),5),
    round(df_err['abs_diff_pos'].abs().median(),5)

######### determine errors of state parameters apart from position
def remainErrorAssess(vel_df, ang_df, rate_df):
vel = round(vel_df['abs_diff_vel'].abs().mean(),5)
x_ang = round(ang_df['x_angle'][ang_df['x_angle']<10.][ang_df['x_angle']>-10.].abs().mean(),5)
y_ang = round(ang_df['y_angle'][ang_df['y_angle']<10.][ang_df['y_angle']>-10.].abs().mean(),5)
z_ang = round(ang_df['z_angle'][ang_df['z_angle']<10.][ang_df['z_angle']>-10.].abs().mean(),5)
x_w   = round(rate_df['x_diff_w'].abs().mean(),5)
y_w   = round(rate_df['y_diff_w'].abs().mean(),5)
z_w   = round(rate_df['z_diff_w'].abs().mean(),5)
return vel, x_ang, y_ang, z_ang, x_w, y_w, z_w

######### determine errors of attitude in quaternions
def quatErrorAssess(q_df):
qx = round(q_df['x_diff_q'].abs().mean(),5)
qy = round(q_df['y_diff_q'].abs().mean(),5)
qz = round(q_df['z_diff_q'].abs().mean(),5)
qs = round(q_df['s_diff_q'].abs().mean(),5)
return qx, qy, qz, qs

######### convert errors to percentage errors
def improvementAssessment(df_comp, df):
improv = round((1. - (df_comp)/df)*100,2)
print('% improvement to BF', round((1. - (df_comp)/df)*100,2),'%')
return improv

########################### ASSESSMENT ######################################
#load in data into data frame
TestCase = 1
swingTime = 10.0

for TestCase in range(1,17):

# read in EKFS, EKF, UKF
df_ekfs = pd.read_csv('./Data/TestCases/TestCase'+str(TestCase)+'/EkfsOutLog.csv',
    delimiter=';')

df_bf = pd.read_csv('./Data/TestCases/TestCase'+str(TestCase)+'/BfOutLog.csv', delimiter=';')

df_ukf = pd.read_csv('./Data/TestCases/TestCase'+str(TestCase)+'/UkfOutLog.csv',
    delimiter=';')

#### add relevant columns with error computation
```

```
df_ekfs = posDataFrame(df_ekfs.copy())
df_bf = posDataFrame(df_bf.copy())
df_ukf = posDataFrame(df_ukf.copy())

#######velocity framework
df_ekfs_vel = velDataFrame(df_ekfs.copy())
df_bf_vel = velDataFrame(df_bf.copy())
df_ukf_vel = velDataFrame(df_ukf.copy())

####### quaternion framework
df_ekfs_q = quatDataFrame(df_ekfs.copy())
df_bf_q = quatDataFrame(df_bf.copy())
df_ukf_q = quatDataFrame(df_ukf.copy())
df_ekfs_q_ang = quatAngleDataFrame(df_ekfs_q.copy())
df_bf_q_ang = quatAngleDataFrame(df_bf_q.copy())
df_ukf_q_ang = quatAngleDataFrame(df_ukf_q.copy())

####### attitude rate framework
df_ekfs_w = rateDataFrame(df_ekfs.copy())
df_bf_w = rateDataFrame(df_bf.copy())
df_ukf_w = rateDataFrame(df_ukf.copy())

############################## QUALITY ASSESSMENT ###################################
##sum of total error, first quality check for filter
sum_df_bf = sum(df_bf['abs_diff_pos'])
sum_df_ekfs = sum(df_ekfs['abs_diff_pos'])
sum_df_ukf = sum(df_ukf['abs_diff_pos'])

## DETERMINE BOUNDARY POINTS FOR ASSESSING FLUCTUATION AND CONVERGENCE TIME POSITION
# filter assessment of EKF
swingTail_bf = determineSwingTail(df_bf.copy(), swingTime)
total_err_bf, mean_err_bf, median_err_bf = errorAssessment(df_bf.copy())

sT_bf_vel = determineSwingTail(df_bf_vel.copy(), swingTime)
sT_bf_ang = determineSwingTail(df_bf_q_ang.copy(), swingTime)
sT_bf_rate = determineSwingTail(df_bf_w.copy(), swingTime)

swingtail_err_bf, swingtail_mean_err_bf, swingtail_median_err_bf =
    errorAssessment(swingTail_bf.copy())

# deterime errors that are not in position
mean_vel_bf, ang_x_bf, ang_y_bf, ang_z_bf, w_x_bf, w_y_bf, w_z_bf =
    remainErrorAssess(sT_bf_vel.copy(),sT_bf_ang.copy(),sT_bf_rate.copy())
qx_bf, qy_bf, qz_bf, qs_bf = quatErrorAssess(df_bf_q)

conv_pnt_time_bf, bnd_pnt_time_bf, bnd_pnt_err_bf, maxtail_bf, bnd_tail_bf, df_f_hp_bf =
    systemAssess(df_bf.copy())
if (len(df_f_hp_bf.index) > 0):
print ('Time to convergence point basic filter :', conv_pnt_time_bf,' sec')
print ('Time to boundary point basic filter :', bnd_pnt_time_bf,' sec')
print ('Boundary point estimation error to true target basic filter :',
    round(bnd_pnt_err_bf,5),' meters')
print ('Maximum deviation of a highpoint from true after boundary point :',
    round(maxtail_bf,5),' meters')
df_xls_fill_bf = pd.DataFrame(np.array([[conv_pnt_time_bf, bnd_pnt_time_bf,
    round(bnd_pnt_err_bf,5), round(maxtail_bf,5)]]))
else:
print("No conversion reached within time period, extend time of analysis")
```

Bibliography

[1] Martin Adams, Wijerupage Sardha Wijesoma, and Andrew Shacklock. Autonomous Navigation : Achievements in Complex Environments. (June), 2007.

[2] Franz Andert, Nikolaus Ammann, Stefan Krause, Sven Lorenz, Dmitry Bratanov, and Luis Mejias. Optical-Aided Aircraft Navigation using Decoupled Visual SLAM with Range Sensor Augmentation. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 88(2-4):547–565, 2017. ISSN 15730409. doi: 10.1007/s10846-016-0457-6.

[3] Heike Benninghoff, Toralf Boge, and Florian Rems. Autonomous Navigation for On-Orbit Servicing. *KI - Künstliche Intelligenz*, 28(2):77–83, 2014. ISSN 0933-1875. doi: 10.1007/s13218-014-0297-0. URL `http://link.springer.com/10.1007/s13218-014-0297-0`.

[4] Heike Benninghoff, Florian Rems, and Toralf Boge. Development and hardware-in-the-loop test of a guidance, navigation and control system for on-orbit servicing. *Acta Astronautica*, 102:67–80, 2014. ISSN 00945765. doi: 10.1016/j.actaastro.2014.05.023. URL `http://dx.doi.org/10.1016/j.actaastro.2014.05.023`.

[5] Heike Benninghoff, Florian Rems, and Toralf Boge. End?to?end simulation and verifcation of GNC and robotic systems considering both space segment and ground segment. *CEAS Space Journal*, 2017. doi: 10.1007/s12567-017-0192-2. URL `https://doi.org/10.1007/s12567-017-0192-2`.

[6] Jose Antonio Boluda and Fernando Pardo. A reconfigurable architecture for autonomous visual-navigation. *Machine Vision and Applications*, 13(5-6):322–331, 2003. ISSN 09328092. doi: 10.1007/s00138-002-0078-x.

[7] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 53(3):263–296, 2008. ISSN 09210296. doi: 10.1007/s10846-008-9235-4.

[8] Eun-jung Choi, Jae-cheol Yoon, Byoung-sun Lee, and Sang-young Park. Onboard orbit determination using GPS observations based on the unscented Kalman filter. *Advances in Space Research*, 46(11):1440–1450, 2010. ISSN 0273-1177. doi: 10.1016/j.asr.2010.07.022. URL `http://dx.doi.org/10.1016/j.asr.2010.07.022`.

[9] Pingyuan Cui, Xizhen Gao, Shengying Zhu, and Wei Shao. Visual navigation using edge curve matching for pinpoint planetary landing. *Acta Astronautica*, 146(November 2017):171–180, 2018. ISSN 00945765. doi: 10.1016/j.actaastro.2018.02.033. URL `https://doi.org/10.1016/j.actaastro.2018.02.033`.

[10] R. Van der Merwe and E. A. Wan. The square-root unscented kalman filter for state and parameter-estimation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 6, pages 3461–3464 vol.6, May 2001. doi: 10.1109/ICASSP.2001.940586.

[11] G. N. Desouza and A. C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, Feb 2002. ISSN 0162-8828. doi: 10.1109/34.982903.

[12] Geir Evensen. The ensemble kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53(4):343–367, Nov 2003. ISSN 1616-7228. doi: 10.1007/s10236-003-0036-9. URL `https://doi.org/10.1007/s10236-003-0036-9`.

[13] Lina He, Hairui Zhou, and Gongyuan Zhang. Improving extended Kalman filter algorithm in satellite autonomous navigation. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 231(4):743–759, 2017. ISSN 20413025. doi: 10.1177/0954410016641708.

[14] Daniel Helmick, Anelia Angelova, and Larry Matthies. Terrain Adaptive Navigation for Planetary Rovers. *Archivos Espanoles de Urologia*, 71(5):486–494, 2018. ISSN 00040614. doi: 10.1002/rob.

[15] Christian Ivancsits and Min Fan Ricky Lee. Visual navigation system for small unmanned aerial vehicles. *Sensor Review*, 33(3):267–291, 2013. ISSN 02602288. doi: 10.1108/02602281311324726.

[16] Eagle S. Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research*, 30(4):407–430, 2011. ISSN 02783649. doi: 10.1177/0278364910388963.

[17] Hong Chae Jung. Visual navigation for a mobile robot using landmarks. *Advanced Robotics*, 9(4):429–442, 1994. ISSN 0169-1864. doi: 10.1163/156855395X00490. URL `https://www.tandfonline.com/doi/full/10.1163/156855395X00490`.

[18] Eric Krotkov, Martial Hebert, Lars Henriksen, Paul Levin, Mark Maimone, Reid Simmons, and James Teza. Evolution of a prototype lunar rover: Addition of laser-based hazard detection, and results from field trials in lunar analog terrain. *Autonomous Robots*, 7(2):119–130, Sep 1999. ISSN 1573-7527. doi: 10.1023/A:1008926000060. URL `https://doi.org/10.1023/A:1008926000060`.

[19] Sridhar R. Kundur and Daniel Raviv. A vision-based pragmatic strategy for autonomous navigation. *Pattern Recognition*, 31(9):1221–1239, 1998. ISSN 00313203. doi: 10.1016/S0031-3203(97)00151-9.

[20] Kai Zhou Liu, Jing Li, Wei Guo, Pu Qiang Zhu, and Xiao Hui Wang. Navigation system of a class of underwater vehicle based on adaptive unscented Kalman fiter algorithm. *Journal of Central South University*, 21(2):550–557, 2014. ISSN 21622388. doi: 10.1007/s11771-014-1973-9.

[21] F. Landis Markley, Yang Cheng, John L. Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007. doi: 10.2514/1.28949. URL `https://doi.org/10.2514/1.28949`.

[22] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation: performance evaluation and enhancement. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 433–440 vol.1, Aug 1995. doi: 10.1109/IROS.1995.525832.

[23] ROBERT N. MILLER, EVERETT F. CARTER Jr., and SALLY T. BLUE. Data assimilation into nonlinear stochastic models. *Tellus A*, 51(2):167–194. doi: 10.1034/j.1600-0870.1999.t01-2-00002.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1034/j.1600-0870.1999.t01-2-00002.x`.

[24] Masaaki Mokuno and Isao Kawano. In-Orbit Demonstration of an Optical Navigation System for Autonomous Rendezvous Docking. *Journal of Spacecraft and Rockets*, 48(6):1046–1054, 2011. ISSN 0022-4650. doi: 10.2514/1.52193. URL `http://arc.aiaa.org/doi/10.2514/1.52193`.

[25] B Polle, B Frapard, X Sembely, Astrium Sas, Av Des Cosmonautes, and Toulouse Cédex. Autonomous Navigation Concepts for Interplanetary Missions. *IFAC Proceedings Volumes*, 37(6):203–208, 2004. ISSN 14746670. doi: 10.1016/S1474-6670(17)32174-2. URL `http://dx.doi.org/10.1016/S1474-6670(17)32174-2`.

[26] F Rems, Heike Benninghoff, and E.-A. Risse. Rendezvous GNC-System for autonomous orbital servicing of uncooperative targets. (1):7–8, 2017. ISSN 1529-2401. doi: 22/18/8238[pii].

[27] S. Theil, N. Ammann, F. Andert, T. Franz, H. Krüger, H. Lehner, M. Lingenauber, D. Lüdtke, B. Maass, C. Paproth, and J. Wohlfeil. ATON (Autonomous Terrain-based Optical Navigation) for exploration missions: recent flight test results. *CEAS Space Journal*, 10(3):325–341, 2018. ISSN 18682510. doi: 10.1007/s12567-018-0201-0. URL `https://doi.org/10.1007/s12567-018-0201-0`.

[28] Zhonghua Wang, Chengzhi Deng, Chao Pan, and Jianguo Liu. A visual integrated navigation for precise position estimation. *Computers and Electrical Engineering*, 40(3):897–906, 2014. ISSN 00457906. doi: 10.1016/j.compeleceng.2013.05.011. URL `http://dx.doi.org/10.1016/j.compeleceng.2013.05.011`.

[29] Noel Weber. Review of visual navigation systems and navigation filters for close proximity satellite operations. Technical report, Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands, 2019.

[30] Changxuan Wen and Pini Gurfil. Guidance, navigation and control for autonomous R-bar proximity operations for geostationary satellites. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 231(3):452–473, 2017. ISSN 20413025. doi: 10.1177/0954410016638877.

[31] James R. Wertz. *Spacecraft Attitude Determination and Control*. Springer Netherlands, 1978. ISBN 978-90-277-1204-2. doi: 10.1007/978-94-009-9907-7.

[32] Yan Xu, Haibin Duan, Cong Li, and Yimin Deng. On-board visual navigation system for unmanned aerial vehicles autonomous aerial refueling. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 0(0):095441001774818, 2017. ISSN 0954-4100. doi: 10.1177/0954410017748182. URL `http://journals.sagepub.com/doi/10.1177/0954410017748182`.

[33] Ruigang Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, June 2003. doi: 10.1109/CVPR.2003.1211356.

[34] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015. ISSN 2363-6912. doi: 10.1007/s40903-015-0032-7. URL `http://link.springer.com/10.1007/s40903-015-0032-7`.

[35] ZHE CHEN. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. *Pediatric Nephrology*, 23(5):841–845, 2008. ISSN 00220949. doi: 10.1.1.107.7415. URL `http://citeseerx.ist.psu.edu/`.