Navigation Performance Enhancement Using Online Mosaicking

Vadim Indelman

Navigation Performance Enhancement Using Online Mosaicking

Research thesis

Submitted in Partial Fulfilment of the
Requirement for the degree of
Doctor of Philosophy

Vadim Indelman

Submitted to Senate of
the Technion - Israel Institute of Technology

Nisan, 5771 Haifa April 2011

The research thesis was done under the supervision of Prof. Pini Gurfil, Prof. Ehud Rivlin and Dr. Hector Rotstein in the Faculty of Aerospace Engineering.

The generous financial help of the Technion is gratefully acknowledged.

I would like to thank my supervisors Prof. Pini Gurfil, Prof. Ehud Rivlin and Dr. Hector Rotstein for their support and guidance throughout this research.

I would also like to thank my family for the infinite support and care, that has kept me going through the rough times.

To Peter

Contents

List of Figures

List of Tables

Abstract

Autonomous navigation of a platform from one point to another is a challenging task. One essential capability for successfully performing this task is calculation of an accurate navigation solution. Since the navigation solution, computed based on either inertial or odometry measurements, is accompanied by persistently increasing errors, it is common to utilize measurements from external sensors and additional available information for correcting the navigation errors, a process called *navigation aiding*. The Global Positioning System (GPS) is undoubtedly the most common approach for navigation aiding or, alternatively, for a direct calculation of a navigation solution. However, in certain cases, the GPS is unavailable or unreliable, and therefore alternative methods must be applied for navigation aiding.

One of the alternatives for navigation aiding is to use cameras, or more generally, imaging systems, giving rise to *vision-aided navigation* (VAN). The navigation aiding task becomes even more challenging when there is no prior information regarding the environment in which the platform is required to operate. In these cases, it may be also needed to construct a map of the observed environment. Performing these two tasks simultaneously is an approach known as Simultaneous Localization and Mapping (SLAM).

The current research focuses on VAN in unknown environments. It is assumed that the platform is equipped with a standard inertial navigation system and a single camera *only*. The camera-captured images are associated with navigation data and stored in a repository, which represents a mapping of the observed environment. The repository can be also used for constructing mosaic images. Consequently, the camera-captured images are used both for mapping and for navigation aiding. In contrast to SLAM, the mapping (i. e., mosaic construction and repository refinement), is performed in a background process, thereby considerably reducing the computational load.

The research described herein provides a few contributions to the vision-aided navigation literature, both theoretical and practical. In the first new algorithm developed in this research, the main idea is to couple online mosaic construction process to a camera scanning pattern, assuming that the camera is mounted on gimbals. It is shown that improved vision-based motion estimation is obtained in challenging operational scenarios such as when a narrow filed-of-view camera observes low-texture scenes. These motion estimations are fused with an inertial navigation system, allowing to reduce navigation errors in some of the navigation parameters, including position and velocity errors normal

to the motion heading. On the other hand, errors along the motion heading cannot be reduced using this algorithm, motivating the development of other methods.

Another new method developed in this research utilizes constraints, obtained by observing the same scene from three different views, for navigation aiding. A new formulation of such constraints is presented and proven, and a Kalman filter formulation is developed for fusing these three-view constraints with a standard inertial navigation system. Given three images with a common overlapping area, two of which were captured in the past and retrieved from a repository along with the attached navigation data, the new algorithm reduces the position errors in all axes to the level of errors present while the first two images were captured. Errors in other navigation parameters are also reduced. Trajectories that contain loops, in which the platform revisits a scene after some unknown time, are naturally handled by the new algorithm.

The second part of this research is concerned with cooperative navigation. A general multi-platform measurement model is considered. This measurement model involves navigation data and readings of onboard sensors from different platforms, possibly taken at different time instances. Since, in the general case, these various sources of information are correlated, the appropriate correlation terms must be calculated to obtain a consistent state estimation. The present research develops a new method for on-demand calculation of the required correlation terms based on the history of all the multi-platform measurements performed thus far. The newly-developed method relies on graph theory and is capable of rigorously handling the involved process and measurement noise for general multi-platform measurement models.

Finally, this research develops a new approach for vision-aided cooperative navigation. As opposed to the common approach, which is based on relative pose measurements between pairs of platforms, in the newly-proposed approach a measurement is formulated whenever the same scene is observed by three views, possibly captured by different platforms, not necessarily at the same time. The captured images, to which some navigation parameters are attached, are stored in repositories by each, or some, of the platforms in the group. As in case of a single platform, applying the three-view constraints for cooperative navigation reduces the position and velocity errors in all axes, as well as other navigation errors, without utilizing range measurements. As opposed to relative pose measurements, in the proposed approach the platforms' cameras are not required to be aimed at other platforms. Since the three-view measurement is a function of imagery and navigation information belonging to different platforms, these different sources of information can be correlated. The required correlation terms in the information fusion phase are explicitly calculated using the above-mentioned algorithm developed for a general multi-platform measurement model.

The algorithms developed in this research were examined in statistical simulations and demonstrated in experiments that involved real imagery and navigation data. The experiments unequivocally validated the developed methods and algorithms. A complete list of publications based on this doctoral research are provided in Appendix D.

Abbreviations

BTT	Bank To Turn
CCD	Charged Coupled Device
CDF	Cumulative Distribution Function
CN	Cooperative Navigation
DAG	Directed Acyclic Graph
DCM	Directional Cosine Matrix
FOV	Field of View
FV	Fictitious Velocity
GPS	Global Positioning System
IEKF	Implicit Extended Kalman Filter
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LLLN	Local Level Local North
LOS	Line of Sight
LS	Least Squares
MP	Multi Platform
NED	North East Down
SL	Straight and Level
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
TOM	Total Observability Matrix
VAN	Vision Aided Navigation

Nomenclature

$\hat{a}$	Estimation of a
$\tilde{a}$	Estimation error of a
a^t	True value of a
$\mathbf{b}$	Accelerometer bias vector
C_B^A	Transformation matrix from system A to system B
$\mathbf{d}$	Gyro drift vector
f	Focal length
$\mathbf{f}, \mathbf{h}$	known nonlinear functions
H	Measurement matrix
I	Image
K	Kalman filter gain matrix or camera calibration matrix
P	Covariance matrix
$P_{k+1\|k}$ or P_{k+1}^-	A priori covariance matrix at time instant t_{k+1}
$P_{k+1\|k+1}$ or P_{k+1}^+	A posteriori covariance matrix at time instant t_{k+1}
$\mathbf{Pos}$	Position vector
Q	Process noise covariance matrix
$\mathbf{q}$	Line of Sight
$\mathbf{q}_{i_j}^{C_i}$	Line of sight related to the jth feature in the ith view, expressed in camera system of ith view
R	Measurement noise covariance matrix
$\mathbf{T}_{ij}$	Translation vector from view i to view j
t	Time
t_i	Time instant t_i
$\mathbf{V}$	Velocity vector
$\mathbf{v}$	Measurement noise vector
$\mathbf{y}$	Measurements of external sensors
$\mathbf{y}_{IMU}$	Measurements of inertial measurement unit
$\mathbf{X}$	Navigation errors state vector
$\mathbf{x}$	Navigation solution
$\mathbf{z}$	Residual measurement
$\boldsymbol{\beta}$	Inertial sensors error model
λ	Scale parameter
$\boldsymbol{\Psi}$	Euler angles vector
$\Delta\mathbf{P}$	Position vector error
$\Delta\mathbf{V}$	Velocity vector error
Δt	Time step
$\Delta\boldsymbol{\Psi}$	Euler angles vector error
$\boldsymbol{\omega}$	Discrete process noise vector

$\boldsymbol{\omega}_c$	Continuous process noise vector
$\boldsymbol{\zeta}$	Navigation solution and inertial sensors error model
Φ	Discrete system matrix
Φ_c	Continuous system matrix

Specific nomenclature to Chapter 2:

H	Measurement matrix or homography matrix
$\mathbf{n}$	Normal to the scene plane vector
$\hat{R}_{C_1}^{C_2}$	Vision-based estimation of a rotational matrix from camera system at t_2 to camera system at t_1
$\hat{\mathbf{t}}_{1\to 2}$	Vision-based estimation of a translation vector from first view to second view
$\mathbf{x}$	Image coordinates
ψ_c	Camera pan angle
θ_c	Camera tilt angle
γ	Scale constant

Specific nomenclature to Chapters 3 and 5:

N	Overall cardinality of matching sets of features

Specific nomenclature to Chapter 4 (see in addition Section 4.3.2):

N	Number of cooperative platforms
$\mathbf{x}_i(t_j)$	Navigation solution of the ith platform at time instant t_j
$G = (V, E)$	Directed acyclic graph, composed of a set of nodes V and a set of arcs E
$w(a, b)$	Weight of an arc connecting the node a to the node b
$T_a = (V_{T_a}, E_{T_a})$	An inverse tree, constructed from the DAG G, containing all the possible paths in G to the node a. V_{T_a} and E_{T_a} are the set of nodes and arcs, respectively, comprising T_a
$\mathcal{M}_k$	Permutation set of kth level

Coordinate Systems:

B	Body coordinate system
C	Camera coordinate system
E	Earth-fixed coordinate system
L	Local-level, local-north coordinate system

Chapter 1

Introduction

Contents

Navigation is an essential capability without which mobile platforms will not be able to carry out even the simplest mission. At a first glance, navigation seems to be a simple issue: once a platform is equipped with dead reckoning sensors, and in particular with inertial navigation sensors, it is straightforward [1] to calculate the navigation solution, i. e. position, velocity and attitude. However, since the inertial navigation sensors provide imperfect measurements, the calculated navigation solution contains errors that develop over time. Depending on the quality of the inertial sensors, after a certain period of time, this navigation error will reach unacceptable levels.

Consequently, one has to use additional information and sensors in order to mitigate the developing inertial navigation errors, a process called *navigation aiding*. Alternatively, these sources of information and sensors can be used to directly calculate the navigation solution. Undoubtedly, since the global positioning system (GPS) was established in the

1970s, it has become the most common method for navigation and navigating aiding. For example, the majority of modern airborne navigation systems rely on the GPS signal. However, the GPS does not work properly in certain scenarios, such as when operating indoors, underwater, in urban environments and on other planets. Moreover, in certain situations the GPS signal is susceptible to jamming. In such cases, alternative techniques are required for navigation aiding. Moreover, it is often desired to have a backup capability for navigation aiding, in case the GPS signal becomes unavailable at some point during the mission.

With the rapid development of computational capabilities over the last few decades, a broad range of methods were proposed utilizing vision sensors for navigation aiding. The vision-aided navigation methods are considered appealing due to their relatively low cost and autonomy.

Another issue that has drawn much attention is the ability to operate in an unknown environment. In such a case, it is often required to map the environment observed by the platform during its motion. Being able to navigate using the incoming imagery, and in the same time to construct a map, is an approach known as *simultaneous localization and mapping*. The map of the observed environment can be represented by the real world locations of features extracted from the images, or by a mosaic image (or several mosaic images) that is constructed from the camera-captured images.

So far, navigation of a single platform was considered. However, many applications require a group of platforms to work in collaboration to perform a certain mission. Precise navigation is a key requirement for carrying out any autonomous mission by a group of cooperative platforms. Assuming the platforms are capable of intercommunication, cooperative navigation is a promising approach for improving navigation performance of the platforms in the group.

In the following section, related work on the different topics, briefly mentioned above, is discussed. In Section 1.2, an overview of the research presented in this dissertation is given. To make the reading of the rest of this manuscript easier, some preliminary material is provided in Section 1.3.

1.1 Related Work

1.1.1 Vision Aided Navigation

Navigation aiding deals with improving the performance of some basic inertial navigation system by fusing measurements from auxiliary sensors or additional, possibly exogenous, sources of information. In vision-aided navigation (VAN), this process is performed based on the imagery captured by an on-board camera. A typical VAN algorithm uses the information extracted from an image registration process, along with the information available from other sensors, for estimating the platform's states and possibly additional

navigation parameters.

VAN has been an active research field for the last few decades. For example, it was proposed to integrate the vision-based estimation of the velocity-to-height ratio with additional on-board sensors [2]; to apply the subspace-constraint approach [3] in order to partially estimate the states of an aircraft, based on measurements from an image registration process injected into an implicit extended Kalman filter [4]; and to utilize epipolar constraints formulated for each pair of matching features to aid the inertial navigation of a ground vehicle [5]. All the preceding methods rely only on information available from inertial navigation sensors and an on-board camera, without using a priori information or additional external sensors. This is also the approach adopted in this research.

Various methods for vision-aided navigation have been proposed assuming some additional external sensors and a priori information. One of the proposed methods used altimeter measurements for scaling the imaging sensors in order to improve state estimation during the landing phase of a space probe [6]. Others showed that absolute pose and motion estimation is possible when a digital terrain map (DTM) is available [7], [8]. Another approach is map-based navigation, which assumes that a map of the operational area is given and that the vehicle navigates by fusing inertial measurements, images of the environment and a map [9], [10], [11].

Images registration and image-based motion estimation are important constituents in all VAN methods. The existence of overlapping regions between processed images is the common assumption to all vision-based motion estimation techniques. A large overlapping region between two images is likely to yield a larger number of matched features and therefore should allow a more accurate motion estimation (and navigation) relying on two-view geometry methods. If a mutual overlapping region for more than two images can be found, the performance may be further enhanced by applying multi-view-geometry-based methods.

The two-view-geometry-based methods include relative motion calculation between two given views based on an estimated *essential matrix* [12], [13]. The motion parameters are then used for estimating the state vector, which is an augmented vector comprised of the vehicle's current pose and past poses for each captured image. When the observed scene is planar, the motion parameters can be calculated by estimating the homography matrix [14], [11], [15], [16]. Having in mind the requirements for real-time performance and a low computational load, in this research the estimated camera motion is related to a *constant*-size state vector comprised of the platform's current parameters only (in contrast to [12]).

However, given two overlapping images, it is only possible to determine camera rotation and up-to-scale translation [13]. Therefore, two-view based methods for navigation aiding are incapable of eliminating the developing navigation errors in all the states. With no additional information or sensors for resolving the scale ambiguity, such as range sensors or stereo vision, the vehicle states are only partially observable (e. g., position and velocity

errors along the flight heading are unobservable [12]).

The multi-view-geometry-based methods, in contrast to two-view-geometry, use connections among several images, assuming that a common overlapping region exists. Imagery information stemming from multiple images (≥ 3) with a common overlapping region enables to determine the camera motion up to a common scale [13]. Indeed, several multi-view methods for navigation aiding have been already proposed.

For example, some authors derive constraints relating features that are observed in several consecutive images, thereby claiming to achieve optimal exploitation of the available information in the observed scene [17]. These features, observed within multiple images, and the platform pose are related using an augmented state vector: The state vector contains the current platform pose and the platform pose for each previously-captured image that has at least one feature that appears in the current image. Once a certain feature, observed in the previous images, is no longer present in the currently-captured image, all the stored information for this feature is used for estimating the platform parameters, and the pose entries that belong to these past images are discarded. However, should the same feature be re-observed at some later time instant (e. g. whenever loops in the trajectory are performed), the method will be unable to use the data for the feature's first appearance. It was later proposed [18] to cope with loops using bundle adjustment [13]. This process involves processing *all* the images that are part of the loop sequence, and therefore real-time performance is hardly possible. In Ref. [19], the authors use the rank condition on the multiple-view-matrix [20] for simultaneously recovering 3D motion and structure during a landing process of an unmanned aerial vehicle, assuming a planar ground scene is observed.

Yu et al. [21] proposed using a trifocal tensor [13] for motion estimation of a single mobile camera. First, the relative motion between the two first images, denoted as base images, is estimated using epipolar geometry. Each next image is then related to the first two images via the trifocal tensor, which is then used for motion and pose estimation in a Kalman filter framework. Yet, whenever the currently-captured image does not share any common features with the first two images, the process is re-initialized by choosing two new base images. Hence, similar to [17], this method is incapable of handling loops.

Despite the advantages of multi-view methods, assuming that an overlapping region among several consecutive images exists may be invalid in many practical applications, such as various airborne applications. Violating this assumption usually degenerates the multi-view methods into two-view methods.

On the other hand, the platform may return to some area, already observed in the past, after some a priori unknown time. In this case, several overlapping images exist, yet these images were not captured simultaneously. Such scenarios, called as *loop scenarios*, are potentially useful for both navigation aiding and for refining the environment mapping. Different state-of-the-art approaches for handling loop scenarios are discussed in Section 1.1.3.

1.1.2 Simultaneous Localization and Mapping

The Simultaneous Localization and Mapping (SLAM) approach allows a mobile platform to construct a map of the observed environment, while at the same time localizing itself with respect to this map. It is generally assumed that the environment in which the platform operates is a priori unknown. SLAM methods can be found in a variaty of applications, including indoor [22], outdoor [23], aerial [24] and underwater [25] applications. A survey can be found in [26].

In SLAM, the estimation of the platform's navigation parameters and the construction of a representation of the observed environment are performed simultaneously. The general approach for solving the SLAM problem is to use an augmented state vector, composed of navigation states (e. g. position, velocity) and of parameters describing the observed environment, which are usually the feature coordinates in the real world. Thus, upon observing and identifying a new feature, its parameters are augmented into the state vector. Another variation [27] is to augment the state vector with parameters describing the camera-captured images locations (and perhaps other parameters) in a constructed mosaic image.

Several different assumptions regarding the available sensors can be found in the SLAM literature: Range and bearing measurements [28], [24], [29], bearing-only measurements [30], [22], and range-only measurements [31]. When processing a measurement, the augmented state vector yields an update both in the navigation states and in the environment model. Consequently, correlation between the platform's states and the environment parameters is consistently maintained. On the other hand, the computational requirements are constantly increasing as the state vector grows in size.

Different approaches were proposed for handling this computational bottleneck. These include neglecting low-correlation bonds in the augmented state vector [32], maintaining only currently-visible features in the state vector [33], and using several submaps, representing the overall observed environment [34].

In contrast to SLAM, in this research it is proposed to separate the process of constructing a representation of the environment (e. g. mapping) from the process of motion estimation and navigation aiding. Thus, navigation aiding can be performed based on the current representation of the environment, while this representation is refined in a background process. Although the obtained navigation performance could be compromised compared to SLAM, such an approach allows navigation aiding using significantly lower computational resources, in particular when handling loop scenarios.

1.1.3 Methods for Handling Loop Scenarios

Special research attention has been devoted to developing navigation-aiding methods for handling loop scenarios, or, in other words, determine how the navigation solution can be updated when the platform revisits some area. Several approaches were proposed over

the years.

Applying smoothing over the images that were captured in the loop chain is one common approach [35], [36]. By refining the transformations that relate these images with a common reference, it is possible to considerably reduce the growing image registration errors, thereby producing a consistent mosaic. Assuming an available range to-the-scene information allows positioning the platform and hence reduce some of its navigation errors.

Bundle adjustment [13], already mentioned in Section 1.1.1, is another approach for handling loop scenarios. In this approach, an optimization is performed, seeking to minimize a cost function that includes the actual and predicted feature measurements from all the images captured in the loop chain [18].

Loop scenarios are also naturally handled in SLAM methods, as discussed in Section 1.1.2. The augmented state vector contains the coordinates of the crossover features, i. e. features that belong to the revisited scene. Re-observing the scene allows to refine both the coordinates of these features and the platform position, thereby reducing navigation errors according to the estimation precision of these features.

Although the above approaches for handling loop scenarios are capable of refining the map and of localizing the platform upon identifying a loop event, real-time performance is hardly possible due to the involved computational requirements: The smoothing technique processes all the images in the loop chain; bundle adjustment optimizes all the features observed in any of the images in the loop chain; while SLAM maintains an augmented state composed of all the observed features thus far. In addition, SLAM and bundle adjustment methods share the same common property of estimating the feature locations in the real world, i. e., structure reconstruction. As opposed to this, in this research it is proposed to handle loop scenarios (in the context of navigation aiding) by processing only three images and without structure reconstruction, thereby substantially reducing the computational requirements and allowing for real-time implementation.

1.1.4 Mosaicking and Mosaic-Based Navigation

Mosaicking is the process of fusing partially overlapping images into a single image. Traditional methods rely on the homography model [13], which is valid when the platform performs pure rotation while observing a general scene, or when performing translation and rotation while observing a planar scene. A tutorial and further details regarding different methods for mosaic construction can be found in [37].

Alternatively, the mosaic can be represented by a set of original camera-captured images, assigned with appropriate transformations. Warping each image according to the assigned transformation and subsequently fusing all the warped images produces the mosaic image. See, for example, [38], [15].

Navigation based on off-line mosaics has been extensively studied (e. g. [11], [39]). The more challenging problem, however, is navigation aiding based on online mosaic construction. This problem is strongly related to SLAM (cf. [27], [40]): The objective is to

estimate the platform's location (and perhaps other navigation parameters) when operating in an unknown environment, while the mapping step is comprised of constructing a consistent map based on the camera-captured images (instead of extracting features from these images and trying to estimate their locations in the real world).

Several works have considered navigation based on an online construction of the mosaic image, including [38], [15], [36], [25], [41]. The common approach, in the context of motion estimation and navigation aiding, is to assume that the range to the observed scene is available. This assumption allows scale determination, which cannot be determined based on pure imagery data [13], thereby providing complete motion estimation. Consequently, it is possible to obtain a position solution in all axes. Loop scenarios are usually treated by smoothing the thus-far constructed mosaic image [25], [41], which also yields improved navigation solution.

1.1.5 Cooperative Navigation

The ability of a group of cooperative platforms to autonomously carry out various tasks strongly depends on the navigation capabilities of each individual in the group. These tasks include cooperative mapping and localization [42], [43], [44], formation flying [45], cooperative tracking [46], autonomous multi-vehicle transport [47], and other applications. While various methods exist for navigation-aiding of a single platform, collaboration among several, possibly heterogeneous, platforms, each equipped with its own set of sensors, is expected to improve performance even further [48].

Different methods have been developed for effectively localizing a group of platforms with respect to some reference coordinate system or with respect to the platforms themselves. Most of the proposed methods for cooperative navigation (CN) (including [49], [48], [50], [47], [51], [52], [53]) assume that each platform is capable of measuring the relative range and bearing to other platforms that are located nearby.

One of the pioneering works on cooperative localization proposed to restrain the development of navigation errors by using some of the platforms as static landmarks for updating the other platforms in the group [49]. While the method was further improved by others, all of the derived methods share the same drawback of having to stop the motion of some of the platforms for updating the others, which is, for example, impossible for fixed-wing aircrafts.

Another important work is by Roumeliotis and Bekey [48], where a centralized approach for sensors fusion was applied based on the available relative pose measurements between the platforms in the group. This architecture was then de-centralized and distributed among the platforms. Later, an extension was proposed to handle more general relative observation models [54]. The setup comprising of a group of platforms capable of measuring relative poses to adjacent platforms have been studied in other works, including [47], [51], [55], [56] and [57].

A different body of works [58], [59], [60] suggests to maintain in each platform esti-

mated parameters for all the platforms in the group. For example, in [58] and [59] each platform estimates the pose of every other platform relative to itself, while in [60] each platform estimates the navigation state (position, velocity and attitude) of all the other platforms by exchanging inertial measurement unit (IMU) information and relative pose measurements.

Another approach for CN is to identify a common scene observed by different platforms, and to express the resulting constraints as a measurement to the navigation filter. Such an approach was recently suggested in [61], [44] considering measurements that combine pairs of platforms. Merino et al. [61] suggested using a homography connection for motion estimation between two aerial platforms observing the same scene, assuming the range to the scene is available. Kim et al. [44] considered general measurements generated by pairs of platforms. These may be either relative pose measurements or two-view measurements. Yet, in the absence of a range sensor, measurements between pairs of platforms do not allow three-dimensional localization.

Regardless of the approach applied for CN, the navigation information involved in the measurement is obtained from different platforms, possibly belonging to different time instances. In the general case, these sources of information can be statistically dependent. For instance, the navigation information of any two platforms becomes correlated after the first update is carried out. Ignoring this correlation can result in inconsistent and over-confident estimations [62].

Several approaches have been proposed for coping with the correlation terms in multi-platform (MP) systems, assuming relative pose measurements. In [48], an augmented covariance matrix, comprised of covariance and cross-covariance matrices relating all the platforms in the group, was maintained in a distributed manner. In [50], this approach was applied to cooperative mapping and localization. In this case, the augmented covariance matrix also contains parameters that represent the landmarks observed by each platform in the group. Howard et al. [58] suggested a method that avoids correlated updates in certain situations. Similarly, in [62], the cross-covariance terms were not explicitly estimated. Instead, the authors proposed to maintain a bank of filters, tracking the origins of measurements and preventing multiple use of measurements. References [63] and [45] studied the filter inconsistency when correlated measurement sequences are used.

In [44], a method for consistent information fusion was proposed, considering relative pose measurements and two-view measurements (that involve two images of the same scene, taken by two platforms). In the general case, the two images may be captured at different time instances. The authors formulated an optimization problem that involves the history of the performed measurements between pairs of platforms and measurements of the proprioceptive sensors of each of the platforms in the group. This problem is solved each time a new measurement of any kind is received, yielding an updated pose history of all the cooperative platforms [44].

1.2 Research Overview

This dissertation addresses the problem of VAN in an unknown environment. The platform is assumed to be equipped with a standard inertial navigation system and a single onboard camera, which can be mounted on gimbals. No additional external sensors are assumed to exist, and no additional information is necessarily available, except for an initial navigation solution and camera calibration parameters. In particular, no range sensor is assumed, as opposed to most of the VAN and SLAM works (cf. Section 1.1). In addition, it is assumed that the camera-captured images are associated with an appropriate navigation solution, stored and maintained in a repository. These images are also used for online mosaicking.

Despite of the resemblance of the setup described above to SLAM, the approach developed in this research is different. In the proposed approach, the mapping phase, i. e. imagery repository maintenance and online construction of a mosaic image, is performed in a background process, while navigation aiding is performed in the main process using imagery and associated navigation data taken from the repository. Thus, the imagery repository is maintained outside the filter, using parameters obtained from the navigation system, while the filter state vector does not contain representation of the observed environment. In this research a constant-size state vector, representing only the current navigation solution is maintained (cf. Section 1.3.2). Such an approach allows to considerably reduce the computational requirements compared to SLAM methods, although the performance can be somewhat compromised when several images have a common overlapping area. In particular, the above-mentioned approach of refining the mapping in a background process allows to efficiently perform navigation aiding in loop scenarios (see discussion below), in which a platform revisits the same region after some a priori unknown time.

Chapter 2 presents a method for improving navigation performance while operating in scenarios that are considered to be challenging in the context of vision-based motion estimation, such as when a narrow field of view (FOV) camera observes low-texture scenes. Using a gimbaled camera, it is proposed to couple the camera scanning and online mosaicking processes. This coupling yields increased overlapping regions, which allows to perform motion estimation with improved accuracy. The estimated motion is then fused with a navigation system using an implicit extended Kalman filter. An observability analysis of the proposed method is presented, modeling the linearized system as a piece-wise constant system.

Similarly to all methods for VAN that are based on two-view techniques (cf. Section 1.1.1), the method presented in Chapter 2 is capable of estimating the translational motion up to scale, mitigating only some of the navigation errors. For example, position and velocity errors along the motion heading cannot be reduced. In particular, when loop scenarios are considered, the method is incapable of fully exploiting the available information for navigation aiding, which motivated the development of the method described

in the next chapter.

In Chapter 3, a new VAN method is developed. The method formulates new constraints that are obtained from observing a general static scene by three different views. The newly-developed constraints combine imagery and navigation data at the time instances in which the three images were taken. The developed constraints are fused with an inertial navigation system using an implicit extended Kalman filter, allowing to reduce navigation errors, in particular position and velocity errors in *all* axes, each time a set of three images with a common overlapping area is available. The method requires processing only three images for the navigation aiding phase, while the environment representation refinement (e. g. imagery repository, mosaic) can be performed in a background process by applying various algorithms (e. g. smoothing, bundle adjustment). Loops in the trajectory are naturally handled, allowing navigation aiding by processing only three images and thereby reducing the computational load compared to the state-of-the-art techniques (cf. Section 1.1.3). In contrast to SLAM and bundle adjustment, the suggested approach eliminates the need for an intermediate phase of structure reconstruction. To the best of the Author's knowledge, the concept of utilizing three-view geometry constraints for navigation aiding, including the well-known trifocal tensor, has not been proposed thus far for navigation aiding, let alone for handling loop scenarios.

In the next chapters, the dissertation focuses on cooperative navigation. A group of collaborative inter-communicating platforms is assumed, wherein each platform is equipped with its own dead reckoning or inertial navigation system, onboard camera, and perhaps additional external sensors and/or information.

A general MP measurement model that can be used for CN is considered (Chapter 4). This model relates between the navigation information from any number of platforms and the actual readings of the onboard sensors of these platforms, which are not necessarily taken at the same time. For example, the considered MP model can represent relative pose measurements between pairs of platforms or two-view measurements (cf. Section 1.1.5). In the general case, all the involved sources of information can be correlated.

In addition to the a priori unknown identities of the platforms that participate in an MP measurement, the assumed general MP model contributes a manifold of a priori unknown parameters - the time instances that participate in the measurement. These additional unknown parameters render any approach that is based on maintaining the correlation terms impractical (cf. Section 1.1.5). Another possible approach to tackle this problem is to avoid explicit calculation of the correlation terms by applying the covariance intersection (CI) method [64], or its generalization [65]. CI allows consistent fusion of different, possibly correlative, sources of information, while the actual correlation is unknown. However, as reported in [66], [62], CI is incapable of handling partial updates, i. e., cases in which the measurement matrix contains only a partial representation of the state vector. Thus, although CI was applied in specific applications [67], [68], the CI method cannot be applied for the considered general MP measurement model.

In Chapter 4, it is proposed to explicitly calculate the required correlation terms

based on the MP measurements performed thus far, which therefore need to be stored. The method is capable of handling general scenarios, possibly involving different MP measurement models and regardless of the MP measurements that were performed thus far. The developed method utilizes a graph representation of the history of all the executed MP measurement updates for calculating the correlation terms. It is assumed that this graph is maintained by every platform in the group. Its construction and the involved information that need to be transmitted among the platforms in the group is discussed in the next chapter.

Chapter 5 builds up on Chapters 3 and 4: The three-view geometry measurement, originally proposed in Chapter 3 for navigation aiding of a single platform, is extended for cooperative navigation, while using the approach developed in Chapter 4 to obtain consistent estimation and data fusion. As opposed to CN methods that rely on relative pose measurements (cf. Section 1.1.5), in the newly-developed approach the platform's camera is *not* required to be aimed towards other platforms. Instead, a measurement is formulated whenever the same scene is observed by three views taken by at least two different platforms, i. e. either each view is captured by a different platform, or two of the three views are captured by the same platform.

Another key aspect of the proposed method, is that the three images of the same region are *not* necessarily captured at the same time. All, or some, of the platforms maintain a local repository of captured images that are associated with some navigation parameters. These repositories are accessed on demand to check if a region, currently observed by one of the platforms, denoted as the querying platform, has been observed in the past by other platforms in the group. Images containing the same region are transmitted, with the attached navigation data, to the querying platform. The information received from other platforms, in addition to the navigation and imagery data of the querying platform, can be used for updating the navigation system of the querying platform. As in the case of a single platform, the three-view geometry constrains mitigate the secular growth of navigation errors of the updated platform, including position and velocity errors in all axes, without any additional a priori information or any other sensors.

1.3 Preliminaries

1.3.1 Coordinate Systems Definition

Throughout this thesis, the following coordinate systems are used:

- E - Earth-centered, Earth-fixed (ECEF) coordinate system. Its origin is set at the center of the Earth, the Z_E axis coincides with the axis of Earth rotation, X_E goes through the point latitude 0^o, longitude 0^o, and Y_E completes a Cartesian right-hand system.

- L - Local-level, local-north (LLLN) reference frame, also known as a north-east-down (NED) coordinate system. Its origin is set at the navigation system's location. X_L points north, Y_L points east and Z_L completes a Cartesian right hand system.

- B - Body-fixed reference frame. Its origin is set at the vehicle's center-of-mass. X_B points towards the vehicle's front, Y_B points right when viewed from above, and Z_B completes the setup to yield a Cartesian right hand system.

- C - Camera-fixed reference frame. Its origin is set at the camera center-of-projection. X_C points toward the FOV center, Y_C points toward the right half of the FOV and Z_C completes the setup to yield a Cartesian right-hand system.

  The camera system defined above is used in Chapter 2, while in Chapter 3 the camera system is redefined.

1.3.2 Inertial Navigation Errors Model

Assuming the platform is equipped with an inertial measurement unit, it is capable of calculating its navigation solution $\mathbf{x}$, defined as

$$\mathbf{x} \doteq \begin{bmatrix} \mathbf{Pos}^T & \mathbf{V}^T & \mathbf{\Psi}^T \end{bmatrix}^T \tag{1.1}$$

where $\mathbf{Pos}, \mathbf{V}$ and $\mathbf{\Psi}$ are the position, velocity and angular orientation, respectively. The position is usually computed in terms of latitude, longitude and height, while the velocity is expressed in NED system. Denote by $\mathbf{x}^t$ the (unknown) true navigation solution and let $\mathbf{y_{IMU}}$ represent the IMU measurements. The errors in $\mathbf{y_{IMU}}$ are modeled by an unknown vector of parameters $\boldsymbol{\beta}^t$. Denote by $\boldsymbol{\beta}$ the calculated model of inertial sensor errors, used for correcting the measurements $\mathbf{y_{IMU}}$ (cf. Section 1.3.3). In particular, a simple model of $\boldsymbol{\beta}$ can be defined as

$$\boldsymbol{\beta} \doteq \begin{bmatrix} \mathbf{d}_{IMU}^T & \mathbf{b}_{IMU}^T \end{bmatrix}^T \tag{1.2}$$

where $\mathbf{d}_{IMU} \in \mathbb{R}^3$ is the gyro drift, and $\mathbf{b}_{IMU} \in \mathbb{R}^3$ is the accelerometer bias. This model is used throughout this research, except for Chapter 4, in which a general model of $\boldsymbol{\beta}$ is assumed.

Letting

$$\boldsymbol{\zeta}(t_k) \doteq \begin{bmatrix} \mathbf{x}^T(t_k) & \boldsymbol{\beta}^T(t_k) \end{bmatrix}^T \tag{1.3}$$

the navigation solution is given by

$$\boldsymbol{\zeta}(t_{k+1}) = \mathbf{f}(\boldsymbol{\zeta}(t_k), \mathbf{y}_{IMU}(t_k)) \tag{1.4}$$

As common in navigation-aiding techniques, the nonlinear navigation equations, implemented in the strapdown mechanism [1], are casted into a linearized system, which is

expressed in terms of navigation errors rather than the navigation parameters themselves. The following navigation error state vector is defined

$$\mathbf{X}(t) \doteq \begin{bmatrix} \mathbf{x}(t) - \mathbf{x}^t(t) \\ \boldsymbol{\beta}(t) - \boldsymbol{\beta}^t(t) \end{bmatrix} \equiv \boldsymbol{\zeta}_i(t) - \boldsymbol{\zeta}_i^t(t) \tag{1.5}$$

The evolution of the state vector $\mathbf{X}$ can be modeled by the linear time-varying stochastic model [69], [1]:

$$\dot{\mathbf{X}}(t) = \Phi_c(t)\mathbf{X}(t) + \boldsymbol{\omega}_c(t) \tag{1.6}$$

where Φ_c is the continuous system matrix and $\boldsymbol{\omega}_c$ is the process noise, which is assumed to be white and zero-mean Gaussian. This continuous time model can be replaced by a discrete model

$$\mathbf{X}(t_b) = \Phi(t_a, t_b)\mathbf{X}(t_a) + \boldsymbol{\omega}(t_a, t_b) \tag{1.7}$$

where $\Phi(t_a, t_b)$ is the discrete system matrix relating the state between any two time instances t_a and t_b, $t_b > t_a$, and $\boldsymbol{\omega}(t_a, t_b)$ is the equivalent discrete process noise. An alternative representation for $\Phi(t_a, t_b)$ and $\boldsymbol{\omega}(t_a, t_b)$, which is also used throughout this thesis, is $\Phi_{t_a \to t_b}$ and $\boldsymbol{\omega}_{t_a \to t_b}$, respectively.

Letting $\Delta t \doteq t_b - t_a$, the discrete system matrix Φ is calculated according to

$$\Phi = e^{\Phi_c \Delta t} \tag{1.8}$$

while the discrete process noise $\boldsymbol{\omega}$ is given by

$$\boldsymbol{\omega}(t_a, t_b) = \int_{t_a}^{t_b} \Phi(t_b, \tau)\boldsymbol{\omega}_c(\tau)d\tau \tag{1.9}$$

This discretization process correctly represents the development of $\mathbf{X}$ for small time intervals Δt, or when considering scenarios for which the system (1.6) is time-invariant.

Assuming the specific model of IMU errors, given in Eq. (1.2), the state vector used throughout this research, except for Chapter 4, is

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\boldsymbol{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T \tag{1.10}$$

where $\Delta\mathbf{P} \in \mathbb{R}^3, \Delta\mathbf{V} \in \mathbb{R}^3, \Delta\boldsymbol{\Psi} = (\Delta\phi, \Delta\theta, \Delta\psi)^T \in [0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ are the position, velocity and attitude errors, respectively, and $\mathbf{d}$ and $\mathbf{b}$ are the residual gyro drift and accelerometer bias, respectively:

$$\mathbf{d} \doteq \mathbf{d}_{IMU} - \mathbf{d}_{IMU}^t \quad , \quad \mathbf{b} \doteq \mathbf{b}_{IMU} - \mathbf{b}_{IMU}^t \tag{1.11}$$

with $\mathbf{d}_{IMU}^t, \mathbf{b}_{IMU}^t$ being the unknown true values of $\mathbf{d}_{IMU}, \mathbf{b}_{IMU}$. The position and velocity errors are expressed in the NED system, while $\mathbf{d}$ and $\mathbf{b}$ are given in the body-fixed reference frame.

The continuous system matrix, valid for short periods of operation, significantly smaller than the Schuler period (around 84 minutes), is given by [1]:

$$\Phi_c = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & A_s & 0_{3\times3} & C_L^B \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -C_L^B & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{1.12}$$

where the matrix C_L^B is a directional cosine matrix (DCM) transforming from body system to LLLN system and A_s is a skew-symmetric matrix of the specific force vector $\mathbf{f} = \begin{pmatrix} f_x & f_y & f_z \end{pmatrix}^T$, measured by the accelerometers and expressed in the NED system:

$$A_s = \begin{bmatrix} 0 & -f_D & f_E \\ f_D & 0 & -f_N \\ -f_E & f_N & 0 \end{bmatrix} \qquad \begin{bmatrix} f_N \\ f_E \\ f_D \end{bmatrix} = C_L^B \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{1.13}$$

While the scenario examples considered in this research indeed satisfy this condition, one could adopt less degenerated process models (for medium-term and long-term scenarios [1]) as well. It is worth noting that a similar model of the system matrix is widely used also in the SLAM community (e. g. [70]).

Based on Eqs. (1.7), (1.8) and (1.12), it is possible to obtain the following approximations to the attitude, velocity and position errors:

$$\Delta\boldsymbol{\Psi}(t_b) = -C_L^B \mathbf{d}\Delta t + \Delta\boldsymbol{\Psi}(t_a) \tag{1.14}$$

$$\Delta\mathbf{V}(t_b) = -\frac{1}{2}A_s C_L^B \mathbf{d}(\Delta t)^2 + \left[A_s\Delta\boldsymbol{\Psi}(t_a) + C_L^B\mathbf{b}\right]\Delta t + \Delta\mathbf{V}(t_a) \tag{1.15}$$

$$\begin{aligned}\Delta\mathbf{P}(t_b) = &-\frac{1}{6}A_s C_L^B \mathbf{d}(\Delta t)^3 + \frac{1}{2}\left[A_s\Delta\boldsymbol{\Psi}(t_a) + C_L^B\mathbf{b}\right](\Delta t)^2 + \\ &+\Delta\mathbf{V}(t_a)\Delta t + \Delta\mathbf{P}(t_a)\end{aligned} \tag{1.16}$$

where $C_L^B, A_s, \mathbf{d}$ and $\mathbf{b}$ are evaluated at t_a.

1.3.3 Navigation Aiding Concept

The concept of navigation aiding is illustrated in Figure 1.1. The inertial navigation system typically consists of inertial sensors, whose measurements, $\mathbf{y}_{IMU}$, are processed by the strapdown into a navigation solution [1]. Since the measurements of these sensors are imperfect, the calculated navigation solution contains errors which are developing over time.

In navigation aiding, the navigation errors are estimated based on measurements from external sensors[1] and other sources of information (such as DTM). In addition to the

[1]External sensors refer to the onboard sensors apart from the IMU, such as: GPS receiver, altimeter, camera.

navigation errors $\Delta\mathbf{P}, \Delta\mathbf{V}, \Delta\boldsymbol{\Psi}$, it is also common to estimate a parameterization of IMU errors $\boldsymbol{\beta}$ (cf. Section 1.3.2). The estimated navigation errors are used for correcting the navigation solution calculated by the inertial navigation system (INS), while the IMU readings are corrected according to the estimated IMU errors parameterization. The latter is performed at the sampling frequency of the IMU, which is typically much higher than the filter frequency.

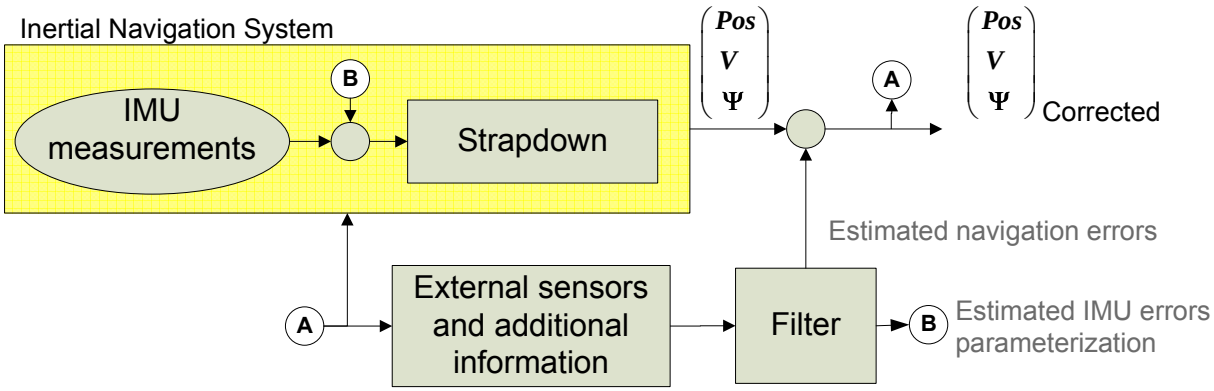


Figure 1.1: Illustration of the navigation aiding concept.

Chapter 2

Navigation Aiding Based on Coupled Mosaicking and Camera Scanning

Contents

In this chapter it is proposed to utilize an online mosaicking process for vision-aided navigation, using an on-board gimbaled camera that scans regions in the vicinity of the platform's trajectory. Although the developed method can be applied for general cameras observing general scenes, the focus in this chapter is on cameras with a narrow field of

view, observing low-texture scenes. Such a constellation is considered challenging, in the context of motion estimation and navigation aiding, since the typical imagery contains very limited useful information, e. g. a small number of high-quality features.

The mosaicking and camera scanning processes are coupled, yielding increased overlapping regions between the incoming imagery and the mosaic image. This results in improved motion estimation when operating in the challenging scenarios mentioned above. The mosaic-based estimated motion is fused with an inertial navigation system, thereby reducing navigation errors in some of the parameters of the state vector.

As discussed in Section 1.2, the environment mapping, represented by a mosaic, is separated from navigation aiding (in contrast to SLAM), resulting in reduced computational requirements. Consequently, two types of mosaics are constructed: A small, temporary mosaic is computed in real-time based on recently captured images, and a larger, main mosaic is computed in a background process including all the images. The motion estimation and navigation aiding are performed on the temporary mosaic, while the main mosaic may be used for global navigation.

The correlation terms between the navigation system and the mosaic construction process are not maintained in the proposed approach. The advantage of this architecture is the low computational load required for navigation aiding. However, the accuracy of the proposed method could be compromised compared to bearing-only SLAM.

2.1 Method Overview

Figure 2.1 shows the main components of the architecture under consideration. The specific system assumed in this chapter is an airborne platform equipped with a gimballed camera and an inertial navigation system. Throughout this chapter, a narrow-FOV camera is assumed, since it is more realistic than a wide-FOV camera for many cases of practical interest. As mentioned above, a narrow-field makes the VAN and the image-based motion estimation problems more challenging. However, the proposed method is not restricted to cameras with narrow FOV, and is valid for other cameras as well. In addition, it is assumed that the observed ground area is sufficiently close to being planar, or alternatively, that the flight altitude above ground level is high relative to ground changes in elevation[1].

The INS consists of an inertial measurement unit and a strapdown algorithm. The strapdown algorithm integrates the accelerations and angular rates (or rather, the velocity and angular increments) from the IMU to produce a navigation solution, which is comprised of platform position, velocity and attitude. Due to the unavoidable errors

[1]This assumption is made due to the construction process of the mosaic image, which is based on the homography transformation. However, the proposed approach for fusing image-based motion estimations and navigation data may be also applied without constructing a mosaic image, in which case non-planar scenes can be handled as well (cf. Section 2.5.2.2).

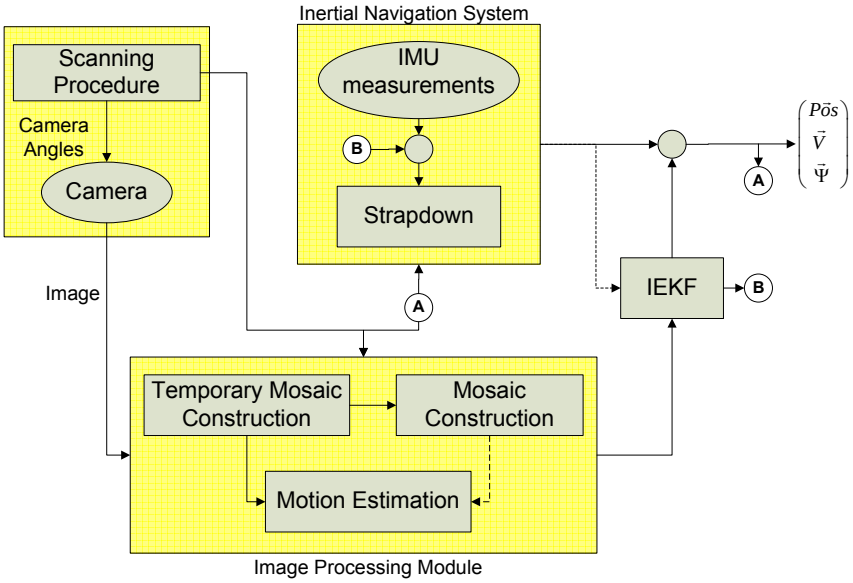


Figure 2.1: Overview of the system concept.

of the IMU sensors, the computed navigation parameters develop errors which increase unboundedly over time. It is well-known (cf. Section 1.3.2) that for relatively low-grade inertial sensors, errors grow proportionally to time cubed, and hence an uncompensated inertial solution becomes useless in a relatively short period of time.

During the flight, an on-board camera captures images of ground regions according to a scanning procedure. The acquired images are directed to the image processing module that is accountable for mosaic image construction and for relative motion estimation. While all the images are used for updating the mosaic image, the motion estimation is performed at a lower frequency, utilizing only some of the images.

The mosaic construction is coupled with the camera scanning procedure, and is processed in two phases: 1) The camera-captured images are used for constructing a small temporary mosaic image. This temporary mosaic image is used for motion estimation at appropriate time instances. 2) After each motion estimation event, the temporary mosaic image is emptied and initialized to the most recent camera-captured image, while the removed images from the temporary mosaic image are used to update the main mosaic image in a background process.

The image-based motion estimation is reformulated into measurements, which are then injected into an implicit extended Kalman filter (IEKF) in order to update the navigation system and thereby arrest the development of inertial navigation errors.

In this chapter, the camera is assumed to be mounted on gimbals, capable of performing pan and tilt movements with respect to the platform. The camera pan and tilt angles between B and C coordinate systems are denoted by ψ_C and θ_C, respectively.

2.2 Camera Scanning Procedure and Mosaic Construction Method

This section presents a detailed description of the camera scanning and mosaic construction procedures. Each procedure by itself is simple and several variations have appeared in the literature before. The section hence focuses on the coupling between scanning and mosaic image construction, in particular on the aspects that allow improving the accuracy of the motion estimation in challenging scenarios such as narrow-FOV cameras and low-texture scenes. In addition, it will be shown that relatively large ground areas may be represented in the mosaic image without restricting the trajectory of the platform. More sophisticated camera scanning methods may be considered, for instance those exploiting the coupling with the platform trajectory or maximizing the ground coverage, but are left for future studies.

2.2.1 Scanning Procedure

During flight, the onboard camera captures images of the ground according to commands either from a human operator, an autonomous tracking algorithm of some features on the ground, or a scanning procedure. Figure 2.2(a) shows a schematic illustration of the implemented scan procedure. When captured, each new frame is processed and used to update the mosaic image of the flight area. A detailed discussion of the on-line mosaic construction appears in Section 2.2.2.

As can be seen, each image partially overlaps the preceding image as well as images from the previous scan stripe. The existence of overlapping regions is essential for performing image matching between captured images. In addition, and as opposed to most motion-from-structure methods, the additional overlapping region, provided by the camera scanning procedure, enables enhancement of motion estimation, as will be seen in Section 2.3. The proposed scan pattern also allows implementation of improved mosaic construction methods.

We assume that the scanning procedure modifies the pan angle of the camera, ψ_c, while keeping the camera tilt angle constant, as shown in Figure 2.2(b). Given camera angles at the current time instant, the calculation of camera angles for the next time instant is performed in two steps. First, the line-of-sight (LOS) vector for the next camera aiming point in the body-fixed reference frame, $\hat{\mathbf{r}}^B$, is determined according to

$$\hat{\mathbf{r}}^B = C_B^C(\psi_c)\frac{[f,\ d\cdot CCD_{\mathbf{Y}_C}/2,\ 0]^T}{\left\|[f,\ d\cdot CCD_{\mathbf{Y}_C}/2,\ 0]^T\right\|} \tag{2.1}$$

where $C_B^C(\psi_c)$ is the DCM transforming from the camera reference frame to the body frame, computed based on current camera angles; f is the camera focal length; d is the scan direction, so that $d = 1$ for increasing the camera pan angle and $d = -1$ for

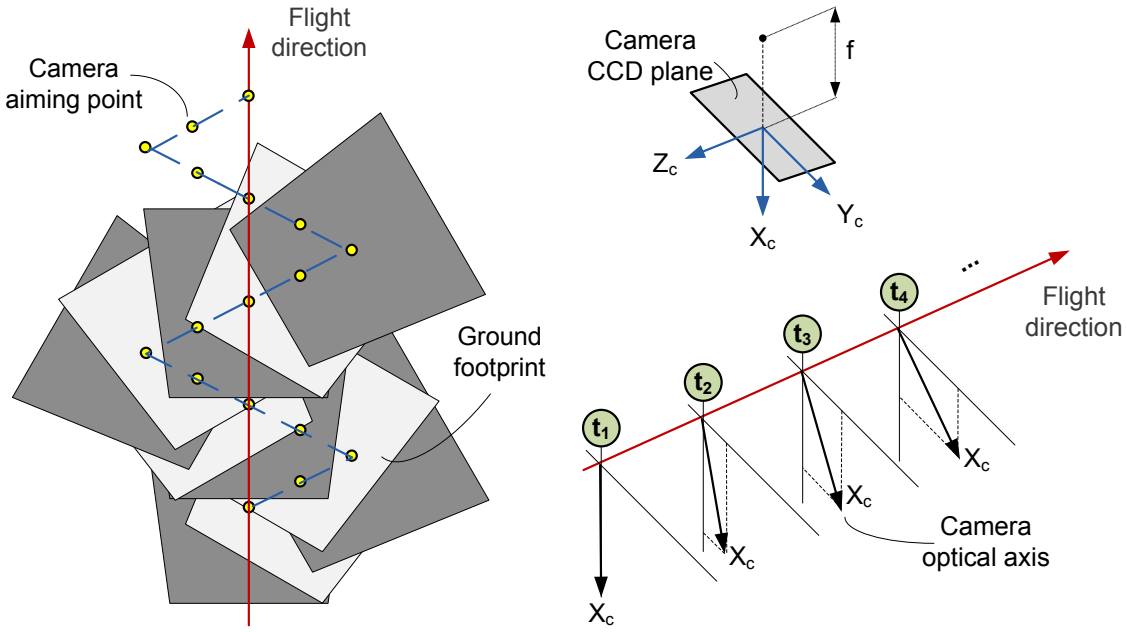


Figure 2.2: (a) A schematic illustration of the scanning procedure. (b) Definition of the camera coordinate system and a schematic illustration of camera angles during the scan procedure.

decreasing the camera pan angle; and $CCD_{\mathbf{Y}_C}$ is the size of the camera charged coupled device (CCD) in pixels along the Y_C axis.

The next step is to compute the new camera angles from $\hat{\mathbf{r}}^B$. The DCM transforming from B to C can be written as

$$C_C^B(\psi_c) = \begin{bmatrix} 0 & \sin\psi_c & \cos\psi_c \\ 0 & \cos\psi_c & -\sin\psi_c \\ -1 & 0 & 0 \end{bmatrix} \tag{2.2}$$

Since the aiming point vector in C is, by definition, $[1 \quad 0 \quad 0]^T$, one can write

$$\hat{\mathbf{r}}^B = C_B^C(\psi_c)\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T = \begin{bmatrix}0 & \sin\psi_c & \cos\psi_c\end{bmatrix}^T \tag{2.3}$$

hence

$$\psi_c = \tan^{-1}\left[\frac{\hat{\mathbf{r}}^B(2)}{\hat{\mathbf{r}}^B(3)}\right] \tag{2.4}$$

The scanning direction, d, is switched once the camera pan angle, ψ_c, reaches a certain pre-specified level; this level is constrained by the corresponding gimbal limit but may be smaller than this mechanical limit.

For simplicity, it is assumed implicitly that the velocity-over-height ratio and the camera sampling frequency provide sufficient overlapping regions between each two adjacent

images along the flight direction. Thus, the proposed scan methodology moves the camera only in a direction perpendicular to the flight trajectory. Notice that in a practical application this imposes a complex trade-off among flying altitude over ground, platform speed, FOV, scanning slant angle and resolution. However, the method presented here may be adjusted to work with a relaxed version of the assumption. Note also that no additional or a priori information is required.

2.2.2 Mosaic Construction Method

We shall now present a detailed description of the mosaic construction method using the camera scanning method described in Section 2.2.1. It is worth stating that, besides being used for navigation aiding as described in this chapter, a mosaic image of the overflown ground region constitutes an important aid to surveillance and mission operation.

During the scan, images are captured using varying camera angles. While all the images contribute to the construction of a mosaic, in the current implementation only images taken while the camera was pointing downwards are used for motion estimation. These images are referred to as *downward-looking images*.

Figure 2.3 provides a block diagram of the implemented mosaic construction process. Two mosaic representations are constructed in the proposed approach: a temporary mosaic image that is used for motion estimation, and the main mosaic image which is the final mosaic image constructed based on the captured images.

The temporary mosaic image is initialized to a downward-looking image, once such an image is captured, and is updated with new non-downward-looking images. When a new downward-looking image is captured, it is matched to a relevant region in the temporary mosaic image, which is calculated utilizing information from the navigation system. Next, motion estimation is performed, as will be discussed in Section 2.3.

The temporary mosaic image is expressed in the preceding downward-looking image system, defined as the coordinate system C of the previous downward-looking image. Therefore, the estimated motion describes the relative motion performed by the camera between two adjacent downward-looking images. This estimation will be used to correct developing inertial navigation errors (cf. Section 2.4).

Due to the coupling between the scanning procedure and the mosaic construction process, an enlarged overlapping area between the new downward-looking image and the temporary mosaic image is achieved. This, and the quality of the constructed temporary mosaic image, are the two factors that allow better motion estimation in certain scenarios, as will be demonstrated in Section 2.5.1.

After motion estimation is performed, the temporary mosaic image is reset and initialized to the new downward-looking image. The images that were removed from the temporary mosaic image are then used for updating the main mosaic image. Since the main mosaic image is not used for motion estimation, it may be updated in a background process. This may be performed by applying various algorithms [37], [71], [72], [73], [41],

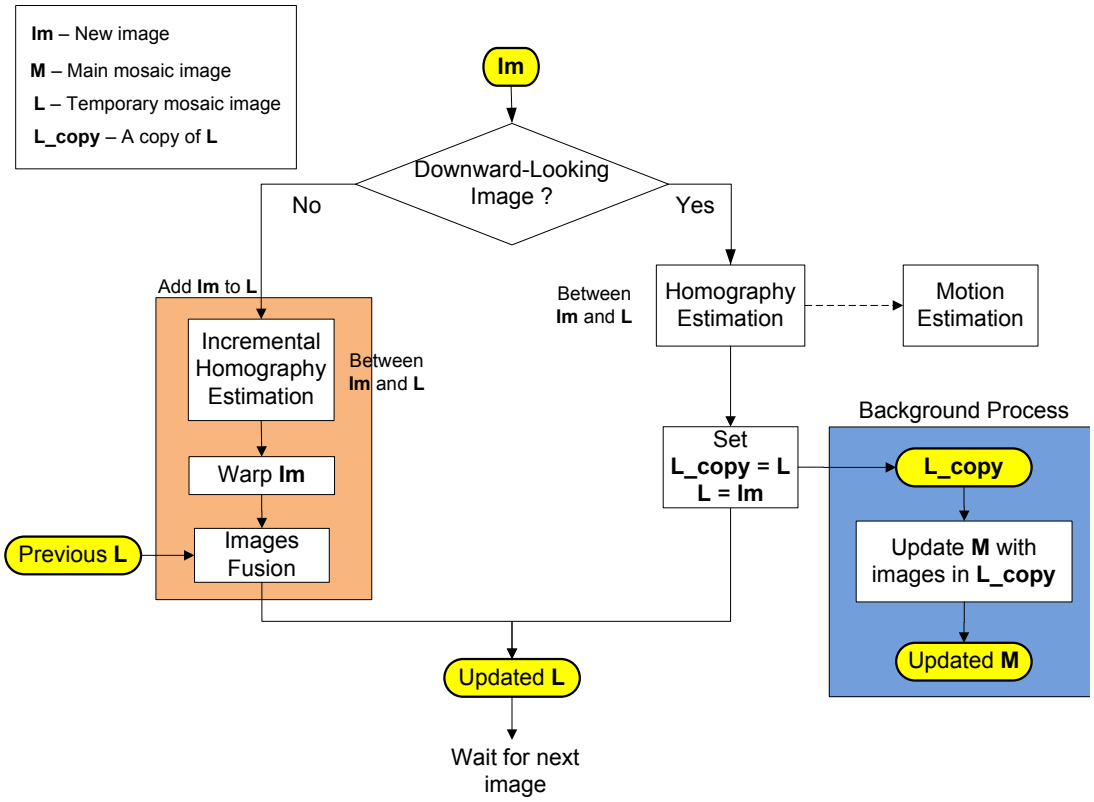


Figure 2.3: Diagram of the proposed mosaic construction method.

[74], [75], [13], [36], depending on the available computational resources.

It should be noted that loop scenarios may be also handled in this background process yielding an improved main mosaic image. In case of a loop in the trajectory, motion estimation and navigation aiding are performed based on the temporary mosaic image, following the method suggested herein. However, a different approach is required for utilizing the full potential of the available information in such an event, which is the subject of Chapter 3.

An example of the mosaic image construction process, based on real images acquired using the scanning procedure described above, is given in Figure 2.4 and Figure 2.5. The images were extracted from Google Earth, as detailed in Section 2.5. Figures 2.4(a)-2.4(d) show the construction of the temporary mosaic image, involving a camera scanning procedure that is comprised of two non-downward-looking images in each direction. The temporary mosaic image is initialized with a downward-looking image (Figure 2.4(a)) and is updated with images until a new downward-looking image is acquired (Figure 2.4(e)). One can easily notice the enlarged overlapping region between this new downward-looking image and the temporary mosaic image (Figures 2.4(d) and 2.4(e)). Figures 2.5(a) and 2.5(b) show the update of the main mosaic image, based on images from the temporary mosaic image, once a new downward-looking image was captured: Figure 2.5(a) is the

main mosaic image before the update; Figure 2.5(b) shows the main mosaic image after the update.

The following sections further elaborate on several aspects of the mosaic images construction process. The first step is to briefly review a standard method for estimating a homography matrix; this is followed by additional implementation details of the mosaic construction process.

2.2.2.1 Homography Matrix Estimation

The homography matrix [13] is a transformation that relates two images of a planar scene for a general camera motion. The homography relation is also valid for a three-dimensional scene if the camera performs a pure rotation, although this is hardly relevant for fixed-wing aerial platforms. Given some point $\mathbf{x}$ in the first image and a matching point $\mathbf{x}'$ in the second image, both expressed in homogeneous coordinates[2] [13], the following constraint can be written for the homography matrix, H:

$$\mathbf{x}'_i \cong H\mathbf{x}_i \tag{2.5}$$

where $\cong$ denotes equality up to scale. The above equation may be written explicitly as

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.6}$$

The entries of H are related to the observed scene plane parameters and to the translational and rotational motion [14]. Assuming that $\mathbf{x} = [x, y, 1]^T$, the second image coordinates (x', y') may be computed based on the inhomogeneous form of Eq. (2.6) as follows [13]:

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{2.7}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.8}$$

The homography matrix is used for updating the mosaic with each new image frame, and also for performing relative motion estimation between these images. There are various methods for estimating the homography matrix given two partially-overlapping images. The method used herein relies on [13]: First, Scale Invariant Feature Transform (SIFT) [76] features and their descriptors are computed for each of the images. If one of

[2] A homogeneous representation of a point $(x, y) \in \mathbb{R}^2$ is the vector $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$, which is defined up to scale. The homogeneous point (x_1, x_2, x_3) represents the point $(x_1/x_3, x_2/x_3) \in \mathbb{R}^2$. In particular, the homogeneous point $(x, y, 1)$ represents the point (x, y). Homogeneous points with $x_3 = 0$ represent points which lie on a plane at infinity [13].

the images is a mosaic image, an overlapping area between the two images is estimated based on information extracted from the navigation system, and the computation of SIFT features is performed only on this part of the mosaic image. Next, features from the two images are matched based on the minimum Euclidean distance of their descriptor vectors, which yields a set of matched points, $\mathcal{S}\left\{(\mathbf{x}_i, \mathbf{x}'_i)\right\}_{i=1}^{N_\mathcal{S}}$.

As the set $\mathcal{S}$ may contain wrong matches (outliers), a robust estimation technique is applied, which provides a refined set of matched points, $\mathcal{R} \subseteq \mathcal{S}$. This is performed by applying the Random Sample Consensus (RANSAC) algorithm [77] for robust estimation of the homography matrix [13], as described in the sequel. The final step in the homography estimation algorithm is to perform least-squares (LS) homography estimation based on the subset $\mathcal{R}$ of feature matches [13].

RANSAC Algorithm for Outliers Rejection

The RANSAC algorithm is applied for rejecting outliers in the matched features set $\mathcal{S}$ over the homography model, as briefly described below. Full details can be found in [13].

First, four feature matches are drawn from the available matched features set $\mathcal{S}$, based on which the homography matrix, H, is calculated. This homography matrix is then used to choose a subset of feature matches $\mathcal{T} \subseteq \mathcal{S}$ that lie within a predefined threshold, $t_{threshold}$. More specifically, a pair of point correspondences $(\mathbf{x}, \mathbf{x}')$ is chosen if

$$d_E(\mathbf{x}, H^{-1}\mathbf{x}')^2 + d_E(\mathbf{x}', H\mathbf{x})^2 < t_{threshold}^2 \tag{2.9}$$

where $d_E(.,.)$ is the Euclidean distance between two points in the same image. The number of iterations, M, should be high enough to guarantee with some probability p that at least one of the subsets $\{\mathcal{T}_i\}_{i=1}^{M}$ is free from outliers (usually $p = 0.99$). Let w be the probability that any chosen feature match is an inlier. Taking into account the fact that in each iteration 4 feature matches are drawn, the following equation may be written:

$$\left(1 - w^4\right)^M = 1 - p \tag{2.10}$$

Let $\epsilon = 1 - w$, i. e. ϵ is the probability that any chosen feature match is an outlier. Substituting ϵ into the above equation and performing some algebraic operations yields an expression for M:

$$M = \frac{\log\left(1 - p\right)}{\log\left(1 - (1 - \epsilon)^4\right)} \tag{2.11}$$

Since ϵ is unknown, it is evaluated at each iteration based on the believed number of inliers, which is the cardinality of the subset $\mathcal{T}$, and the overall number of matched points (the cardinality of the set $\mathcal{S}$). Let the respective number of matched points in $\mathcal{T}$ and $\mathcal{S}$ be $L_\mathcal{T}$ and $L_\mathcal{S}$. Thus, ϵ is calculated at each iteration according to

$$\epsilon = 1 - \frac{L_\mathcal{T}}{L_\mathcal{S}} \tag{2.12}$$

and is then used to update the parameter M based on Eq. (2.11).

After M iterations, the subset with the maximum number of point matches is chosen among $\{\mathcal{T}_i\}_{i=1}^{M}$. This subset, denoted by $\mathcal{R}$, contains the set of point matches that were identified by the RANSAC algorithm as inliers.

2.2.2.2 Non Downward-Looking Images

If the new image is a non downward-looking image, the homography estimation is *incremental*, as described below. The purpose of the incremental estimation is the reduction of the accumulated alignment errors in the temporary mosaic image, while updating the mosaic with new non downward-looking images.

The method proposed here is an adaptation of the procedure suggested by [38], [15] for the camera scanning method used herein. Denote by r the index of the most recent downward-looking image, defining the reference frame in which the current temporary mosaic image is expressed. Each new image I_k, which is a *non* downward-looking image, is matched against the previous image I_{k-1}, yielding a homography between these two images $H_{k\to k-1}$. The next step is to calculate an intermediate homography matrix between the new image I_k and the current temporary mosaic image, relying on information computed for the previous image I_{k-1}:

$$H_{k\to r}^{I} = H_{k-1\to r} \cdot H_{k\to k-1} \tag{2.13}$$

where $H_{k-1\to r}$ is the homography matrix transforming from the previous image, I_{k-1}, to the current temporary mosaic image. This homography matrix was calculated and saved while processing image I_{k-1}. Once this homography is available, the new image I_k is warped towards the temporary mosaic image using the homography matrix $H_{k\to r}^{I}$, yielding the warped image $\tilde{I}_k^r$.

Ideally, the warped image and the temporary mosaic image should be aligned; however, this is usually not true in practice due to homography estimation errors. To improve the estimation, a *correction homography* between the warped image, $\tilde{I}_k^r$, and the current temporary mosaic image, is estimated by applying the standard homography estimation technique discussed in Section 2.2.2.1 on these two images. This homography, H_{corr}, is used to correct the estimated intermediate homography matrix between the new image and the temporary mosaic image,

$$H_{k\to r} = H_{corr} \cdot H_{k\to r}^{I} \tag{2.14}$$

Finally, the new image, I_k, is warped towards the current temporary mosaic image using the improved homography matrix, $H_{k\to r}$, followed by an integration of the two images into an updated mosaic, using one of the available techniques [37], [13]. In addition, $H_{k\to r}$ is saved for future use with new non downward-looking images. The process repeats for an each new image that was not taken when the camera was looking downward.

2.2.2.3 Downward-Looking Images

Once a new downward-looking image, I_d, is captured, a direct estimation of the homography matrix (cf. Section 2.2.2.1) relating this new image to the current temporary mosaic image is performed. During this process, only part of the temporary mosaic image is used. Since height above ground level is unknown, this region may be approximately calculated based on the platform current heading and altitude. The estimated homography matrix is then used for motion estimation (cf. Section 2.3). The next step is to remove all the images from the temporary mosaic image and to initialize it with I_d.

Let r denote the index of the previous downward-looking image. Now, the images $\{I_i\}_{i=r+1}^{d}$ should be used for updating the main mosaic image. This may be performed in a background process, using various approaches, since the main mosaic image is not required for motion estimation. The approach that was implemented in this work is to use the incremental homography estimation technique, discussed above, for adding the images $\{I_i\}_{i=r+1}^{d}$ to the main mosaic image.

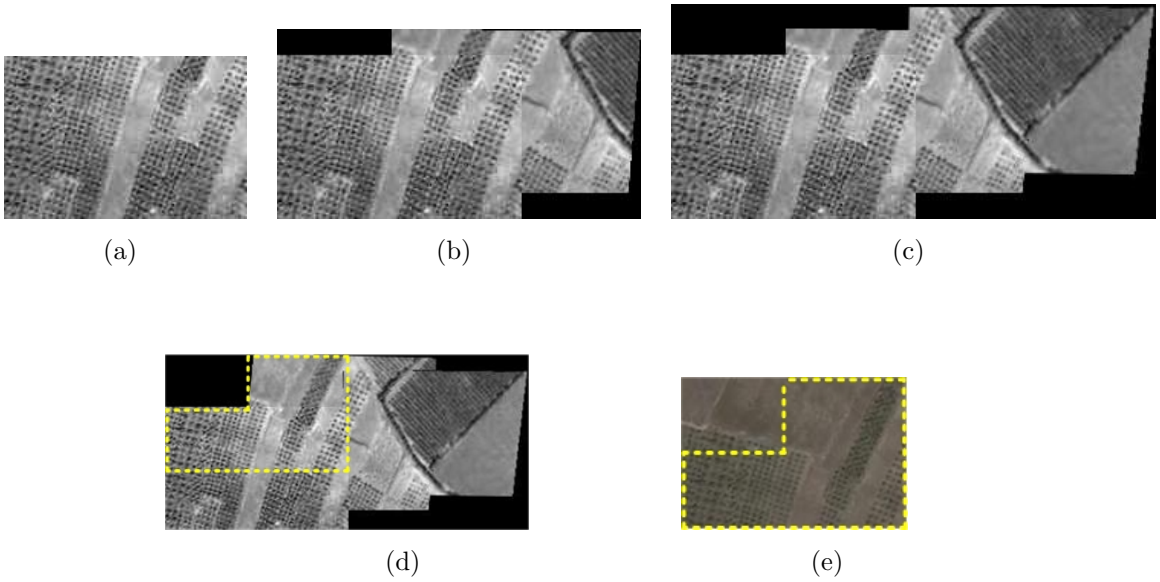
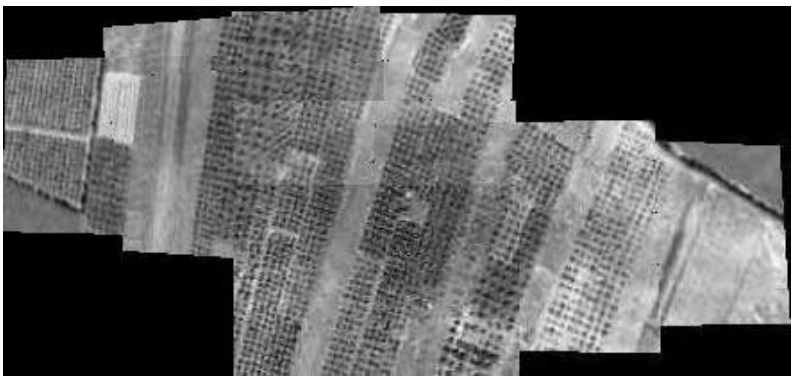


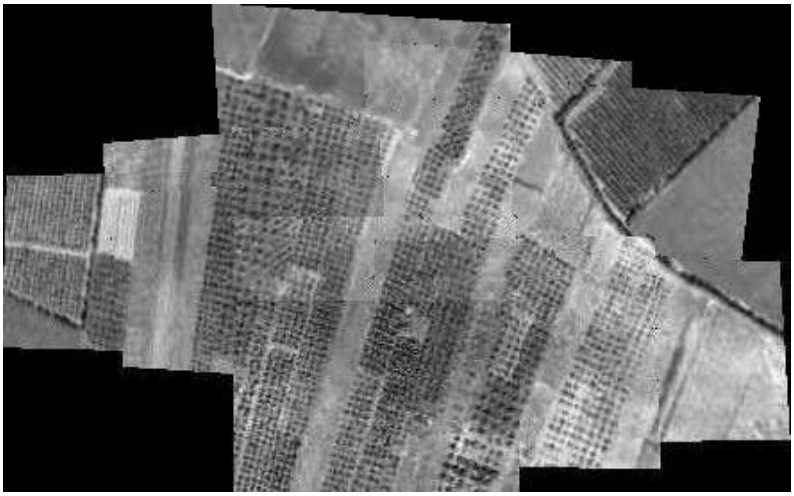
Figure 2.4: Mosaic images construction example. (a)-(d) Temporary mosaic image construction. (e) A new downawrd-looking image. An enlarged overlapping area between this image and the temporary mosaic image is shown in (d) and (e).

2.3 Image-Based Motion Estimation

This section focuses on motion estimation based on a previously-estimated homography matrix. The camera motion between any two views of a planar scene, related by a



(a)



(b)

Figure 2.5: Mosaic images construction example. (a) Previous main mosaic image. (b) Updated main mosaic image based on images from the temporary mosaic image (Figure 2.4(d)) and the new downward-looking image (Figure 2.4(a)).

homography matrix H, is encoded in H according to [14]:

$$H = K' \left[R - \frac{\mathbf{t}}{z} \mathbf{n}^T \right] K^{-1} \tag{2.15}$$

where K', K are the calibration matrices at two image time instances, assumed to be known, $\mathbf{t}$ is the translation vector, R is the rotation matrix, z is the scene depth and $\mathbf{n}$ is a unit vector normal to the scene plane. Since the process of motion estimation is based only on the information provided by the camera, the translation motion between two images can be estimated only up to scale (i. e. only the translation direction can be estimated).

A method for extracting the motion parameters from Eq. (2.15) was suggested in [14], where it was proven that there are at most two valid sets of solutions $(\mathbf{t}, R, \mathbf{n})$. The

correct solution out of these two alternatives can be chosen based on $\mathbf{n}$ while relying on previous estimates [14]. The implementation of the estimation process in this work involves yet another phase, which will be described in the next section. This phase allows improved precision motion estimation assuming a standard approach for estimating the homography matrix (cf. Section 2.2.2.1).

It should be noted that any two views of a *non*-planar scene are related through the more complex epipolar geometry, from which relative motion parameters may be extracted as well [13] via, e.g., the fundamental matrix. However, when assuming a narrow-FOV camera, the epipolar geometry method tends to become ill-conditioned, due to the limited ground information captured by the camera, resulting in a semi-planar scene.

2.3.1 Implementation of Motion Estimation Assuming a Planar Scene

As mentioned in Section 2.2.2.1, the homography estimation process involves the RANSAC algorithm for robust outliers rejection. The output of this algorithm is a subset $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^{N_\mathcal{R}}$, $\mathcal{R} \subseteq \mathcal{S}$, of feature matches that are considered to be inliers. These are then used for LS estimation of the homography matrix. When considering ideal features, this process yields the same results when executed several times. However, the solution varies from one execution to another for noisy data (for a given threshold value), since each execution may yield a different features subset group, and hence a different estimation of the homography matrix (and motion parameters).

More specifically, assume that the extracted SIFT features image coordinates are corrupted by noise. As a consequence, the computed set of all point matches $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^{N_\mathcal{S}}$ is also corrupted by noise, and in addition may contain false matches (outliers). In each iteration of the RANSAC algorithm, four point matches are drawn and used to compute a homography matrix, which is then utilized to construct a subset $\mathcal{T}$ of point matches that are consistent with this homography matrix. Thus, a pair of point correspondences $(\mathbf{x}, \mathbf{x}')$ is chosen if Eq. (2.9) is satisfied:

$$d_E(\mathbf{x}, H^{-1}\mathbf{x}')^2 + d_E(\mathbf{x}', H\mathbf{x})^2 < t^2_{threshold} \tag{2.16}$$

Consider such two different iterations yielding the subsets $\mathcal{T}_1$ and $\mathcal{T}_2$, and assume that these subsets do not contain any false matches. In each of these subsets, the homography matrix was computed based on a different set of drawn 4 point matches. These two homography matrices, H_1 and H_2, are expected to be different, despite the fact that they were computed based on inlier point matches, since all the features in $\mathcal{S}$, and in particular the drawn features, are corrupted by noise.

In the next step of the RANSAC algorithm, all point matches in $\mathcal{S}$ are checked for consistency with the homography matrix, $H_i, i = \{1, 2\}$, according to Eq. (2.16). Only point matches that agree with the condition (2.16) are added to the subset $\mathcal{T}_i$. Since both

homography matrices are legitimate but different, it is obvious from Eq. (2.16) that the two subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ will be identical for a sufficiently large value of $t_{threshold}$. However, decreasing the value of $t_{threshold}$ will yield different subsets $\mathcal{T}_1$, $\mathcal{T}_2$, starting from some critical value. This critical value is influenced by the image features noise characteristics, and is therefore a function of the observed scene: high-texture scenes are likely to be corrupted with less noise compared to low-texture scenes, since in the former case the features can be localized with improved precision. Thus, a specific value of $t_{threshold}$ might yield identical subsets $\mathcal{T}_1$, $\mathcal{T}_2$ in certain scenarios, and different subsets in other scenarios.

The above conclusion is valid also for the output from the RANSAC algorithm (the inliers subset, $\mathcal{R}$), as it is merely one of the subsets constructed during its iterations. Thus, sequential activation of the RANSAC algorithm might give different subsets of $\mathcal{R}$, meaning that the LS homography estimation will yield a number of different homography matrices $\{H_i\}$, and consequently, a set of motion parameters extracted from each homography matrix H_i. This process of homography matrix estimation, involving a sequential execution of the RANSAC algorithm, is denoted in this section as *sequential homography estimation*.

Given the set of motion parameters, $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N}$, obtained from the sequential homography estimation process, one can employ different logic for automatically choosing the most accurate motion estimation. The logic implemented in this work consists of the following steps.

Denote $|\mathbf{q}| = \left|\,[q_1, \ldots, q_n]^T\right| \triangleq [|q_1|, \ldots, |q_n|]^T$. Define the mean unit vector normal to a scene plane, based on the normal unit vectors $\{\mathbf{n}_i^{prev}\}_{i=1}^{N_{prev}}$ from estimates of N_{prev} previous images, as

$$\mathbf{n}_{\mu}^{prev} = \frac{\Sigma_{i=1}^{N_{prev}} |\mathbf{n}_i^{prev}|}{\left\| \Sigma_{i=1}^{N_{prev}} |\mathbf{n}_i^{prev}| \right\|} \tag{2.17}$$

Compute a score for each available solution $(\mathbf{t}_i, R_i, \mathbf{n}_i)$ based on the proximity of its normal unit vector $\mathbf{n}_i$ to the mean unit vector $\mathbf{n}_{\mu}^{prev}$:

$$s_i = | < \mathbf{n}_{\mu}^{prev}, \mathbf{n}_i > | \tag{2.18}$$

where $< ., . >$ is the inner product operator. Calculate the mean and the standard deviation (s_μ, s_σ) of the set $\{s_i\}_{i=1}^{N}$, and reject all the solutions whose score is lower than $s_\mu - s_\sigma$. Denote by N_1 the number of remaining solutions.

Next, a translation matrix $\Lambda = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3)$ is constructed from the absolute values of the translation motion estimation vectors in the set of remaining solutions ($\Lambda \in \mathbb{R}^{N_1 \times 3}$):

$$\Lambda = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) \equiv \begin{bmatrix} |\mathbf{t}_1^T| \\ |\mathbf{t}_2^T| \\ \vdots \\ |\mathbf{t}_{N_1}^T| \end{bmatrix} \tag{2.19}$$

Each of the Λ columns is examined for outliers based on the distribution of its values. More specifically, a histogram of the vector $\boldsymbol{\lambda}_i$ is computed over N_1 slots in the range $[\min(\boldsymbol{\lambda}_i), \max(\boldsymbol{\lambda}_i)]$, followed by a rejection of entries in $\boldsymbol{\lambda}_i$ which do not appear in clusters. Denote by N_2 the number of remaining solutions after this step was applied on all three columns of Λ.

Finally, a solution is chosen among the remaining-solutions set $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$, whose normal is the closest to $\mathbf{n}_\mu^{prev}$, i. e., a solution with the highest score s_i.

If the mean normal vector from previous images is unavailable, a solution $(\mathbf{t}, R, \mathbf{n})$ is chosen whose normal vector $\mathbf{n}$ is the closest to the mean normal vector of all the other solutions in $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$, i. e., a solution i that maximizes $< \mathbf{n}_i, \mathbf{n}_\mu >$ where $\mathbf{n}_\mu$ is defined as

$$\mathbf{n}_\mu = \frac{\Sigma_{i=1}^{N_2} |\mathbf{n}_i|}{\left\| \Sigma_{i=1}^{N_2} |\mathbf{n}_i| \right\|} \tag{2.20}$$

The sequential estimation process of the motion parameters described above is summarized in Algorithm 1. The improvement in the estimation precision is demonstrated in Section 2.5.1.

Algorithm 1 Sequential Estimation of the Motion Parameters

1: Run N times the homography estimation routine, given in Section 2.2.2.1, and calculate the solution set from the estimated homography matrices: $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N}$.
2: **if** at least one image was already processed **then**
3: Compute a score s_i for each solution, based on Eqs. (2.17) and (2.18).
4: Reject solutions whose score is lower than $s_\mu - s_\sigma$, where (s_μ, s_σ) are the mean and standard deviation values of the computed set of scores $\{s_i\}_{i=1}^{N}$.
5: Construct a translation matrix Λ based on Eq. (2.19) and examine each of its columns for outliers. Solutions that contain outliers are rejected, yielding a refined set $\{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ of solutions.
6: **if** at least one image was already processed **then**
7: Choose a solution $(\mathbf{t}, R, \mathbf{n}) \in \{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ with the highest score.
8: **else**
9: Choose a solution $(\mathbf{t}, R, \mathbf{n}) \in \{(\mathbf{t}_i, R_i, \mathbf{n}_i)\}_{i=1}^{N_2}$ which maximizes $< \mathbf{n}, \mathbf{n}_\mu >$, where $\mathbf{n}_\mu$ is computed according to Eq. (2.20).

2.4 Fusion of Image-Based Relative Motion Estimation with a Navigation System

In this section we present a method for fusing the image-based estimated camera relative motion with a navigation system of an aerial platform. A measurement model is developed

that relates the image-based estimated relative motion with the accumulating navigation errors of a standard inertial navigation system. The data fusion is performed using an indirect implicit extended Kalman filter [78] that estimates the navigation parameter errors instead of the parameters themselves. These estimated errors are then used for correcting the navigation parameters. The state vector is given by Eq. (1.10):

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\boldsymbol{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T \tag{2.21}$$

Once an estimate of $\mathbf{X}$ is available, it is used both to correct the output of the navigation system, and to provide a feedback to the inertial sensors (cf. Figure 2.1): The estimated position, velocity and attitude errors are used to correct the INS output. These components in the state vector are then reset, since they have been incorporated into the navigation system, i. e. the first 9 components of the a posteriori estimation at some time instant t_k, $\hat{\mathbf{X}}_{k|k}$, are set to zero. The covariance matrix is not changed, so it represents the uncertainty in the platform's navigation parameters estimation, i. e. the navigation errors. In addition, the IMU measurements readings are corrected with the most recent available estimations of drift and bias parameters at the frequency of the inertial sensors readings[3], which is much higher than the frequency of the IEKF updates.

It is worth stressing that the state vector is a constant-size vector, $\mathbf{X} \in \mathbb{R}^{15}$. Another possible approach is to use a direct data-fusion technique and relate the vision-based estimated motion to an augmented state vector comprised of the platform current parameters (e. g. position, velocity) and past poses for each captured image [12]. Yet, as in the case of SLAM methods, this approach requires increasing computational resources (since the state vector size increases with time), and therefore the use of the indirect fusion approach with a constant-size state vector is preferred.

It is assumed here that the relative motion parameters between each two image time instances, $t = t_1$ and $t = t_2$, were already extracted by the image processing module. Thus, the camera relative rotation matrix, $\hat{R}_{C_1}^{C_2}$, transforming from the camera axes at time t_2 to the camera axes at time t_1, is known. In addition, the relative translation, $\hat{\mathbf{t}}_{1\to 2}^{C_2}$, is known up to some scale, γ.

The estimated relative motion is reformulated into residual measurements $\mathbf{z}_{translation}$, $\mathbf{z}_{rotation}$, which are injected into the filter:

$$\mathbf{z}_{translation} = \left[\mathbf{Pos}_{Nav}(t_2) - \mathbf{Pos}_{Nav}(t_1)\right]^{L_2} \times C_{L_2,Nav}^{C_2} \hat{\mathbf{t}}_{1\to 2}^{C_2} \tag{2.22a}$$

$$\mathbf{z}_{rotation} = \begin{pmatrix} \tan^{-1}\left[\frac{D(3,2)}{D(3,3)}\right] \\ -\sin^{-1}\left[D(1,3)\right] \\ \tan^{-1}\left[\frac{D(1,2)}{D(1,1)}\right] \end{pmatrix} \qquad D \doteq C_{C_1,Nav}^{C_2}\left[\hat{R}_{C_1}^{C_2}\right]^T \tag{2.22b}$$

[3] Another possible variation of this is to estimate the residual bias and drift values, while maintaining the estimations of actual bias and drift parameters outside the filter, as discussed in Section 1.3.2. In this case the whole state vector should be reset after each update step.

where $C_{L_2}^{C_2}$ is the DCM transforming from C to LLLN at the time instance $t = t_2$; $T_{C_1}^{C_2}$ is the DCM transforming from C at $t = t_2$ to C at $t = t_1$; and **Pos** is the platform's position. The subscript Nav denotes the parameters that are taken from the navigation data.

The state vector and the residual measurements are related via a measurement equation

$$\mathbf{Z} \doteq \begin{pmatrix} \mathbf{z}_{translation} \\ \mathbf{z}_{rotation} \end{pmatrix} = H\mathbf{X} + \begin{pmatrix} \mathbf{v}_{tr} \\ \mathbf{v}_{rot} \end{pmatrix} \tag{2.23}$$

where H is the measurement matrix

$$H = \begin{bmatrix} 0_{3\times3} & H_{\Delta V}^{tr} & H_{\Delta\Psi}^{tr} & H_{d}^{tr} & H_{b}^{tr} \\ 0_{3\times3} & 0_{3\times3} & H_{\Delta\Psi}^{rot} & H_{d}^{rot} & 0_{3\times3} \end{bmatrix}, \tag{2.24}$$

$\mathbf{v}_{tr}$ is given by

$$\mathbf{v}_{tr} = [\mathbf{Pos}_{True}(t_2) - \mathbf{Pos}_{True}(t_1)]^{L_2} \times \left[C_{L_2,Nav}^{C_2} \hat{\mathbf{t}}_{1\to2}^{C_2} - C_{L_2,True}^{C_2} \mathbf{t}_{1\to2,True}^{C_2} \right], \tag{2.25}$$

and $\mathbf{v}_{rot}$ represents rotation motion estimation errors and linearization errors. Note that since an IEKF formulation is used, the measurement noise terms are not necessarily white [4]. The development of the above measurement equation and the explicit expression for H are given in Appendix A.

The estimated state vector is initialized to zero, since the actual initial navigation errors are unknown, while the estimation error covariance matrix is set to the believed levels of navigation errors. Although these values are usually known from the performance specifications of the inertial sensors, in all practical applications the initial covariance and process noise covariance matrices are adjusted during the tunning process.

The propagation step involves computation of an a priori covariance matrix $P_{k+1|k}$ according to

$$P_{k+1|k} = \Phi(k+1,k) P_{k|k} \Phi^T(k+1,k) + Q_k \tag{2.26}$$

where $\Phi(k+1,k), P_{k|k}, Q_k$ are the process discrete system matrix, a posteriori covariance matrix, and the discrete process noise covariance matrix, respectively. The discrete system matrix Φ is given by Eq. (1.8). The discrete process noise covariance matrix Q can be calculated as

$$Q = \int_{t_a}^{t_b} \Phi(t_b,\tau) Q_c \Phi(t_b,\tau)^T d\tau \tag{2.27}$$

with $Q_c = E[\boldsymbol{\omega}_c \boldsymbol{\omega}_c^T]$ and $\boldsymbol{\omega}_c$ being the continuous process noise (cf. Section 1.3.2). In practice, however, Q is set during the filter tuning process.

The propagation of the state vector is given by

$$\hat{\mathbf{X}}_{k+1|k} = \Phi(k+1,k) \hat{\mathbf{X}}_{k|k} \tag{2.28}$$

However, since the first 9 components of $\hat{\mathbf{X}}_{k|k}$ are used for correcting the strapdown integration (see above) after the update step and then reset, and the other 6 components are random constants (i. e., $\hat{\mathbf{X}}_{k|k} = [\mathbf{0}_{1\times 9} \quad \mathbf{d}^T \quad \mathbf{b}^T]^T$), Eq. (2.28) is equivalent to $\hat{\mathbf{X}}_{k+1|k} = \hat{\mathbf{X}}_{k|k}$.

After performing the propagation step, a measurement update is performed given the motion estimation $(\mathbf{t}^{C_2}_{1\to 2}, R^{C_2}_{C_1})$ from the image processing module. First, the Kalman filter gain matrix is computed according to

$$K_{k+1} = P_{k+1|k}H^T_{k+1}\left[H_{k+1}P_{k+1|k}H^T_{k+1} + R_{k+1}\right]^{-1} \tag{2.29}$$

The matrix R_{k+1} in Eq. (2.29) is a measurement noise covariance matrix, which is of the following block-diagonal form:

$$R_{k+1} = \begin{bmatrix} R^{tr}_{k+1} & 0_{3\times 3} \\ 0_{3\times 3} & R^{rot}_{k+1} \end{bmatrix} \tag{2.30}$$

where R^{tr}, R^{rot} are the translation and rotation measurement noise covariance matrices, respectively. While R^{rot} is a constant matrix, an adaptive translation measurement noise covariance matrix R^{tr} is calculated based on Eq. (2.25):

$$R^{tr} = -\left[\mathbf{Pos}^{L_2}_{Nav}(t_2) - \mathbf{Pos}^{L_2}_{Nav}(t_1)\right]^{\wedge} R^{est} \left[\mathbf{Pos}^{L_2}_{Nav}(t_2) - \mathbf{Pos}^{L_2}_{Nav}(t_1)\right]^{\wedge} \tag{2.31}$$

where R^{est} is a 3×3 tuning matrix that represents the level of accuracy in the vision-based estimation of the translation direction and $(.)^{\wedge}$ denotes the matrix cross-product equivalent. For example, in the experiments with real imagery presented in Section 2.5.2, it was assumed to be close to $I_{3\times 3}$. It should be noted that the matrices R^{est}, R^{rot} may also be estimated as part of the image-based motion estimation procedure [16].

Once the gain matrix K is available, a posteriori values of the state vector and covariance matrix are computed using the standard IEKF formulas [78], [4]:

$$\hat{\mathbf{X}}_{k+1|k+1} = \hat{\mathbf{X}}_{k+1|k} + K_{k+1}\mathbf{Z}_{k+1} \tag{2.32}$$

$$P_{k+1|k+1} = \left[I - K_{k+1}H_{k+1}\right]P_{k+1|k}\left[I - K_{k+1}H_{k+1}\right]^T + K_{k+1}R_{k+1}K^T_{k+1} \tag{2.33}$$

2.4.1 Fictitious Velocity Measurement

Some of the errors in the image-based relative motion estimation may be projected onto the unobservable states, analyzed in Section 2.6, and yield poor estimation performance even when compared with the pure inertial case. In order to mitigate this phenomenon, several heuristic methods may be considered. For the current implementation, a fictitious ideal velocity measurement was used in addition to the relative motion measurements to overcome the scaling ambiguity, so that

$$(\mathbf{V}^L_{true})^T \Delta\mathbf{V} = \mathbf{0}, \tag{2.34}$$

Namely, the velocity errors in the direction of the flight are assumed to be zero, and hence errors from the image-processing block are essentially not projected onto this direction. The term $\mathbf{V}^L_{true}$ refers to the true value of the platform velocity in the LLLN system. Since this velocity is unknown, it is replaced by the platform velocity $\mathbf{V}^L$ taken from the navigation system.

A Kalman filter gain matrix, K, is computed according to Eq. (2.29) based on an a priori covariance matrix $P_{k+1|k}$, an augmented measurement matrix, $H_{aug} = [H^T, H_v^T]^T$, and an augmented measurement noise covariance matrix, R_{aug}, where

$$H_v = \begin{bmatrix} \mathbf{0}_{1\times3} & (\mathbf{V}^L)^T & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix} \tag{2.35}$$

and H is the measurement matrix of Eq. (2.24).

The augmented measurement noise covariance matrix R_{aug} is given by

$$R_{aug} = \begin{bmatrix} R & 0 \\ \mathbf{0}_{1\times3} & R_v \end{bmatrix} \tag{2.36}$$

where R is given in Eq. (2.31) and R_v is the fictitious velocity (FV) measurement noise covariance matrix, which constitutes a tuning parameter. Small-valued entries in R_v indicate that this additional measurement is reliable, and therefore other measurements will have a minor influence on the entries of the gain matrix K, corresponding to position and velocity along the flight heading. This, in turn, prevents from erroneous image-based relative motion measurements to affect the unobservable states.

Once K is computed, the column related to the fictitious velocity measurement is discarded; in this way, the measurement limits the corrections in the direction of the flight but does not render the problem inconsistent, since the measurement is not actually performed. The advantage of using the FV measurement is demonstrated in Section 2.5.2.2 for vision-aided navigation using two-view based motion estimations; thus, all the results of mosaic-aided navigation, presented in Section 2.5.2.3, were obtained with the FV measurement active. Note that the FV measurement does not limit the platform's motion to any specific type. This is in contrast, for example, to nonholonomic constraints that may be applied only for land vehicles [79]. In addition, due to the varying quality of the image measurements (cf. Section 2.5.1), a measurement-rejection mechanism must also be used to avoid fusion of low-quality measurements or other outliers.

2.4.2 Computational Requirements

The overall computational requirements of the proposed navigation aiding architecture consist of applying a standard Kalman filter for constant-size state and measurement vectors of 15 and 6 elements, respectively, and of the image processing phase. The latter consists of a temporary mosaic image construction, which is limited to a few images (4 in the current example), and of motion estimation. The main mosaic image is not used in the navigation aiding scheme.

As discussed earlier, the inclusion of a new image into a temporary mosaic image involves the computation of SIFT features, estimation of the homography matrix, warping the new image and fusing the two images. Only a partial area in the temporary mosaic image is used for calculating SIFT features. These operations require modest computational resources and pose no difficulties for real-time operation.

In conventional SLAM, as opposed to the proposed method the state vector is augmented with relevant data from each captured image, and thus the whole state vector needs to be updated each time. Denoting by d the number of elements that are added to the state vector once a new image is acquired, SLAM generates, after 100 seconds of flight (assuming the same image sampling frequency of 5 Hz) a state vector of $15 + 500d$ elements that should be processed in real time, which is far more demanding than applying a Kalman filter to a 15-element state vector and constructing a temporary mosaic image[4] (cf. Section 2.4.2), as suggested in the new approach.

2.5 Results

This section contains simulation and experimental results of the presented mosaic-aided navigation method. The simulation is comprised of the following modules: A navigation module, a camera scanning module, and an image processing module.

The navigation phase consists of the following steps: (a) Trajectory generation; (b) velocity and angular velocity increments extraction from the created trajectory; (c) IMU error definition and contamination of pure increments by noise; and (d) strapdown calculations. The strapdown mechanism provides, at each time step, the calculated position, velocity and attitude of the platform. In parallel to the strapdown calculations, at a much slower rate, Kalman filter calculations are performed based on the available measurements. At the end of each filter cycle, the strapdown output is updated with the estimated state vector of the filter. See also Section 1.3.3.

The camera scanning module provides camera angle commands that yield a continuous scan, according to the camera scanning procedure discussed in Section 2.2.1.

The image processing module constructs mosaic images and performs motion estimation each time a downward-looking image is acquired (cf. Sections 2.2.2 and 2.3). The inputs to this module are real images obtained from Google Earth[5] based on the proposed camera scanning procedure. In addition, the module is capable of calculating an ideal camera motion based on the true platform trajectory, without actually using any real images. Naturally, in this mode of operation the mosaic images are not constructed. The

[4]The construction of the main mosaic image in a background process may involve different algorithms. However, while some of them may be computationally expensive (such as global optimization), they are to be applied *only* in case of a loop in a trajectory, or in some low frequency. This is in contrast to the constantly increasing high-computational requirements of the update step in the SLAM approach.

[5]http://earth.google.com/index.html, last accessed June 2009.

Table 2.1: Trajectory Parameters

Parameter	Description	Value	Units
λ	Initial latitude	32.8285005298	deg
Λ	Initial longitude	35.1479222075	deg
alt	Initial altitude above sea level	1500	m
$\mathbf{V}^L$	Velocity in LLLN system	$(100, 0, 0)^T$	m/s
Ψ	Platform attitude	$(0, 0, 0)^T$	deg

ideal motion estimations are used as baseline for evaluating the best possible performance of the proposed method, since motion estimation based on real images will be imperfect.

The experiments presented in this section are based on real image sequences acquired using Google Earth, which contains 3D geo-data of Earth based on real imagery, i. e. a 3D environment based on real images and a digital terrain model. For this purpose, an interface that bridges between the navigation simulation and Google Earth was developed. The interface allows to obtain images from Google Earth at specified camera position and attitude, further details are provided in Appendix A.3.

It should be noted that any two images of the same ground region observed from different viewpoints will yield a correct relative image transformation. However, this approach does not mimic real-world images perfectly, since it lacks the effect of lighting variations when some region is observed from different directions. Yet, since the presented trajectories do not involve loops[6], the implemented camera scanning procedure will take different images of the same ground region under similar conditions. Thus, the effect of varying lighting conditions is expected to be marginal.

Unless otherwise stated, the experiments presented in this chapter were conducted while the platform performed a straight-and-level north-heading trajectory, whose initial conditions are given in Table 2.1. The observed scene along this trajectory is of a planar nature with about 50 m elevation above sea level.

The results are presented in the next sections. Improved image-based motion estimation when handling difficult scenarios is demonstrated in Section 2.5.1. Next, results of vision-aided navigation are shown in several steps. First, statistical results of fusing ideal motion estimations with an INS are presented. Section 2.5.2.2 demonstrates the improvement in navigation performance when using the FV measurement. Finally, Section 2.5.2.3 presents results of mosaic-aided navigation.

[6] As mentioned in Section 2.2.2, the described method in this chapter is not intended for handling loop scenarios. These scenarios can be handled using the algorithm presented in Chapter 3.

2.5.1 Mosaic-based Motion Estimation

Before presenting the results for mosaic-based motion estimation, we demonstrate the improvement in the precision of motion estimation when applying the sequential estimation procedure, summarized in Algorithm 1. In the current implementation, this routine was executed with $N = 10$.

Figure 2.6 presents the results of translation motion estimation when a low-texture scene is observed by a narrow-FOV camera. A pair of such images are shown in Figures 2.6(a) and 2.6(b).

The improvement in the estimation precision is clearly evident in Figure 2.6(c), where the same pair of images was used to perform motion estimation with and without the sequential estimation procedure. The figure presents a cumulative distribution function (CDF) of errors in the estimation of the translation direction; the x-axis values represent different thresholds of errors (in degrees), while the y-axis represents the percentage of estimations with an estimation error lower than the threshold values.

The results in Figure 2.6(c) were obtained by retrieving motion parameters from a conventionally-estimated homography matrix (cf. Section 2.2.2.1) and by applying the sequential estimation procedure of motion parameters (Algorithm 1). Both of the methods were executed 100 times on a pair of low-texture images taken with a $7^o \times 4^o$-FOV camera (Figures 2.6(a) and 2.6(b)). The advantage of the sequential estimation method is significant. For example, nearly 80% of the estimation errors were below 20^o when applying sequential estimation, compared to only 50% with a standard homography estimation.

Next, the performance of the proposed mosaic-based motion estimation method is presented. The results are compared with a standard two-view method, in which the motion estimation is based on camera-captured images, without constructing the mosaic image.

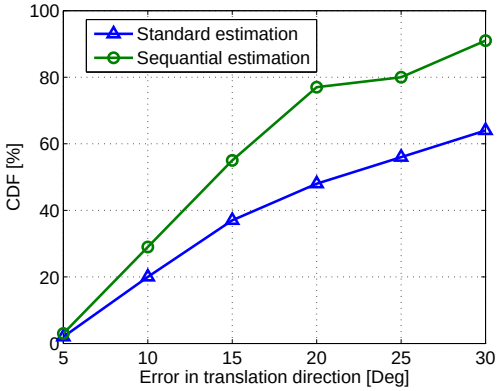
In both cases the motion parameters are estimated using the proposed sequential estimation method. Image sequences were acquired from Google Earth, using the same trajectory, for each of the examined motion estimation methods: Images for the traditional two-view motion estimation method were captured using a constant downward-looking camera at a 1 Hz frequency, while images for the mosaic-based motion estimation method were captured according to the camera scanning procedure at a 5 Hz frequency. Among all the images acquired during the camera scanning, the downward-looking images were captured every second, and therefore motion estimation in the mosaic-based method was also applied at a 1 Hz frequency (cf. Figure 2.3).

The results are presented in Figure 2.7, showing the CDF of the translation direction estimation error and of the rotation estimation error. The shown rotation error is the maximum value of the error in the estimated rotation vector, i. e.

$$\Delta\eta \triangleq \max(|\Delta\phi|, |\Delta\theta|, |\Delta\psi)| \tag{2.37}$$

where $\Delta\phi, \Delta\theta, \Delta\psi$ are the Euler angle errors of the estimated rotation matrix, computed

(a) (b)



(c)

Figure 2.6: (a),(b) Images of a low-texture scene captured from Google Earth by a $7^o \times 4^o$ FOV camera. (c) Motion sequential estimation vs. standard estimation over the pair of images presented in (a),(b): CDF of the translation direction estimation error. Significantly improved estimation accuracy in favor of sequential estimation.

from the DCM R_{err}:

$$R_{err} \equiv R_{true} \cdot R^T \tag{2.38}$$

Here R_{true} and R are the true and estimated values of the rotation matrix, respectively.

It is important to understand when the mosaic-based method is expected to outperform the two-view-based method. In the context of motion estimation, the two methods differ only in the size of the image overlap region. Due to the camera scanning process, the constructed mosaic image contains an enlarged overlapping region compared to the overlapping region between two regular images. This region is comprised of the original overlapping area between two regular images and *an additional* overlapping region - see a schematic illustration in Figure 2.2(a) and a mosaic example image in Figure 2.4 and Figure 2.5. However, since the mosaic construction process is by itself affected by errors, features from the additional overlapping area tend to be of lower-quality compared to those from the original overlapping region, while features from the original overlapping region are of the same quality in both cases (the camera-captured image and the mosaic image), due to the mosaic construction process (cf. Section 2.2.2). Thus, there is an in-

herent tradeoff: On one hand, the mosaic provides an increased number of features, while on the other hand, part of the features are of a lower quality. Hence, the performance of the mosaic-based method is expected to be superior over the two-view framework in "difficult" scenarios, in which the overlapping region between the two captured images yields a small number of high-quality features. In other words, it is expected that the mosaic will outperform the two-view method for the narrow-FOV camera and low-texture scenes.

The above observation is clearly evident in Figure 2.7, which describes the scenario of a narrow-FOV camera ($5^o \times 3^o$) and a low-texture scene. This relatively small FOV is common in many airborne applications. It can be seen that the mosaic-based motion estimation yields considerably better results compared to the two-view motion estimation. For example, in case of the translation direction estimation (Figures 2.7(a)), 50% of the estimates using the mosaic method are provided with an accuracy better than 15^o, compared to only 20% using the two-view method. As will be seen in the next section, these motion estimations can be effectively utilized for improving the performance of navigation systems.

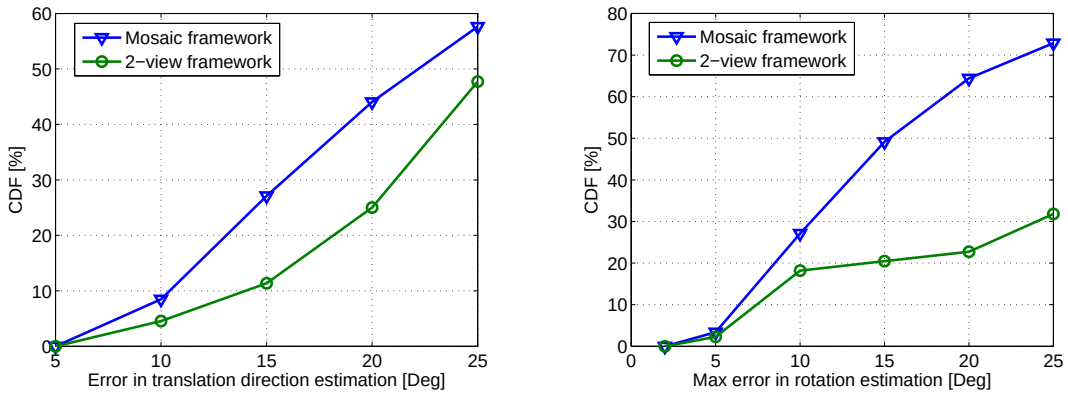


Figure 2.7: Image-based motion estimation accuracy for a low-texture scene and a narrow-FOV camera of $5^o \times 3^o$ (CDF). The mosaic framework significantly improves the estimation accuracy compared to a traditional two-view method.

2.5.2 Mosaic-Aided Navigation

This section contains simulation results of the developed mosaic-aided navigation method following the fusion technique discussed in Section 2.4. First, navigation results are presented assuming an ideal motion estimation (without using any real images). Next, the contribution of the FV measurement is demonstrated, followed by results of mosaic-aided navigation. These results are compared to two-view aided navigation. It is noted that the simulation runs were performed without a captive flight stage.

Table 2.2: Initial Navigation Errors and IMU Errors

Parameter	Description	Value	Units
$\Delta\mathbf{P}$	Initial position error (1σ)	$(100, 100, 100)^T$	m
$\Delta\mathbf{V}$	Initial velocity error (1σ)	$(0.3, 0.3, 0.3)^T$	m/s
$\Delta\mathbf{\Psi}$	Initial attitude error (1σ)	$(0.1, 0.1, 0.1)^T$	deg
$\mathbf{d}$	IMU drift (1σ)	$(1, 1, 1)^T$	deg/hr
$\mathbf{b}$	IMU bias (1σ)	$(1, 1, 1)^T$	mg

The assumed 1σ values of IMU errors and initial navigation errors are given in Table 2.2. Actual values of initial navigation errors and IMU errors in the statistical simulation runs are determined by drawing samples from a zero-mean normal distribution with a standard deviation σ, that is, the value of some parameter s_i is drawn according to $s_i \sim N(0, \sigma_{s_i})$.

The IMU errors are then used for contaminating the pure IMU readings, while the initial input to the strapdown module is set according to initial platform parameters and initial navigation errors (cf. Section 2.5).

2.5.2.1 Navigation Performance Using Ideal Motion Estimation

Figures 2.8-2.11 show Monte-Carlo results for a straight and level north-heading trajectory, in which the measurements based on an ideal motion estimation were injected into a Kalman filter at a 1 Hz frequency. Each figure contains 4 curves: mean (μ), mean+standard deviation ($\mu + \sigma$), and the square root of the filter covariance, defined for the i-th component in the state vector $\mathbf{X}$ as $\sqrt{P(i,i)}$, where P is the a posteriori covariance matrix. In addition, a comparison is provided to an inertial scenario ($\mu + \sigma$ inertial).

The velocity errors are presented in Figure 2.9. Velocity errors normal to the flight heading are significantly reduced relative to the inertial scenario; however, they are not nullified due to errors introduced by expressing the translation measurement in the LLLN system (cf. Appendix A). It can also be seen that these errors are constant and do not grow with time. As a consequence, position errors (Figure 2.8) normal to the flight heading are considerably reduced compared to an inertial scenario. Velocity errors and position errors along the flight heading are not reduced due to lack of observability, as analyzed in Section 2.6.

The roll angle error $\Delta\Phi$ (Figure 2.10) is partially estimated and is kept constant relative to the increasing error obtained in an inertial scenario. While the pitch and yaw angles errors $(\Delta\Theta, \Delta\Psi)$ are not estimated, the error growth is restrained relative to the inertial scenario. This is due to a precise estimation of the drift state $\mathbf{d} = (d_x, d_y, d_z)^T$

(Figure 2.11), that was obtained since ideal rotation motion estimation is used. However, when real images are used, the obtained precision of rotation motion estimation, in the current implementation, is insufficient for estimating the drift state (see also Section 2.5.2.2). The bias state, $\mathbf{b} = (b_x, b_y, b_z)^T$, is estimated in the z-direction (Figure 2.11). In general, the filter covariance is consistent with the actual 1σ errors.

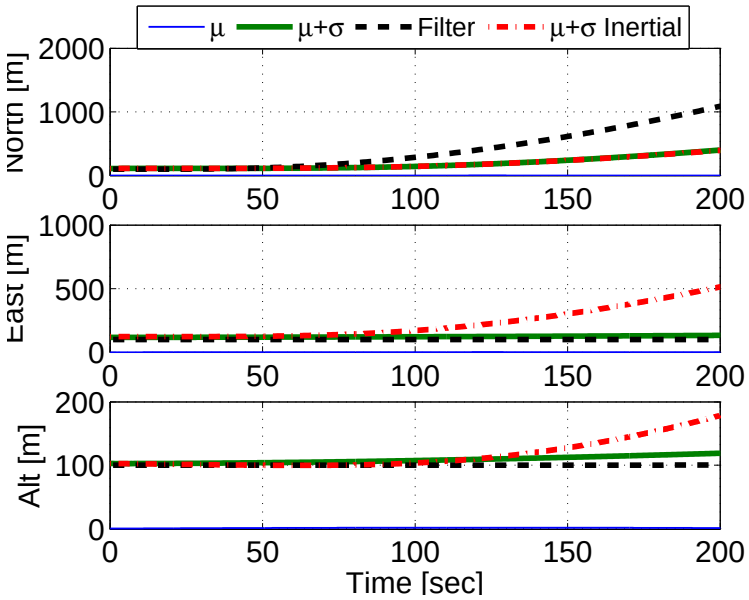


Figure 2.8: Navigation errors statistics vs. filter covariance (Monte-Carlo runs) using ideal motion estimation: Position errors. Errors normal to the flight heading are reduced, errors along the flight heading are not diminished due to lack of observability.

2.5.2.2 Contribution of the Fictitious Velocity Measurement

This section demonstrates the beneficial effect of the FV measurement on the developing navigation errors in the unobservable states. Experiment navigation-aiding results are presented based on Google Earth high-texture imagery, captured by a wide-FOV camera. Image-based motion estimations, injected to the navigation system, were computed based on the fundamental matrix model (cf. [13]), without the mosaic construction process. However, the contribution of the FV measurement is of the same nature when mosaic-based motion estimations are used.

One of the typical Google Earth images used in the experiment is shown in Figure 2.12. The trajectory in this experiment consists of a straight and level north-heading flight 1600 meters above sea level (height above ground ranges from 600 to 1300 meters) and a velocity of 150 m/s. The unobservable states for this trajectory are analyzed in Section 2.6. The same IMU errors and initial navigation errors as in Table 2.2 were assumed.

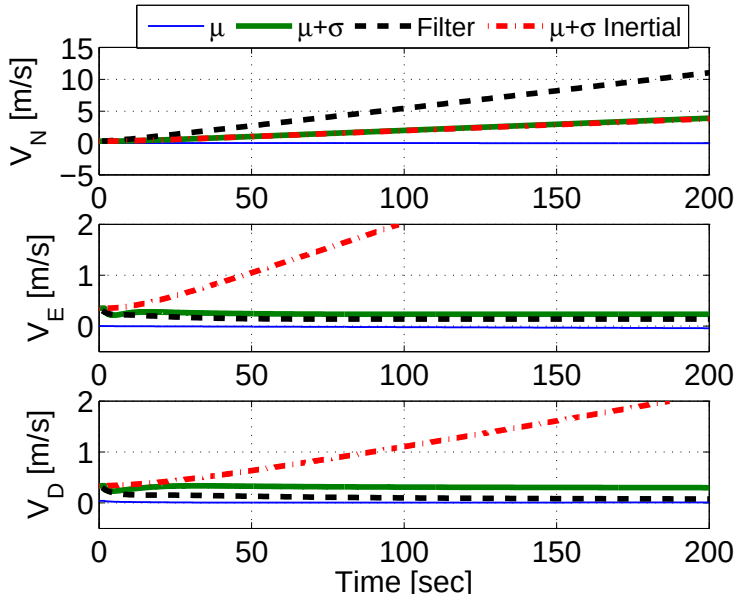


Figure 2.9: Navigation errors statistics vs. filter covariance (Monte-Carlo runs) using ideal motion estimation: Velocity errors. Errors normal to the flight heading are significantly reduced relative to the inertial scenario, and are kept constant.

Figures 2.13-2.15 provides the experimental results, comparing between the performance with the FV measurement on and off. In addition, an inertial navigation scenario is presented for reference.

Position and velocity errors (Figures 2.13 and 2.14) normal to the flight heading are significantly reduced compared to the inertial scenario (regardless of whether the fictitious velocity measurement was applied or not), as was already demonstrated in Section 2.5.2.1. The errors along the flight heading (north), which are unobservable, indeed behave as in the inertial scenario when the fictitious velocity measurement is applied, and are much degraded when this measurement is not applied. The same applies to the pitch angle error, $\Delta\Theta$, as shown in Figure 2.15. It can be concluded that, while the FV measurements prevent the erroneous updates of the unobservable states, they do not deteriorate estimation of the observable states. Therefore, subsequent results will be presented with the FV measurement on.

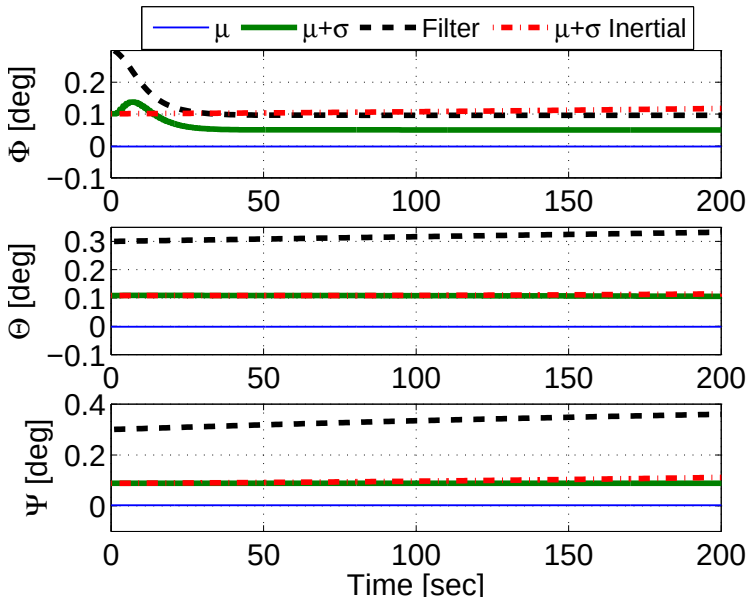


Figure 2.10: Navigation errors statistics vs. filter covariance (Monte-Carlo runs) using ideal motion estimation: Euler angle errors. Roll angle error, $\Delta\Phi$, is partially estimated and is kept constant relative to the inertial scenario; pitch and yaw angles errors $(\Delta\Theta, \Delta\Psi)$, development is arrested.

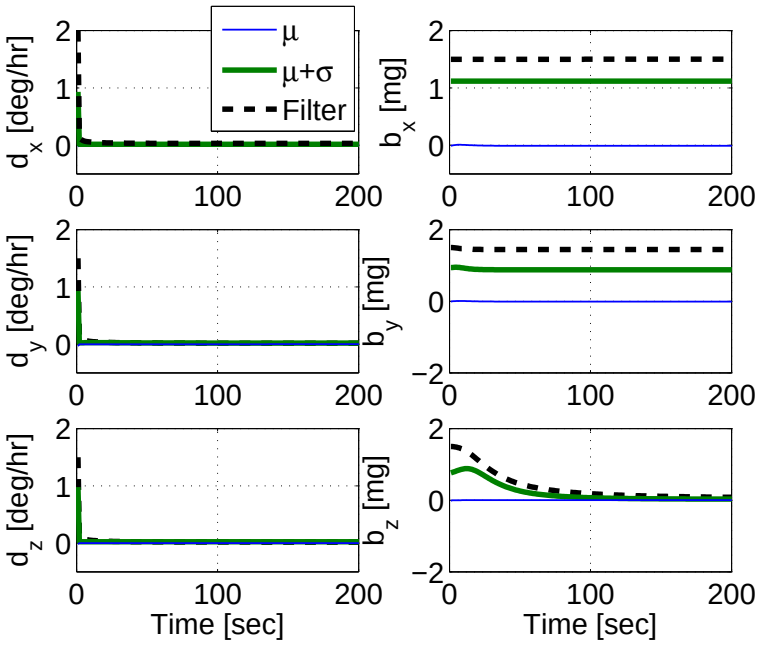


Figure 2.11: Navigation errors statistics vs. filter covariance (Monte-Carlo runs) using ideal motion estimation: Drift and Bias estimation errors. Full drift estimation due to ideal relative rotation measurement. The bias in the z direction is estimated after about 50 sec.

Figure 2.12: A Google Earth image of a high-texture scene, captured with a wide-FOV camera.

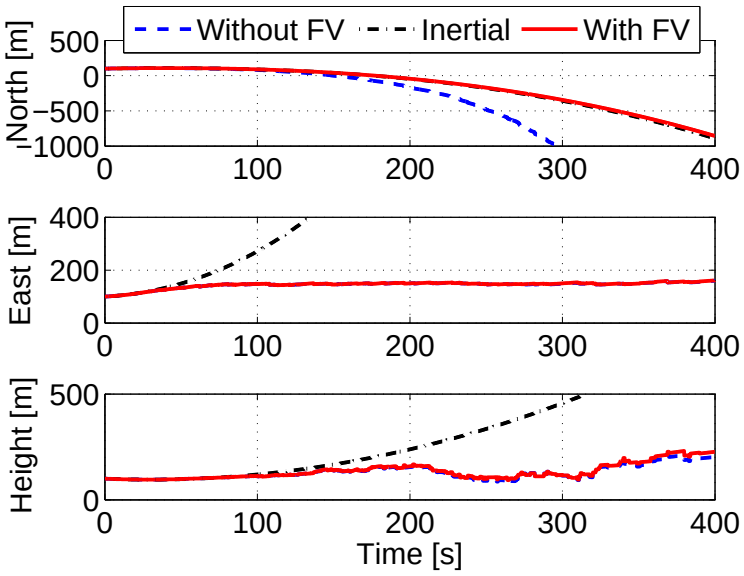


Figure 2.13: Experiment with a wide-FOV camera demonstrating the beneficial effect of the FV measurement - Position errors. Errors normal to the flight heading are considerably reduced compared to the inertial scenario; inertial behavior of the errors along the flight heading is obtained if FV measurement is applied. Significantly larger errors along the flight heading are obtained if the FV is not applied.

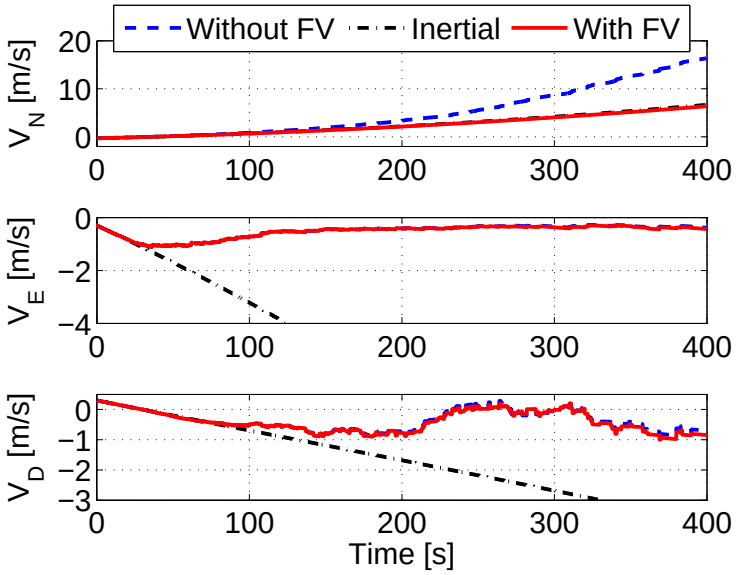


Figure 2.14: Experiment with a wide-FOV camera demonstrating the beneficial effect of the FV measurement - Velocity errors. See Figure 2.13 for details.

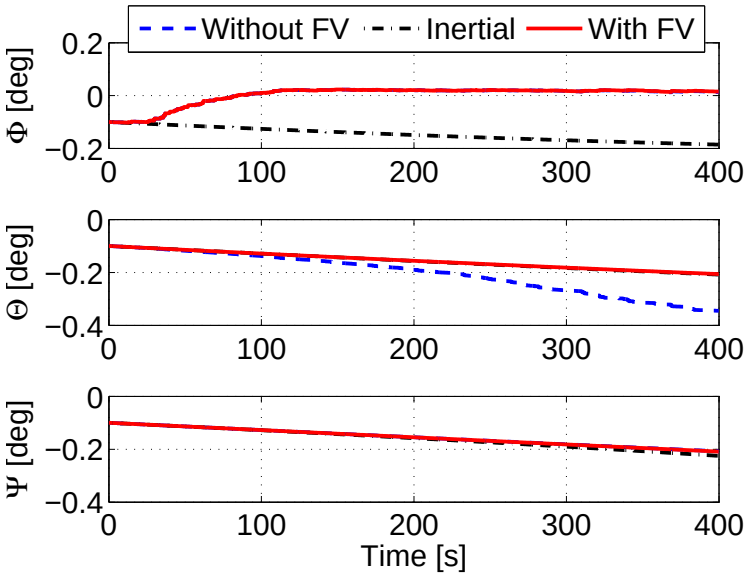


Figure 2.15: Experiment with a wide-FOV camera demonstrating the beneficial effect of the FV measurement - Euler angles errors. Inertial pitch angle error in case the FV measurement is applied, and enlarged error otherwise (due to enlarged velocity errors along the flight heading).

2.5.2.3 Navigation Performance Using Mosaic-Based Motion Estimation

This section demonstrates the superior performance of mosaic-aided navigation over vision-aided navigation that utilizes two-view motion estimation. The examined scenario consists of a narrow-FOV camera ($5^o \times 3^o$) and a low-texture scene. The platform performs a straight and level north-heading flight, as discribed in Section 2.5.

The experiment consisted of 50 seconds of inertial flight, followed by a 50 seconds of vision-aided phase, during which the mosaic- and two-view-based motion estimations were injected into the navigation system. The last phase is another inertial navigation flight segment for 50 seconds. Figures 2.16-2.19 provide the experimental results, comparing the navigation performance for the two examined methods (mosaic and two-view). In addition, the development of inertial navigation errors is given for reference.

The enhanced performance of mosaic-aided navigation can be clearly seen. During the vision-aided phase, the position and velocity errors (Figures 2.16 and 2.17) perpendicular to the flight heading are significantly reduced. The mosaic-based aiding yields better results than two-view-based aiding, due to more accurate vision-based motion estimation. It can be concluded from these graphs that the number of measurements accepted by the filter is considerably higher in case of the mosaic framework (between 60 sec and 80 sec, all the measurements in the two-view method were rejected by the filter). As for the roll angle error (Figure 2.18), although this error is smaller with the two-view method, it is expected to reach higher values if more measurements were accepted by the filter.

When examining the behavior of navigation errors in an inertial segment (after the vision-aided phase), one can notice the slow development of inertial errors when using mosaic aiding. The reason for this is the improved bias estimation compared to the estimation using the two-view method, as shown in Figure 2.19: b_z is almost exactly estimated and thus it does not contribute to the growth of inertial position and velocity errors in the down axis. The drift state was not estimated at all, because all the relative rotation measurements were rejected by the filter due to their low quality.

The relative motion measurements have another interesting effect: Although the position error state is unobservable (cf. Section 2.6), the measurements still reduce, but not nullify, the position errors (Figure 2.16), due to the developing cross-covariance terms in the covariance matrix of the state vector.

Figures 2.20 - 2.23 compare the filter covariance to the actual developed errors. As seen, the covariance is consistent. However, in the last segment of the inertial flight (after t=100 sec), the covariance development rate does not match the actual rate of the developing inertial navigation errors. After the vision-aided segment, part of the IMU error parameters are estimated by the filter (e. g. b_z) and are used to correct the actual IMU measurements. As a consequence, the actual IMU measurements injected into the navigation system are corrupted by only the residual IMU errors, resulting in a much slower development of navigation errors. One possible alternative to account for this behavior is to perform a dynamic adjustment of the filter noise covariance matrix Q as a

function of the actual covariance values of the estimated IMU states.

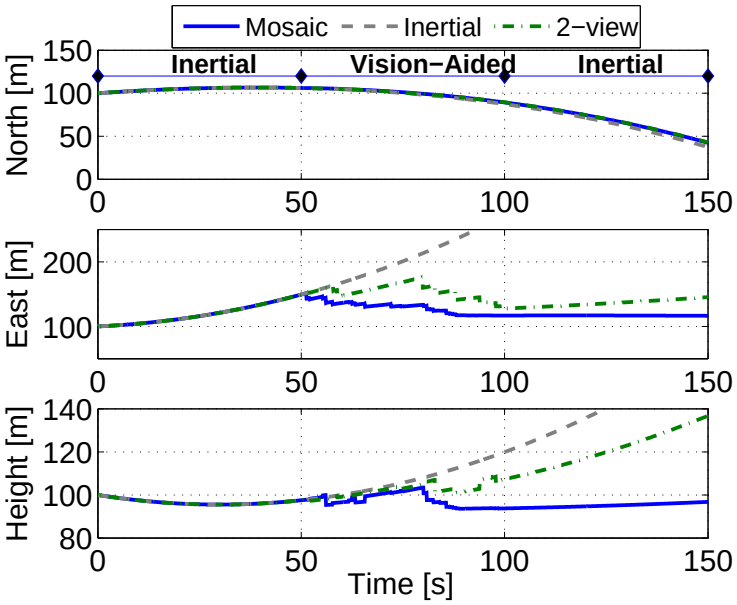


Figure 2.16: Vision-aided navigation: Mosaic aiding vs. two-view aiding: Position errors. Inertial error development in the north direction due to lack of observability. Reduced errors in the east and down directions, with a significant improvement in favor of the mosaic aiding.

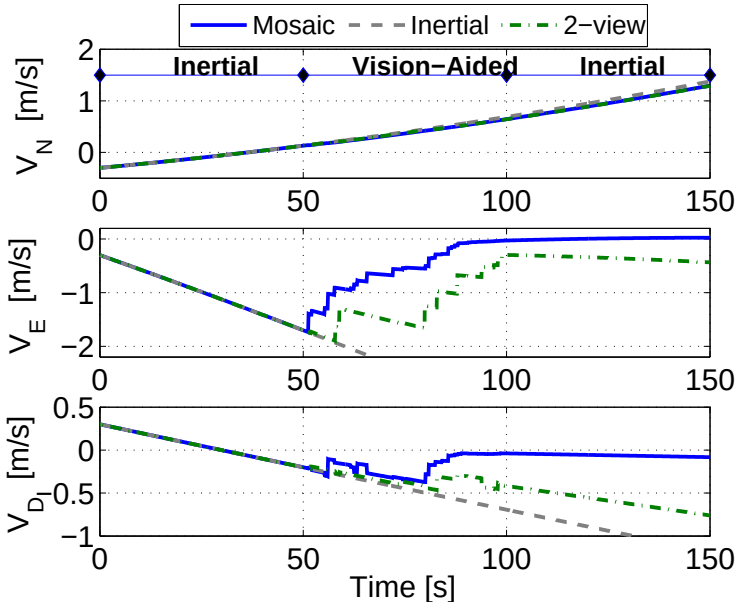


Figure 2.17: Vision-aided navigation: Mosaic aiding vs. two-view aiding: Velocity errors. See Figure 2.16 for details.

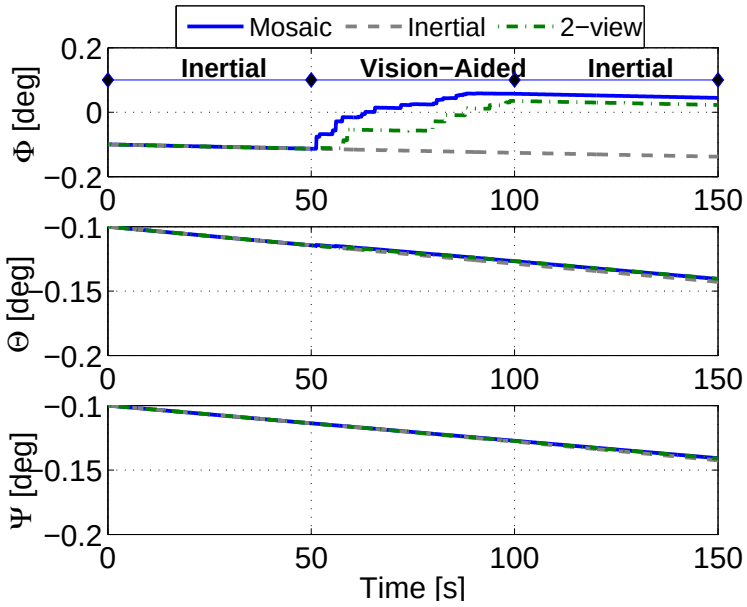


Figure 2.18: Vision-aided navigation: Mosaic aiding vs. two-view aiding: Euler angle errors. Roll angle error estimation for both motion estimation methods. Pitch and yaw angles errors are not reduced due to lack of observability.

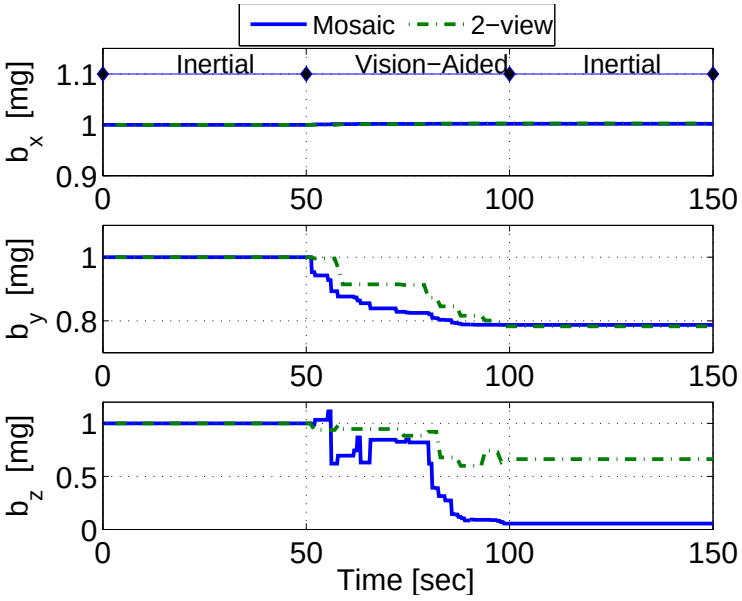


Figure 2.19: Vision-aided navigation: Mosaic aiding vs. two-view aiding: Bias estimation errors. Considerably improved b_z estimation in favor of mosaic-aided navigation.

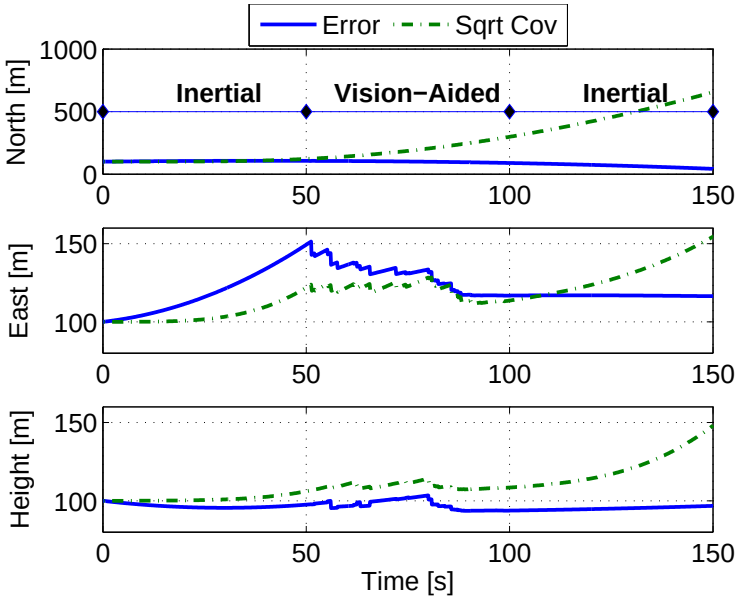


Figure 2.20: Actual navigation errors vs. filter covariance - Mosaic aiding: Position Errors.

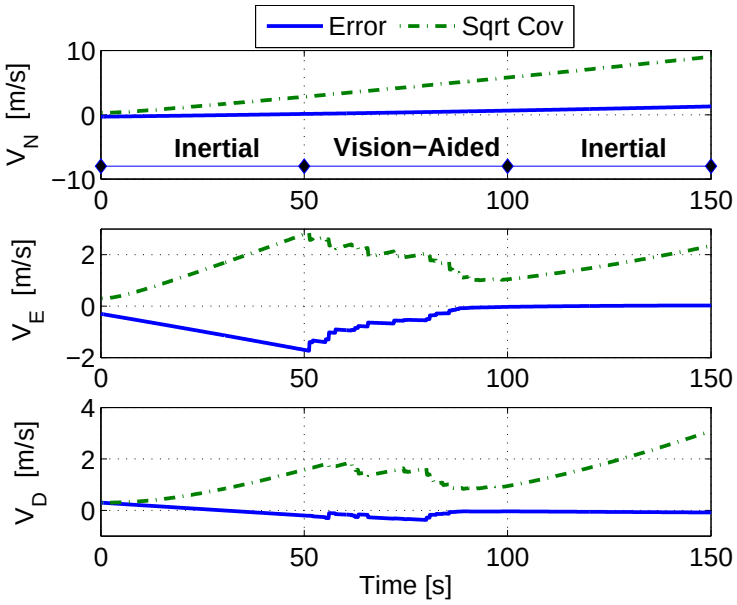


Figure 2.21: Actual navigation errors vs. filter covariance - Mosaic aiding: Velocity Errors.

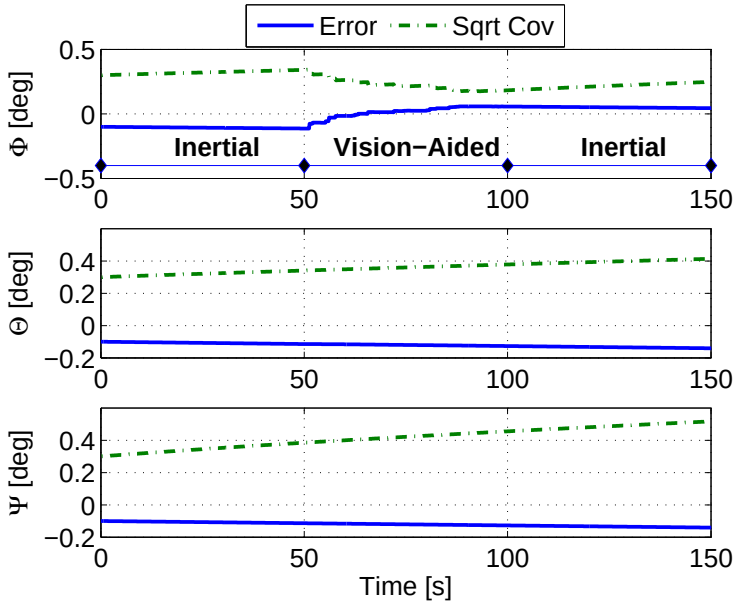


Figure 2.22: Actual navigation errors vs. filter covariance - Mosaic aiding: Euler Angles Errors.

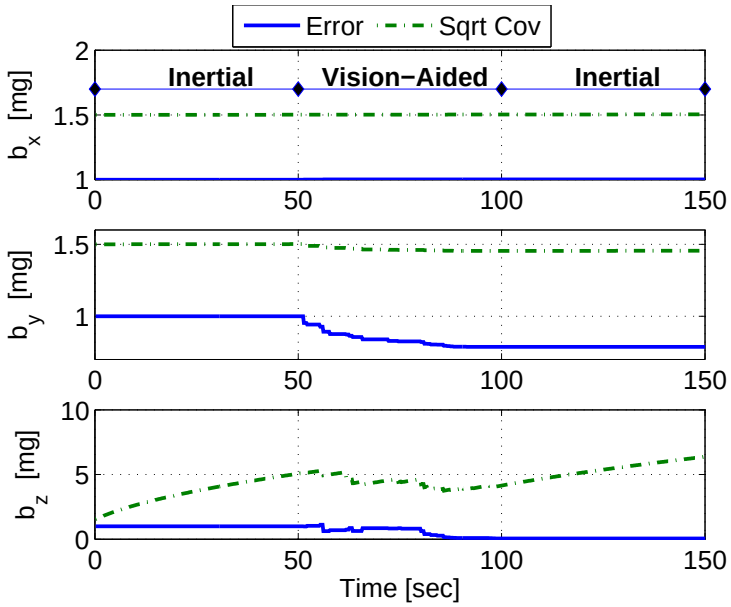


Figure 2.23: Actual navigation errors vs. filter covariance - Mosaic aiding: Bias Estimation Errors.

2.6 Observability Analysis

The observability analysis provided in this section is performed by modeling the system as a piece-wise constant system, following a variation of a method suggested in [80]. It is shown that incorporating maneuvers with sufficient duration into the platform trajectory renders the system observable except the position terms, which are always unobservable. In addition, the components of the unobservable modes are numerically investigated.

The following system is considered:

$$\mathbf{X}(k+1) = \Phi(k+1,k)\mathbf{X}(k) + \boldsymbol{\omega} \tag{2.39a}$$

$$\mathbf{Z}(k) = H(k)\mathbf{X}(k) + \mathbf{v} \tag{2.39b}$$

where Φ and $\boldsymbol{\omega}$ are the discrete system matrix and discrete process noise, respectively (cf. Section 1.3.2), H is the measurement matrix, given by Eq. (2.24), and $\mathbf{v}$ is the measurement noise defined as $\mathbf{v} = \begin{bmatrix}\mathbf{v}_{tr}^T & \mathbf{v}_{rot}^T\end{bmatrix}^T$.

Since the observability analysis does not incorporate process and measurement covariance matrices, the noise terms will be omitted, in this section, from the equations to follow. When a stochastic system is considered, the estimation errors depend on the involved noise parameters. In particular, when the system is completely observable, the lower bound of its estimation error depends only on the noise parameters [81], while in case of an unobservable system, the variance of the estimation error of an unobservable state cannot be decreased by incorporating measurements [82].

A general discrete time-invariant system

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \tag{2.40a}$$

$$\mathbf{y}(k) = C\mathbf{x}(k) \tag{2.40b}$$

is observable [82], [80] if the rank of the matrix Q_d, defined as

$$Q_d = \begin{bmatrix} C^T \mid (CA)^T \mid (CA^2)^T \mid \ldots \mid (CA^{n-1})^T \end{bmatrix}^T \tag{2.41}$$

is n, where n is the dimension of the state vector $\mathbf{x}$.

If $k = \text{rank}(Q_d) < n$, the system has $n-k$ unobservable modes. These modes can be found by computing the Observability Grammian $\mathcal{G}$ (e.g. cf. [70]):

$$\mathcal{G} = Q_d^T Q_d \tag{2.42}$$

The unobservable modes are the non-zero elements of eigenvectors belonging to zero eigenvalues of $\mathcal{G}$. Thus, one looks for all the vectors $\mathbf{a}$ for which $\mathcal{G}\mathbf{a} = \mathbf{0}$, i. e. the right null-space of the matrix $\mathcal{G}$.

However, the system given in Eq. (2.39) cannot be considered in general as time-invariant. Instead, it may be expressed as a discrete piecewise constant system [80] as

follows

$$\mathbf{X}(k+1) = \Phi_{d_j}\mathbf{X}(k) \tag{2.43a}$$
$$\mathbf{Z}_j(k) = H_j\mathbf{X}(k) \tag{2.43b}$$

where for each segment $j = 1, \ldots, r$ the matrices Φ_{d_j} and H_j are constant.

Following [80], assuming at least n measurements in each segment (cf. Figure 2.24(a)), the measurements can be expressed as a function of $\mathbf{X}(1)$:

$$\begin{aligned}
\mathbf{Z}_1(1) &= H_1\mathbf{X}(1) \\
\mathbf{Z}_1(2) &= H_1\Phi_{d_1}\mathbf{X}(1) \\
&\vdots \\
\mathbf{Z}_1(n) &= H_1\Phi_{d_1}^{n-1}\mathbf{X}(1) \\
\mathbf{Z}_2(n) &= H_2\Phi_{d_1}^{n-1}\mathbf{X}(1) \\
\mathbf{Z}_2(n+1) &= H_2\Phi_{d_2}\Phi_{d_1}^{n-1}\mathbf{X}(1) \\
&\vdots
\end{aligned}$$

which can be written as

$$\mathcal{Z} = \mathcal{Q}_d(r)\mathbf{X}(1) \tag{2.44}$$

here $\mathcal{Q}_d(r)$ is the Total Observability Matrix (TOM) [80]. Denoting by Q_{d_j} the observability matrix, defined for segment j as

$$Q_{d_j} = \left[H_j^T \mid (H_j\Phi_{d_j})^T \mid (H_j\Phi_{d_j}^2)^T \mid \ldots \mid (H_j\Phi_{d_j}^{n-1})^T\right]^T \tag{2.45}$$

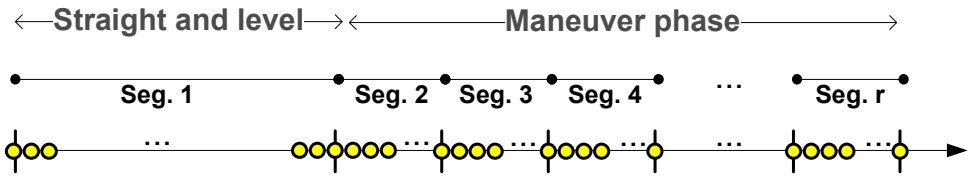
the TOM can be written as

$$\mathcal{Q}_d(r) = \begin{bmatrix} Q_{d_1} \\ Q_{d_2}\Phi_{d_1}^{n-1} \\ Q_{d_3}\Phi_{d_2}^{n-1}\Phi_{d_1}^{n-1} \\ \vdots \\ Q_{d_r}\Phi_{d_{r-1}}^{n-1}\ldots\Phi_{d_1}^{n-1} \end{bmatrix} \tag{2.46}$$

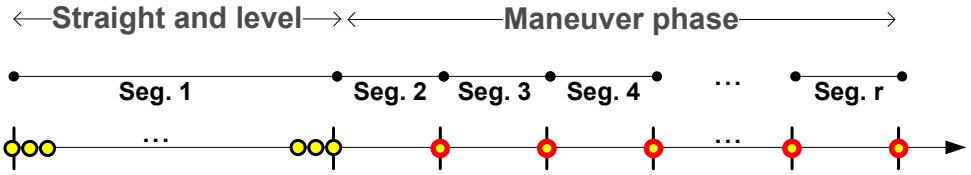
However, this derivation of the TOM relies on the assumption that each segment indeed contains at least n measurements. In practical applications, the time period between consecutive measurements, Δt, cannot be considered arbitrarily small. For example, in the proposed mosaic-aided navigation method $\Delta t = \mathcal{O}(1)$ seconds. Moreover, in case the motion estimation is performed based on the fundamental matrix, too small value of Δt will lead to a small baseline and thus yield an ill-conditioned problem [13]. On the other hand, the system matrices Φ_d and H might undergo some non-negligible changes during the time period of n measurements, violating the assumption of a time-invariant system within each segment.

Assuming at least n measurements in a segment is certainly valid for trajectory segments for which the system is time-invariant. Otherwise, it makes sense to assume less than n measurements in a segment. The worst scenario in the observability context, is to consider only *one* measurement per segment. Note that all the other scenarios can be obtained by stacking the one-measurement segments. For example, if a certain phase in the trajectory is considered to be represented by constant system parameters during a time period that allows 3 measurements to be taken, the equivalent representation of such a phase in the proposed analysis would be to take 3 one-measurement segments with identical system matrices.

It is assumed that the system is time-invariant during the first phase of the trajectory, and therefore this phase contains at least n measurements. This phase is followed by a maneuver phase (described in the sequel), which is divided into $r-1$ one-measurement segments, as illustrated in Figure 2.24(b).



(a) At least n measurements in the first segment and in each maneuver segment.



(b) At least n measurements in the first segment, one measurement in each maneuver segment.

Figure 2.24: First segment - straight and level flight. Maneuver phase is represented by $r-1$ segments. The system is considered to be constant within each such segment.

This model of segments is a variation of the model assumed in the development of the TOM [80], and thus the matrix that is used for the observability analysis should be adjusted accordingly. The first segment has at least n measurements, thus the basic observability matrix for this segment Q_{d_1}, given in Eq. (2.45), is still valid. The measurement of the second segment can be written as

$$\mathbf{Z}_2(n+1) = H_2\mathbf{X}(n+1) = H_2\Phi_{d_2}\mathbf{X}(n) = H_2\Phi_{d_2}\Phi_{d_1}^{n-1}\mathbf{X}(1) \tag{2.47}$$

Consequently, the measurements of all the maneuver segments are

$$\begin{aligned}
\mathbf{Z}_2(n+1) &= H_2\Phi_{d_2}\Phi_{d_1}^{n-1}\mathbf{X}(1) \\
\mathbf{Z}_3(n+2) &= H_3\Phi_{d_3}\Phi_{d_2}\Phi_{d_1}^{n-1}\mathbf{X}(1) \\
&\vdots \\
\mathbf{Z}_r(n+r-1) &= H_r\Phi_{d_r}\ldots\Phi_{d_2}\Phi_{d_1}^{n-1}\mathbf{X}(1)
\end{aligned}$$

which yields the following variation of the TOM:

$$\bar{\mathcal{Q}}_d(r) = \begin{bmatrix} Q_{d_1} \\ H_2\Phi_{d_2}\Phi_{d_1}^{n-1} \\ H_3\Phi_{d_3}\Phi_{d_2}\Phi_{d_1}^{n-1} \\ \vdots \\ H_r\Phi_{d_r}\ldots\Phi_{d_2}\Phi_{d_1}^{n-1} \end{bmatrix} \tag{2.48}$$

The observability analysis procedure described in the beginning of this section should be applied now to the matrix $\bar{\mathcal{Q}}_d(r)$. An analytical calculation of the nullspace of the observability Grammian $\mathcal{G} = \bar{\mathcal{Q}}_d^T(r)\bar{\mathcal{Q}}_d(r)$ poses computational difficulties after the first several segments. Motivated by the relation [83]

$$\text{rank}\,(A) = \text{rank}\left(A^TA\right) \tag{2.49}$$

instead of computing the nullspace of $\mathcal{G}$, we focused on calculating the rank of $\bar{\mathcal{Q}}_d$, from which it is straightforward to derive the number of the unobservable modes (but not their components).

The observability analysis was applied to a specific family of scenarios, comprised of a straight and level (SL) flight with a constant acceleration and of several bank-to-turn (BTT) maneuvers[7], characterized by the accelerometer measurements $f_{\text{NED}} = \begin{bmatrix} a\cdot sin(\Psi) & -a\cdot cos(\Psi) & -g \end{bmatrix}^T$, expressed in the NED system, where a is some constant parameter and g is the gravitational acceleration. During the SL flight the system is time-invariant[8], and therefore at least n measurements are assumed to be carried out in this phase. As mentioned above, the maneuver phase was divided into one-measurement segments.

The analysis was carried out assuming general parameters characterizing the considered scenario (such as the initial velocity and initial heading) and assuming ideal motion estimation. The results are given in Table 2.3, that provides the rank of $\bar{\mathcal{Q}}_d(r)$ and the number of unobservable modes, including the position state, gradually taking into account

[7] Skid-to-turn maneuvers were also analyzed, yielding very similar results in the context of observability.

[8] Since the system matrices Φ_d and H are actually the Jacobians of non-linear systems, they are functions of the navigation solution from the preceding time. These matrices, therefore, change with the accumulation of inertial navigation errors, and thus, even in the case of an SL flight, with a constant acceleration, the system can be considered to be time-invariant only for a limited amount of time.

Table 2.3: Analytical observability analysis results for a piece-wise constant system

Number of segments	rank of $\bar{\mathcal{Q}}_d$	Number of Unobservable modes
SL	7	8
SL + 1 seg	9	6
SL + 2 seg	11	4
SL + 3 seg	12	3

additional maneuver segments. As can be seen, the observability of the system improves as additional maneuver segments are processed. Thus, there are 8 unobservable modes in the case of a SL flight, while the number of unobservable modes reduces to 3 after three one-measurement maneuver segments were incorporated. These 3 modes are the position terms, which are always unobservable. This is not surprising, since the vision-based measurements provide only relative motion information and do not supply any absolute information.

The obtained unobservable modes in an SL flight scenario (5 modes, not including the 3 unobservable position states), can be compared to the 3 unobservable modes obtained in in-flight-alignment (IFA) [84]. The difference is due to the measurement model: In IFA, the measurement is the 3-axis velocity [84], while in the vision-aided navigation method, the measurement is comprised of up-to-scale translation (which is equivalent to velocity direction) and of relative rotation (cf. Eq. (2.24)). It is worth stating that in practice, due to process and measurement noise, not all the observable states can be indeed estimated. For example, as seen in Section 2.5.2.3, the drift state could not be estimated due to insufficient precision of the vision-based rotation motion estimation.

2.6.1 Numerical Investigation

As mentioned earlier, the observability analysis requires calculation of the nullspace of the observability Grammian $\mathcal{G} = \bar{\mathcal{Q}}_d^T \bar{\mathcal{Q}}_d$. However, since numerical calculations are involved, instead of calculating the nullspace $\mathcal{G}$, the implemented procedure consists of obtaining the singular value decomposition (SVD) of $\bar{\mathcal{Q}}_d$ and analyzing the eigenvectors of small singular values. This procedure is preferable to numerical calculation of $\mathcal{G}$'s nullspace, since the latter considers only zero eigenvalues while the former takes into account also infinitesimally small eigenvalues.

The SVD of $\bar{\mathcal{Q}}_d$ is given by

$$\bar{\mathcal{Q}}_d = USV^T \tag{2.50}$$

here S is a diagonal matrix of singular values of $\bar{\mathcal{Q}}_d$, which are the square root eigenvalues of $\mathcal{G}$; U and V are orthogonal matrices comprised of eigenvectors of $\mathcal{G}^T$ and $\mathcal{G}$, respectively. Thus, the unobservable modes are the elements of vectors in the matrix V that correspond to zero entries in the diagonal of S.

The specific trajectory assumed in this section is presented in Figure 2.25. As described in the previous section, the trajectory consists of a straight-and-level north-heading flight phase, followed by several BTT maneuvers. The maneuvers were divided into one-measurement segments. The duration of each such segment is one second.

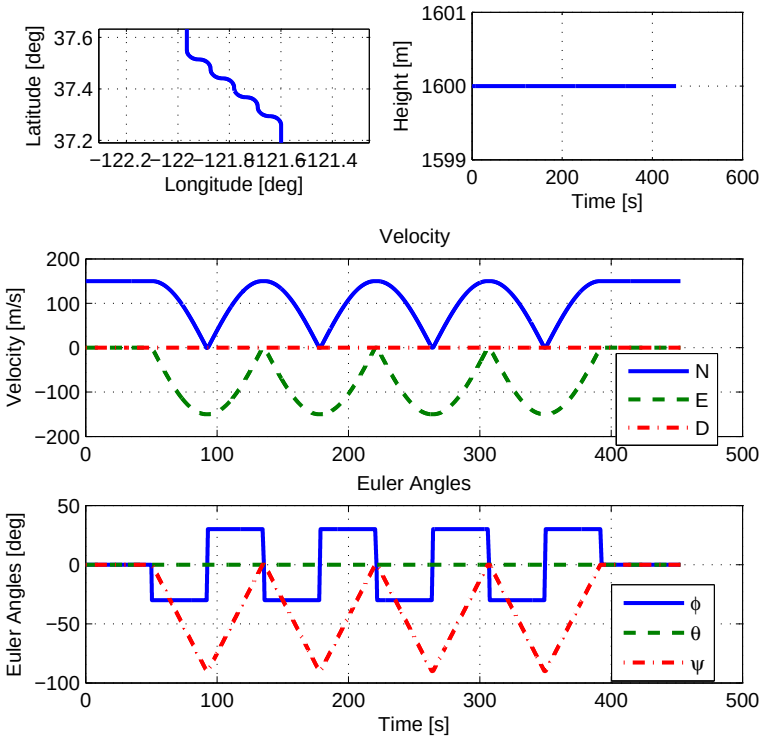


Figure 2.25: State variables of a trajectory containing a maneuver.

Figure 2.26 provides the behavior of singular values of the matrix $\bar{\mathcal{Q}}_d$ as a function of the segments being processed. The singular values, that were identified as numerical zeros and were further processed for calculating the unobservable modes, are denoted by square marks. The components of the unobservable modes are given in Figure 2.27 for the first 6 maneuver segments. The components are connected by a line for each unobservable mode.

Overall, the singular values increase when additional segments are added. Thus, some of the singular values that were considered as numeral zeros at the beginning, have significantly evolved and therefore ceased to be considered as such, reducing the number of unobservable modes. See singular values number 8-12 (Figure 2.26).

As can be seen, when only SL flight is considered (i. e. first segment only) there are 8 unobservable modes (including the position state). Thus, only $\Delta V_E, \Delta V_D, d_x, d_y, d_z$ and

b_z may be estimated. After considering several maneuver segments, only 3 unobservable modes were left, which represent the position terms. Figure 2.27 summarizes the number and the components of the unobservable modes as a function of the involved maneuver segments.

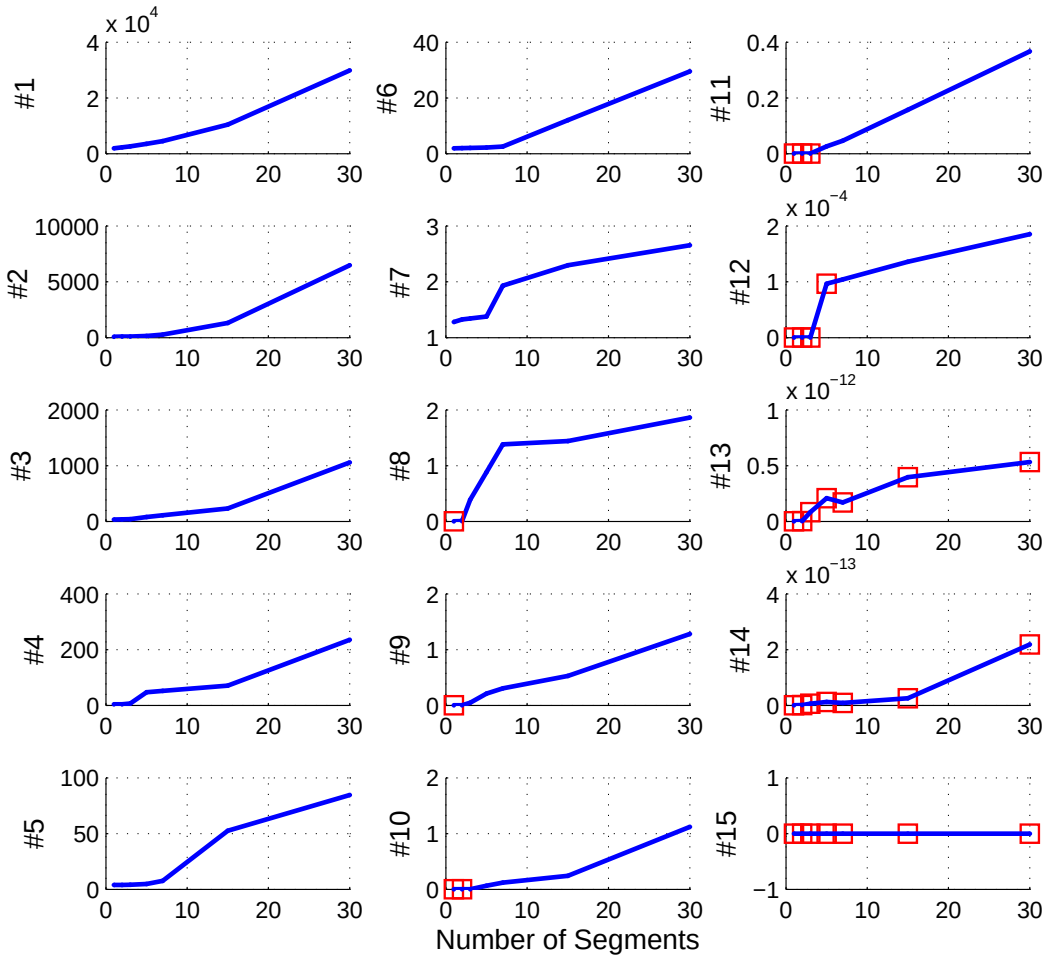


Figure 2.26: Singular Values #1 to #15 of $\bar{\mathcal{Q}}_d$. Singular values that were identified as unobservable modes are denoted with a SQUARE mark. Singular values #8,#9 and #10 were considered as unobservable modes in the first 2 maneuver segments, and are no longer considered as such afterwards due to the increase in the observability of the system. Singular values #11 and #12 ceased to be identified as unobservable modes after 2 and 4 maneuver segments were processed, respectively. Singular values #13-#15 are identified as unobservable modes throughout the whole trajectory since they represent the position error state, which is unobservable.

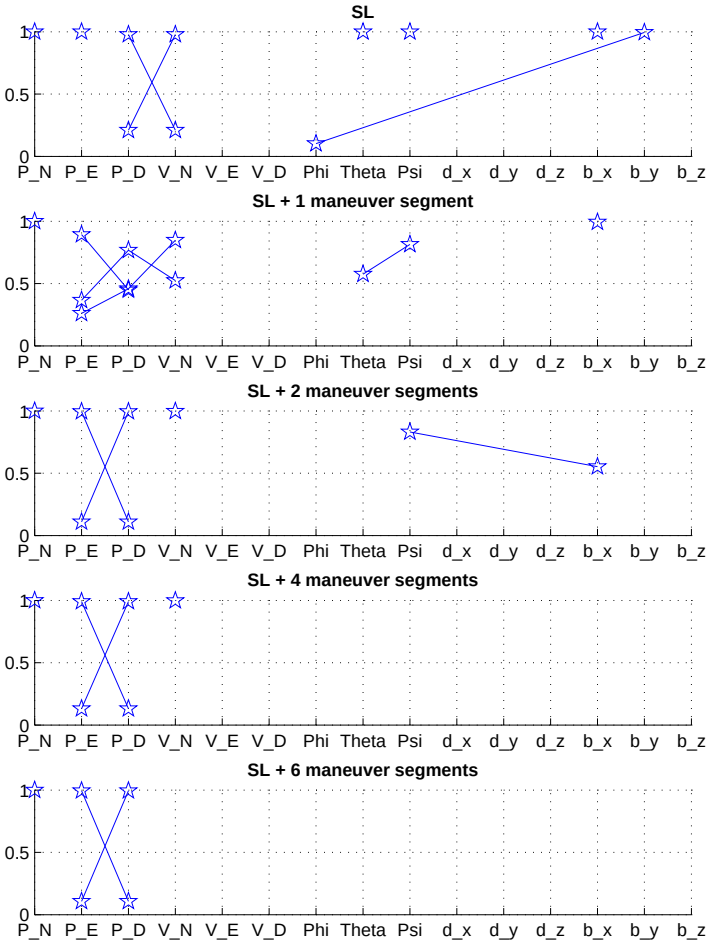


Figure 2.27: Unobservable modes behavior vs. number of maneuver segments: 8 unobservable modes when considering only straight and level flight; 6 when one maneuver segment is added; 5 when two maneuver segments are processed and 4 after considering three maneuver segments. Once at least six one-measurement maneuver segments are processed, the system becomes observable up to the position terms.

2.7 Conclusions

This chapter presented a method for vision-aided navigation for an airborne platform equipped with an inertial navigation system and a camera with a relatively small field-of-view. The camera was mounted on gimbals so that it could scan the over-flown ground regions during flight. The images captured by the camera were used for both constructing a mosaic image and performing motion estimation. Motion estimation was fused with the inertial navigation using an implicit extended Kalman filter.

The main idea of the new method was to combine camera scanning and online mosaic construction, which yielded enlarged overlapping areas. Due to the imperfectness of the mosaic construction process, features from the additional overlapping area tend to be of a lower quality compared to those from the original overlapping area. Consequently, the proposed method allowed to obtain improved-precision image-based motion estimation when the original overlapping area between the captured images contained only a small set of high-quality features, which is the case when a narrow field-of-view camera observes low-texture ground regions.

Two types of mosaic images were constructed. The first type is a small mosaic image that was used for motion estimation and constructed in real time, thereby allowing real time navigation aiding. The second type is the main mosaic image, constructed in a background process from all the captured images.

The proposed method was examined using a statistical simulation study assuming ideal motion estimations, and experiments involving realistic scenarios based on real imagery from Google Earth. These experiments included implementation of camera scanning and mosaic construction. Superior performance was demonstrated, compared to traditional two-view methods for motion estimation and navigation aiding, for challenging scenarios, such as cameras with narrow field-of-views and low-texture scenes. In particular, it was shown that estimation of position and velocity errors normal to the flight heading, as well as of the roll angle, can be significantly improved.

Since the developed method utilized a two-view technique for motion estimation, the translation motion was estimated only up to scale, which did not allow reducing some of the navigation errors, including position and velocity errors along the motion heading. These insights were also validated by an observability analysis, which modeled the system as piece wise constant. The analysis indicated that the system becomes observable, up to position errors, if sufficient maneuver segments are incorporated in the trajectory. As expected, position errors remain always unobservable.

It was noted that the method is incapable of utilizing the full potential of the available information in case of loops in the trajectory: While different algorithms can be applied in such scenarios for refining the main mosaic image in a background process, the navigation aiding phase treats loop scenarios as any other scenarios, without being able to reduce the position errors, developed during the loop sequence, in all axes. This observation motivated the development of a new method, which is described in the next chapter.

Chapter 3

Navigation Aiding Based on Three-View Geometry

Contents

In this chapter, a new method for vision-aided navigation is developed. The method utilizes constraints stemming from a general three-view geometry. A new formulation of these constraints is developed following the rank condition approach [13] [20], relating between any three images observing the same static scene and the navigation solutions present while these images were captured. These constraints include, in addition to the well-known epipolar constraints [13], a new constraint related to the three-view geometry of a general scene. The scale ambiguity, inherent to pure computer vision-based motion estimation techniques, is resolved by utilizing the navigation data attached to each image.

The developed constraints are fused with an inertial navigation system using an implicit extended Kalman filter, allowing estimation of the position vector, by reducing the position errors in all axes to the levels present while the first two images were taken. Navigation errors of other states are reduced as well, including velocity errors in all axes.

Three, not necessarily consecutive, views with a common overlapping area[1] constitute a measurement for navigation aiding. Navigation aiding in case of loops in the trajectory is handled naturally, also requiring processing only three images and therefore presenting reduced computational load compared to state-of-the-art techniques for handling loop scenarios (cf. Section 1.1.3). Following the overall architecture assumed throughout this thesis (cf. Section 1.2), the refinement of the environment representation can be performed in a background process by applying different algorithms (cf. Section 1.1.3).

The developed constraints and the well-known trifocal tensor [13] are both constituted assuming a general three-view geometry. However, while the trifocal tensor utilizes only features that are observed from all the three images, the developed constraints can be also separately applied using features that are observed in each pair of images of the given three images. It should be noted that the trifocal tensor has been suggested for camera motion estimation [21], [85], and for localization of a robot and observed landmarks while performing a planar motion [86]. However, the trifocal tensor and the three-view geometry constraints developed herein have not been proposed so far for navigation aiding, and in particular for handling loop scenarios.

Consequently, the main contributions of this chapter are: 1) A new formulation of the constraints stemming from a general static scene captured by three views; 2) application of the developed three-view constraints for navigation aiding, and in particular for handling loop scenarios, and 3) reduced computational requirements compared to other methods capable of handling loops in the trajectory.

3.1 Method Overview

A simplified diagram of the proposed method for navigation aiding is given in Figure 3.1. The vehicle is equipped with a standard inertial navigation system and a camera (which may be mounted on gimbals). The INS is comprised of an inertial measurement unit whose readings are processed by the strapdown algorithm into a navigation solution.

During motion, the camera-captured images and partial navigation data, to be defined in the sequel, are stored and maintained. When a new image is captured, it is checked whether this image has a common overlapping area with two previously stored images. One possible outcome of this step is a set of three overlapping images captured in close timing. Another possibility is a loop in the vehicle's trajectory, in which case the new image overlaps two stored images captured while the vehicle visited the region previously.

[1] The term *common overlapping area* refers to an area that is present in all the three images.

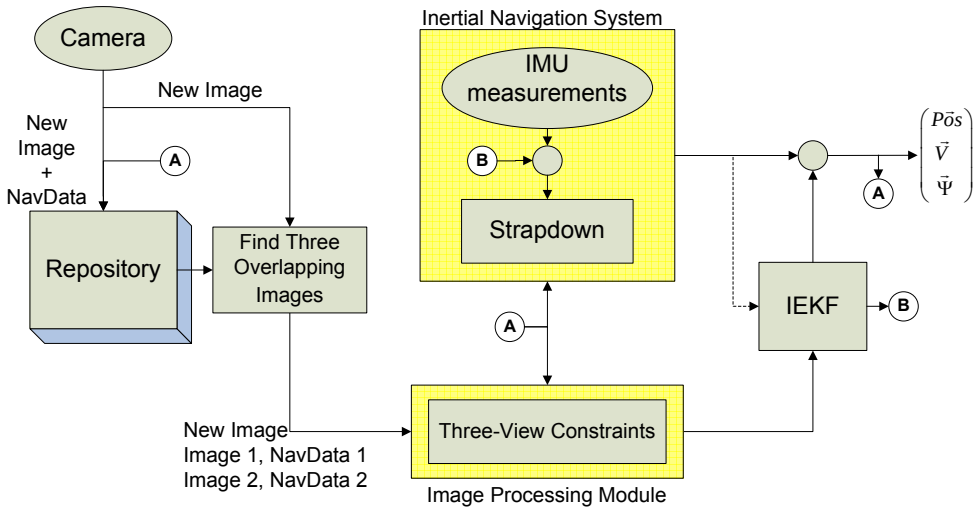


Figure 3.1: Aiding an inertial navigation system with three-view geometry constraints.

Once a set of three images containing a common overlapping area has been identified, the images and the navigation data associated to each image are used for calculating the constraints developed in Section 3.2. These constraints are then reformulated into measurements and injected into an IEKF for estimating the developed navigation error and IMU errors (cf. Section 3.3). These estimates are ultimately used for correcting the navigation solution and the IMU measurements.

While some of the images in the repository are eventually used for navigation aiding, the overall set of stored images may be used for constructing a representation of the observed environment, e. g. a mosaic. The mosaic can be just a single image constructed from the set of camera-captured images (e. g. Chapter 2), or alternatively, the mosaic can be represented by the original images accompanied by homography matrices that relate each image to a common reference frame [36]. In any case, since the navigation aiding step does not rely on the mosaic, but rather on the original images and the concomitant navigation data, the mosaic image construction can be performed in a background (low-priority) process.

Throughout this chapter, we use the coordinate systems, defined in Section 1.3.1, with the camera system redefined as follows:

- C - Camera-fixed reference frame. Its origin is set at the camera center-of-projection. Z_C points toward the FOV center, X_C points toward the right half of the FOV when viewed from the camera center-of-projection, and Y_C completes the setup to yield a Cartesian right hand system.

3.2 Three-View Geometry Constraints Development

We begin by presenting a development of constraints based on a general three-view geometry. Figure 3.2 shows the considered scenario, in which a single ground landmark p is observed in three images captured at time instances t_1, t_2 and t_3, where $t_1 < t_2 < t_3$. Denote by $\mathbf{T}_{ij}$ the camera translational motion from the ith to the jth view, with $i, j \in \{1, 2, 3\}$ and $i \neq j$. Let also $\mathbf{q}_i$ and λ_i be a line of sight vector and a scale parameter, respectively, to the ground landmark p at time t_i, such that $||\lambda_i \mathbf{q}_i||$ is the range to this landmark. In particular, if $\mathbf{q}_i$ is a unit LOS vector, then λ_i is the range to the ground landmark.

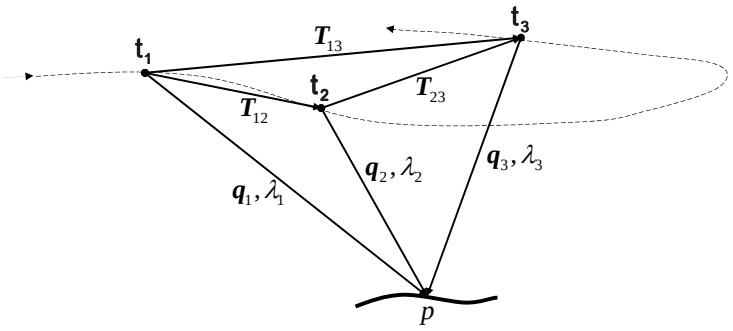


Figure 3.2: Three view geometry: a ground landmark observed in three different images.

Assuming $t_3 - t_2 > t_2 - t_1$, the translation vectors between the different views, when calculated solely based on the navigation data, will be obtained with different accuracy due to the developing inertial navigation errors: $\mathbf{T}_{12}$ contains navigation errors developed from t_1 to t_2, while $\mathbf{T}_{23}$ (and $\mathbf{T}_{13}$) is mainly affected by position errors developed from t_2 (or t_1) to t_3. Since $t_3 - t_2 > t_2 - t_1$, the accuracy of $\mathbf{T}_{23}$ is deteriorated compared to the accuracy of $\mathbf{T}_{12}$. The purpose of this section is to formulate constraints for determining $\mathbf{T}_{23}$ based on information extracted from the three images and partial navigation information (from which $\mathbf{T}_{12}$ may be calculated), thereby improving the accuracy of $\mathbf{T}_{23}$, bringing it to the accuracy levels of $\mathbf{T}_{12}$.

The position of a ground landmark p relative to the camera position at t_1, expressed in the LLLN system of t_2, can be written as:

$$\lambda_1 C_{L_2}^{C_1} \mathbf{q}_1^{C_1} = C_{L_2}^{C_1} \mathbf{T}_{12}^{C_1} + \lambda_2 C_{L_2}^{C_2} \mathbf{q}_2^{C_2} \tag{3.1a}$$

$$\lambda_1 C_{L_2}^{C_1} \mathbf{q}_1^{C_1} = C_{L_2}^{C_1} \mathbf{T}_{12}^{C_1} + C_{L_2}^{C_2} \mathbf{T}_{23}^{C_2} + \lambda_3 C_{L_2}^{C_3} \mathbf{q}_3^{C_3} \tag{3.1b}$$

where $\mathbf{q}_i^{C_i}$ is a LOS vector to the ground feature at t_i, expressed in a camera system at t_i; $C_{L_2}^{C_i}$ is a DCM transforming from the camera system at t_i to the LLLN system at t_2; and $\mathbf{T}_{ij}^{C_i}$ is the platform translation from time t_i to t_j, expressed in the camera system at t_i. Here $i, j \in \{1, 2, 3\}, i \neq j$.

Subtraction of Eq. (3.1a) from Eq. (3.1b) and some basic algebraic manipulations give

$$\mathbf{0} = \lambda_1 C_{L_2}^{C_1}\mathbf{q}_1^{C_1} - \lambda_2 C_{L_2}^{C_2}\mathbf{q}_2^{C_2} - C_{L_2}^{C_1}\mathbf{T}_{12}^{C_1} \tag{3.2a}$$

$$\mathbf{0} = \lambda_2 C_{L_2}^{C_2}\mathbf{q}_2^{C_2} - \lambda_3 C_{L_2}^{C_3}\mathbf{q}_3^{C_3} - C_{L_2}^{C_2}\mathbf{T}_{23}^{C_2} \tag{3.2b}$$

Since the scale parameters $\lambda_1, \lambda_2, \lambda_3$ are neither required nor known, we wish to form constraints on $\mathbf{T}_{23}$ without using these parameters, or in other words, avoid structure reconstruction. For this purpose, Eq. (3.2) is rewritten into the matrix form

$$\begin{bmatrix} \mathbf{q}_1 & -\mathbf{q}_2 & \mathbf{0}_{3\times1} & -\mathbf{T}_{12} \\ \mathbf{0}_{3\times1} & \mathbf{q}_2 & -\mathbf{q}_3 & -\mathbf{T}_{23} \end{bmatrix}_{6\times4} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ 1 \end{bmatrix}_{4\times1} = \mathbf{0}_{6\times1} \tag{3.3}$$

For the sake of brevity, the superscript L_2 was omitted, e. g. $\mathbf{q}_1 \equiv \mathbf{q}_1^{L_2} = C_{L_2}^{C_1}\mathbf{q}_1^{C_1}$.

Let

$$A = \begin{bmatrix} \mathbf{q}_1 & -\mathbf{q}_2 & \mathbf{0}_{3\times1} & -\mathbf{T}_{12} \\ \mathbf{0}_{3\times1} & \mathbf{q}_2 & -\mathbf{q}_3 & -\mathbf{T}_{23} \end{bmatrix} \in \mathbb{R}^{6\times4} \tag{3.4}$$

In a similar manner to Refs. [13] and [20], since all the components in $\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & 1\end{bmatrix}^T$ are nonzero, it follows that $\text{rank}(A) < 4$. The following theorem provides necessary and sufficient conditions for rank deficiency of A.

Theorem 3.2.1 $\text{rank}(A) < 4$ *if and only if all the following conditions are satisfied:*

$$\mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) = 0 \tag{3.5a}$$

$$\mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) = 0 \tag{3.5b}$$

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \tag{3.5c}$$

The proof of Theorem 3.2.1 is provided in Appendix B.

The first two constraints in Eq. (3.5) are the well-known epipolar constraints, which force the translation vectors to be co-planar with the LOS vectors. Given multiple matching features, one can determine from Eqs. (3.5a) and (3.5b) the translation vectors $\mathbf{T}_{12}$ and $\mathbf{T}_{23}$, respectively, up to scale. In general, these two scale unknowns are different. The two scales are connected through Eq. (3.5c), which relates between the magnitudes of $\mathbf{T}_{23}$ and $\mathbf{T}_{12}$. Consequently, if the magnitude of $\mathbf{T}_{12}$ is known, it is possible to calculate both the direction and the magnitude of $\mathbf{T}_{23}$, given multiple matching features. To the best of the Author's knowledge, the constraint (3.5c) has not appeared in previous publications.

Several remarks are in order. First, Eq. (3.5) also contains rotation parameters, since all the quantities are assumed to be expressed in the LLLN system at t_2. Second, structure reconstruction is not required. As shown in the sequel, this allows to maintain a constant-size state vector comprised of the vehicle's parameters only, resulting in a reduced computational load.

3.2.1 Multiple Features Formulation

In typical scenarios there is a set of matching pairs of features between the first two views, another set between the second and third view, and a set of matching triplets between all the three views, which is the intersection of the previous two sets. These sets are denoted by $\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}\}_{i=1}^{N_{12}}$, $\{\mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\}_{i=1}^{N_{23}}$ and $\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\}_{i=1}^{N_{123}}$, respectively, where N_{12}, N_{23} and N_{123} are the number of matching features in each set, and $\mathbf{q}_{j_i}^{C_j}$ is the ith LOS vector in the jth view, $j \in (1,2,3)$. Note that each LOS vector is expressed in its own camera system. These LOS vectors can be expressed in the LLLN system at t_2, as was assumed in the development leading to Eq. (3.5), using rotation matrices whose entries are taken from the navigation system. Thus, omitting again the explicit notation of the LLLN system at t_2, we have the matching sets $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}}$, $\{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}$ and $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}}$. Obviously,

$$\begin{aligned}(\mathbf{q}_1, \mathbf{q}_2) \in \{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}} \quad &\rightarrow \quad (\mathbf{q}_1, \mathbf{q}_2) \in \{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}} \\ (\mathbf{q}_2, \mathbf{q}_3) \in \{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}} \quad &\rightarrow \quad (\mathbf{q}_2, \mathbf{q}_3) \in \{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}\end{aligned}$$

The matching sets are assumed to be consistent in the following sense. Denote by $(\mathbf{q}_1^*, \mathbf{q}_2^*, \mathbf{q}_3^*)$ the jth element in $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}}$. Then, the matching pairs $(\mathbf{q}_1^*, \mathbf{q}_2^*)$ and $(\mathbf{q}_2^*, \mathbf{q}_3^*)$ appear in the matching pairs sets $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}}$ and $\{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}$, respectively, in the jth position as well.

Since the constraints in Eq. (3.5) are linear in $\mathbf{T}_{12}$ and $\mathbf{T}_{23}$, it is convenient to re-organize the equations into the following form:

$$(\mathbf{q}_1 \times \mathbf{q}_2)^T [\mathbf{q}_3]_\times \mathbf{T}_{23} = (\mathbf{q}_2 \times \mathbf{q}_3)^T [\mathbf{q}_1]_\times \mathbf{T}_{12} \tag{3.6}$$

$$(\mathbf{q}_2 \times \mathbf{q}_3)^T \mathbf{T}_{23} = 0 \tag{3.7}$$

$$(\mathbf{q}_1 \times \mathbf{q}_2)^T \mathbf{T}_{12} = 0 \tag{3.8}$$

Here $[.]_\times$ is the operator defined for some vector $\mathbf{a} = [a_1 \ a_2 \ a_3]^T$ as

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3.9}$$

Defining the vectors $\mathbf{f}, \mathbf{g}, \mathbf{u}, \mathbf{w} \in \mathbb{R}^{3\times 1}$ as

$$\mathbf{f}^T \doteq (\mathbf{q}_2 \times \mathbf{q}_3)^T \tag{3.10}$$

$$\mathbf{g}^T \doteq (\mathbf{q}_1 \times \mathbf{q}_2)^T \tag{3.11}$$

$$\mathbf{u}^T \doteq (\mathbf{q}_1 \times \mathbf{q}_2)^T [\mathbf{q}_3]_\times = \mathbf{g}^T [\mathbf{q}_3]_\times \tag{3.12}$$

$$\mathbf{w}^T \doteq (\mathbf{q}_2 \times \mathbf{q}_3)^T [\mathbf{q}_1]_\times = \mathbf{f}^T [\mathbf{q}_1]_\times \tag{3.13}$$

and considering all the matching pairs and triplets, Eqs. (3.6) - (3.8) turn into

$$\left[\mathbf{u}_i^T\right]_{1\times 3} \mathbf{T}_{23} = \left[\mathbf{w}_i^T\right]_{1\times 3} \mathbf{T}_{12} \tag{3.14}$$

$$\left[\mathbf{f}_j^T\right]_{1\times 3} \mathbf{T}_{23} = 0 \tag{3.15}$$

$$\left[\mathbf{g}_k^T\right]_{1\times 3} \mathbf{T}_{12} = 0 \tag{3.16}$$

with $i = 1 \dots N_{123}$, $j = 1 \dots N_{23}$, $k = 1 \dots N_{12}$. Stacking these equations together yields

$$\begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N \times 3} \mathbf{T}_{23} = \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N \times 3} \mathbf{T}_{12} \tag{3.17}$$

where $N \doteq N_{12} + N_{23} + N_{123}$ and

$$U = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_{N_{123}} \end{bmatrix}^T \qquad W = \begin{bmatrix} \mathbf{w}_1 & \dots & \mathbf{w}_{N_{123}} \end{bmatrix}^T \tag{3.18}$$

$$F = \begin{bmatrix} \mathbf{f}_1 & \dots & \mathbf{f}_{N_{23}} \end{bmatrix}^T \qquad G = \begin{bmatrix} \mathbf{g}_1 & \dots & \mathbf{g}_{N_{12}} \end{bmatrix}^T \tag{3.19}$$

If $\mathbf{T}_{12}$ and the rotation matrices are given (e. g. by the navigation system), the minimum number of matching features required for determining the vector $\mathbf{T}_{23}$ are a single matching pair between the second and the third views, and one matching triplet that may be utilized both in the trifocal constraint (3.6) and in the epipolar constraint (3.7). Moreover, since $\mathbf{T}_{12}$ is known with a certain level of accuracy, it is not essential to use the epipolar constraint for the first two views. Application of this constraint, however, is expected to improve the a priori accuracy of $\mathbf{T}_{12}$, and therefore reduce the estimation error of $\mathbf{T}_{23}$.

An alternative formulation of the constraints induced by three-view geometry of a general scene is described by the trifocal tensor [13]. Indeed, the application of the trifocal tensor was already suggested for estimating the camera motion [21], [85]. However, three-view geometry, and in particular the trifocal tensor and the constraints proposed herein, have not been used thus far for navigation aiding. Moreover, while the trifocal tensor approach is solely based on matching triplets, the constraints formulation presented in Eq. (3.17) allows using matching pairs as well. This is expected to improve the state estimation accuracy, since in typical applications the cardinality of the sets of matching pairs $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}}$ and $\{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}$ is much larger than the cardinality of the set of matching triplets $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}}$.

While the development of the constraints in Eq. (3.17) assumed a general ground scene, when a planar scene is under consideration, an additional constraint, expressing the fact that all the observed features are located on the same plane [20], [19], can be incorporated.

One may estimate $\mathbf{T}_{23}$ based on Eq. (3.17) using standard techniques (e. g. SVD) and then fuse $\mathbf{T}_{23}$ with the INS. However, a better alternative is to utilize the implicit nature of Eq. (3.17) using an implicit extended Kalman filter [78], as discussed in the next section.

3.3 Fusion with a Navigation System

In this section we present a technique for fusing the three-view geometry constraints with a standard navigation system, assuming three images with a common overlapping area

had been identified. The data fusion is performed using an indirect IEKF that estimates the navigation parameter errors instead of the parameters themselves. These estimated errors are then used for correcting the navigation solution computed by the navigation system (cf. Figure 3.1).

When real imagery and navigation data are considered, the existence of navigation errors and image noise renders the constraints of Eq. (3.17) inaccurate. Thus, the following residual measurement is defined:

$$\mathbf{z} \doteq \begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N\times 3} \mathbf{T}_{23} - \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N\times 3} \mathbf{T}_{12} \doteq \mathcal{A}\mathbf{T}_{23} - \mathcal{B}\mathbf{T}_{12} \tag{3.20}$$

Since $\mathbf{T}_{12} = \mathbf{Pos}(t_2) - \mathbf{Pos}(t_1)$, $\mathbf{T}_{23} = \mathbf{Pos}(t_3) - \mathbf{Pos}(t_2)$, and the matrices F, G, U, W are functions of the LOS vectors, the residual measurement $\mathbf{z}$ is a nonlinear function of the following parameters[2]:

$$\mathbf{z} = \mathbf{h}\left(\mathbf{Pos}(t_3), \mathbf{\Psi}(t_3), \mathbf{Pos}(t_2), \mathbf{\Psi}(t_2), \mathbf{Pos}(t_1), \mathbf{\Psi}(t_1), \left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}\right) \tag{3.21}$$

Here (t_3, t_2, t_1) denote the time instances in which the three overlapping images were captured, with t_3 being the current time.

We now recall the definition of the state vector, given in Eq. (1.10):

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\mathbf{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T \tag{3.22}$$

Since it is unknown a priori which three images will have a common overlapping area, and in order to maintain a constant-size state vector, each captured image should be stored and associated with the relevant navigation information. The navigation data that should be attached to each image are the platform position, attitude, gimbal angles and the filter's covariance matrix.

Linearizing $\mathbf{h}$ about $\mathbf{Pos}(t_3), \mathbf{\Psi}(t_3), \mathbf{Pos}(t_2), \mathbf{\Psi}(t_2), \mathbf{Pos}(t_1), \mathbf{\Psi}(t_1)$ and $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}$, and keeping the first order terms yields

$$\mathbf{z} \approx H_3\mathbf{X}(t_3) + H_2\mathbf{X}(t_2) + H_1\mathbf{X}(t_1) + D\mathbf{v} \tag{3.23}$$

where $H_3, H_2, H_1 \in \mathbb{R}^{N\times 15}$ are defined as

$$H_3 \doteq \nabla_{\boldsymbol{\zeta}(t_3)}\mathbf{h} \quad , \quad H_2 \doteq \nabla_{\boldsymbol{\zeta}(t_2)}\mathbf{h} \quad , \quad H_1 \doteq \nabla_{\boldsymbol{\zeta}(t_1)}\mathbf{h} \tag{3.24}$$

while $\boldsymbol{\zeta}$, defined in Eq. (1.3), is comprised of the navigation solution $\mathbf{x}$ and IMU errors parametrization $\boldsymbol{\beta}$.

[2]In Eq. (3.21), the notation $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}$ refers to the fact that LOS vectors from all the three images are used for calculating the residual measurement $\mathbf{z}$. Note that each of the matrices F, G, U, W is a function of a different set of matching points.

The terms $\mathbf{X}(t_3), \mathbf{X}(t_2)$ and $\mathbf{X}(t_1)$ in Eq. (3.23) are the navigation errors at the three time instances; in general, $\mathbf{X}(t_1), \mathbf{X}(t_2)$ and $\mathbf{X}(t_3)$ may be correlated.

Noting that we are only interested in estimating the navigation errors at the current time instant, $\mathbf{X}(t_3)$, the navigation errors at the first two time instances are considered as random parameters in the measurement equation. Therefore, since $\mathbf{X}(t_2)$ and $\mathbf{X}(t_1)$ are not estimated, the estimation error $\tilde{\mathbf{X}} \doteq \mathbf{X} - \hat{\mathbf{X}}$ in these two time instances is $\tilde{\mathbf{X}}(t_2) \equiv \mathbf{X}(t_2)$ and $\tilde{\mathbf{X}}(t_1) \equiv \mathbf{X}(t_1)$, respectively. These errors are represented by the filter covariance matrices $P(t_1), P(t_2)$, respectively, which are attached to the first two images.

The matrix D in Eq. (3.23) is the gradient of $\mathbf{h}$ with respect to the LOS vectors, i. e. $D \doteq \nabla_{\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}} \mathbf{h}$, and $\mathbf{v}$ is the image noise associated with the LOS vectors, having a covariance matrix R. Thus, the measurement noise is modeled as a combination of image noise, with the appropriate Jacobian matrix D, and the estimation errors $\tilde{\mathbf{X}}(t_2)$ and $\tilde{\mathbf{X}}(t_1)$ with the Jacobian matrices H_2 and H_1, respectively. The development of the matrices H_3, H_2, H_1, D and R is given in Appendix B (Section B.2).

The propagation step of the filter is carried out using the matrix Φ_d and the state vector $\mathbf{X} \in \mathbb{R}^{15\times 1}$, as explained in Section 2.4. The update step is executed *only* when a set of three overlapping images becomes available. In this step the current state vector, $\mathbf{X}(t_3)$, is estimated based on the LOS vectors and the first two state vectors $\mathbf{X}(t_1), \mathbf{X}(t_2)$, as explained next. This is in contrast to the SLAM approach, in which both the propagation and update steps of the filter are performed on a state vector that constantly increases in size.

The Kalman gain matrix is given by

$$\begin{aligned} K &= P_{\mathbf{X}(t_3)\mathbf{z}(t_3,t_2,t_1)} P^{-1}_{\mathbf{z}(t_3,t_2,t_1)} = E[\tilde{\mathbf{X}}^{-}\tilde{\mathbf{z}}^T]E[\tilde{\mathbf{z}}\tilde{\mathbf{z}}^T]^{-1} = \\ &= E[(\mathbf{X} - \hat{\mathbf{X}}^{-})(\mathbf{z} - \hat{\mathbf{z}})^T]E[(\mathbf{z} - \hat{\mathbf{z}})(\mathbf{z} - \hat{\mathbf{z}})^T]^{-1} \end{aligned} \tag{3.25}$$

where the explicit time notations were omitted for conciseness.

Since $\hat{\mathbf{z}} = H_3\hat{\mathbf{X}}^{-}(t_3)$

$$\tilde{\mathbf{z}} = \mathbf{z} - \hat{\mathbf{z}} = H_3\tilde{\mathbf{X}}^{-}(t_3) + H_2\tilde{\mathbf{X}}(t_2) + H_1\tilde{\mathbf{X}}(t_1) + D\mathbf{v} \tag{3.26}$$

Hence

$$P_{\mathbf{X}(t_3)\mathbf{z}(t_3,t_2,t_1)} = P_3^{-}H_3^T + P_{32}^{-}H_2^T + P_{31}^{-}H_1^T \tag{3.27}$$

$$P_{\mathbf{z}(t_3,t_2,t_1)} = H_3P_3^{-}H_3^T + \begin{bmatrix} H_2 & H_1 \end{bmatrix} \begin{bmatrix} P_2^{-} & P_{21} \\ P_{21}^T & P_1^{-} \end{bmatrix} \begin{bmatrix} H_2 & H_1 \end{bmatrix}^T + DRD^T \tag{3.28}$$

where $P_i = E[\tilde{\mathbf{X}}_i\tilde{\mathbf{X}}_i^T]$ and $P_{ij} = E[\tilde{\mathbf{X}}_i\tilde{\mathbf{X}}_j^T]$.

As the measurement noise, $H_2\mathbf{X}(t_2) + H_1\mathbf{X}(t_1) + D\mathbf{v}$, is statistically dependent with the state vector to be estimated, $\mathbf{X}(t_3)$, the basic assumption of the Kalman filter is contradicted. Eqs. (3.27) and (3.28) are an ad-hoc approach for taking into consideration

this dependence within the Kalman filter framework, that has given good results. Note that if all the three state vectors, $\mathbf{X}(t_3), \mathbf{X}(t_2)$ and $\mathbf{X}(t_1)$, were to be estimated, the measurement noise in Eq. (3.23) would be $D\mathbf{v}$, which is still statistically dependent with the state vectors. However, this dependence would only be due to the Jacobian D, as modeled by a standard IEKF formulation [78], [4]. Explicit equations in such case are given, in the context of cooperative navigation, in Chapter 5 (Section 5.3.1).

Referring to Eqs. (3.27) and (3.28), while the matrices P_3^-, P_2^- and P_1^- are known, the cross-correlation matrices P_{32}^-, P_{31}^- and P_{21} are unknown, and therefore need to be calculated. However, since $\mathbf{X}(t_2)$ and $\mathbf{X}(t_1)$ are stored *outside* the filter, these terms cannot be calculated without additional information or assumptions. In Chapter 4, a method is developed for calculating the cross-covariance terms assuming information from all past navigation updates is stored.

Alternatively, this issue can be handled as follows. Inertial navigation between t_1 and t_2 is assumed. Denoting by $\Phi(t_2, t_1)$ the transition matrix between $\mathbf{X}(t_1)$ and $\mathbf{X}(t_2)$, the term P_{21} may be calculated as

$$P_{21} = E[\tilde{\mathbf{X}}(t_2)\tilde{\mathbf{X}}^T(t_1)] = \Phi(t_2, t_1)P_1 \tag{3.29}$$

The other two cross-correlation terms, $P_{32}^- = E[\tilde{\mathbf{X}}^-(t_3)\tilde{\mathbf{X}}^T(t_2)]$ and $P_{31}^- = E[\tilde{\mathbf{X}}^-(t_3)\tilde{\mathbf{X}}^T(t_1)]$, may be neglected if $t_3 \gg t_2$ (e. g. loops), or when the first two images and their associated navigation data have been received from an external source (e. g. some other vehicle).

Several approaches exist for handling all the other cases in which $t_3 - t_2$ is not considerably large. One possible approach is to keep a limited history of the platform navigation parameters by incorporating these parameters into the state vector each time a new image is captured within a certain sliding window [17]. This approach is capable of handling scenarios in which all the three images are captured within the assumed sliding window. Another alternative would be to develop a bound on $t_3 - t_2$ under which the cross-correlation terms P_{32}^- and P_{31}^- can be considered negligible, and select sets of overlapping images accordingly. These two approaches may also be jointly applied. Covariance intersection (CI) [64], [66] could also be potentially used to deal with the cross-correlation terms. However, CI is incapable of handling cases in which the measurement matrix contains only a partial representation of the state vector [66], [62], which is the situation in the present case.

In this chapter, it is assumed that the current navigation parameters are *not* correlated with the navigation parameters that are associated with the first two images, i. e. $P_{32}^- = 0$ and $P_{31}^- = 0$.

In case the above assumptions regarding P_{12}^-, P_{31}^- and P_{32}^- are not satisfied, these terms can be explicitly calculated using the method developed in Chapter 4.

After the residual measurement and the gain matrix have been computed using Eqs. (3.21) and (3.25), respectively, the state vector and the covariance matrix can be updated based on the standard equations of the IEKF.

3.3.1 Computational Requirements

A single filter update step, given three images with a common overlapping area, involves computation of the matrices $\mathcal{A}, \mathcal{B}$ and the Jacobian matrices H_3, H_2, H_1 and D. These calculations are linear in N, the overall size of the matching sets $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{123}}$, $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}\right\}_{i=1}^{N_{12}}$ and $\left\{\mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{23}}$. Noting that the state vector is constant in size, the most computationally expensive operation in the filter update step is the inversion of an $N \times N$ matrix required for the calculation of the gain matrix.

The computational load of the proposed method does not change significantly over time (depending on the variation of N), regardless of the scenarios in which the algorithm is applied to (including loop scenarios). Moreover, if the computational capability is limited, it is possible to utilize only part of the available matching pairs and triplets (cf. Section 3.2.1), or eliminate the epipolar constraint for the first two views, thus reducing the computational load even further.

3.3.2 Extensions

It is straightforward to extend the developed method for handling more than three overlapping images, which may improve robustness to noise. In the general case, assume k given images, such that each three neighboring images are overlapping (a common overlapping area for all the k images is not required). Assume also that all these images are associated with the required navigation data. In the spirit of Eq. (3.5), we write an epipolar constraint for each pair of consecutive images, and a constraint for relating the magnitudes of the translation vectors (similar to Eq. (3.5c)) for each three adjacent overlapping images. Next, the residual measurement $\mathbf{z}$ is redefined and the calculations of the required Jacobian matrices in the IEKF formulation are repeated.

For example, consider the case of four images captured at time instances $t_1, \ldots, t_4$, with t_4 being the current time, and assume existence of common overlapping areas for the first three images and for the last three images. One possible formulation of the constraints is

$$
\begin{aligned}
(\mathbf{q}_1 \times \mathbf{q}_2)^T [\mathbf{q}_3]_\times \mathbf{T}_{23} &= (\mathbf{q}_2 \times \mathbf{q}_3)^T [\mathbf{q}_1]_\times \mathbf{T}_{12} && (3.30)\\
(\mathbf{q}_2 \times \mathbf{q}_3)^T \mathbf{T}_{23} &= 0 && (3.31)\\
(\mathbf{q}_1 \times \mathbf{q}_2)^T \mathbf{T}_{12} &= 0 && (3.32)\\
(\mathbf{q}_2 \times \mathbf{q}_3)^T [\mathbf{q}_4]_\times \mathbf{T}_{34} &= (\mathbf{q}_3 \times \mathbf{q}_4)^T [\mathbf{q}_2]_\times \mathbf{T}_{23} && (3.33)\\
(\mathbf{q}_3 \times \mathbf{q}_4)^T \mathbf{T}_{34} &= 0 && (3.34)
\end{aligned}
$$

Considering all the available matches and following the same procedure as in Section 3.3, the residual measurement $\mathbf{z}$ will assume the form

$$\mathbf{z} = \mathcal{J}\mathbf{T}_{\mathbf{34}} - \mathcal{V}\mathbf{T}_{\mathbf{23}} - \mathcal{L}\mathbf{T}_{\mathbf{12}}$$

where the matrices $\mathcal{J}, \mathcal{V}, \mathcal{L}$ are constructed based on Eqs. (3.30)-(3.34).

Since $\mathbf{T}_{12}, \mathbf{T}_{23}$ and all the rotation matrices that implicitly appear in Eqs. (3.30)-(3.34) can be calculated based on the navigation data associated with the images, the residual measurement $\mathbf{z}$ is given by

$$\mathbf{z} = \mathbf{h}\left(\mathbf{Pos}(t_4), \mathbf{\Psi}(t_4), \mathbf{Pos}(t_3), \mathbf{\Psi}(t_3), \mathbf{Pos}(t_2), \mathbf{\Psi}(t_2), \mathbf{Pos}(t_1), \mathbf{\Psi}(t_1), \left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}, \mathbf{q}_{4_i}^{C_4}\right\}\right)$$

in which $\mathbf{Pos}(t_4), \mathbf{\Psi}(t_4)$ are part of the *current* navigation solution. This measurement may be utilized for estimating the developed navigation errors in the same manner as discussed in Section 3.3. The involved computational requirements will increase only in the update step, according to the total size of the matching sets. The propagation step of the filter remains the same.

3.4 Simulation and Experimental Results

This section presents statistical results obtained from simulated navigation data and synthetic imagery data, as well as experimental results utilizing real navigation and imagery data.

3.4.1 Implementation Details

3.4.1.1 Navigation Simulation

The navigation simulation is described in Section 2.5. Once a set of three images with a common overlapping area is available, the developed algorithm is executed: the state vector is estimated based on the developed algorithm using IEKF, which is then used for updating the navigation solution (cf. Figure 3.1). The estimated bias and drift are used for correcting the IMU measurements.

3.4.1.2 Image Processing Module

Given three images with a common overlapping area, the image processing phase includes features extraction from each image using the SIFT algorithm [76] and computation of sets of matching pairs between the first two images, $\{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^{N_{12}}$, and between the last two images, $\{\mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{23}}$, where $\mathbf{x}^i = (x^i, y^i)^T$ are the image coordinates of the ith feature. This computation proceeds as follows. First, the features are matched based on their descriptor vectors (that were computed as part of the SIFT algorithm), yielding the sets $\{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^{\tilde{N}_{12}}, \{\mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{\tilde{N}_{23}}$. Since this step occasionally produces false matches (outliers), the RANSAC algorithm [77] is applied over the fundamental matrix [13] model in order to reject the existing false matches, thus obtaining the refined sets $\{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^{N_{12}}$ and $\{\mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{23}}$. The fundamental matrices are not used in further computations.

The next step is to use these two sets for calculating matching triplet features, i. e. matching features in the three given images. This step is performed by matching all $\mathbf{x}_1 \in \{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^{N_{12}}$ with all $\mathbf{x}_3 \in \{\mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{23}}$, yielding a set of matching triplets $\{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$. The matching process includes the same steps as described above.

When using synthetic imagery data, a set of points in the real-world are randomly drawn. Then, taking into account the camera motion, known from the true vehicle trajectory, and assuming specific camera calibration parameters, the image coordinates of the observed real-world points are calculated using a pinhole projection [13] at the appropriate time instances. See, for example, Ref. [4] for further details. Consequently, a list of features for each time instant of the three time instances, which are manually specified, is obtained: $\{\mathbf{x}_1^i\}$, $\{\mathbf{x}_2^i\}$ and $\{\mathbf{x}_3^i\}$. The mapping between these three sets is known, since these sets were calculated using the pinhole projection based on the same real-world points. Thus, in order to find the matching sets $\{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$, $\{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^{N_{12}}$ and $\{\mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{23}}$ it is only required to check which features are within the camera FOV of the appropriate views.

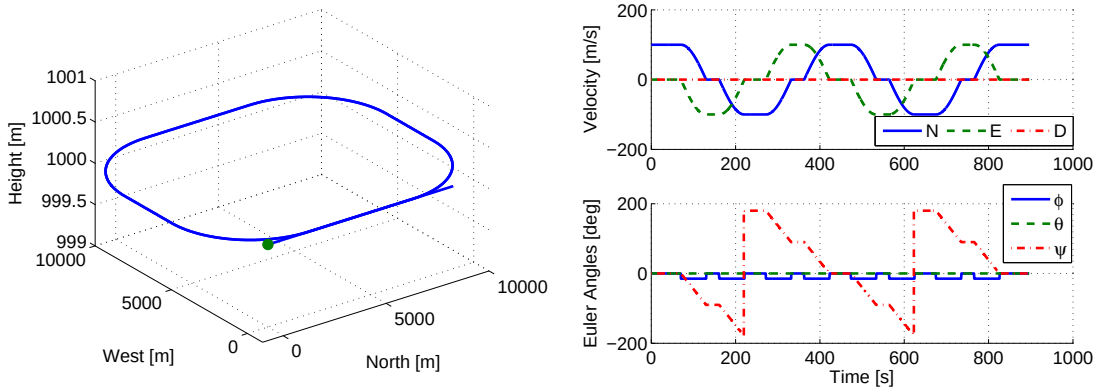
Finally, the calculated sets of matching features are transformed into sets of matching LOS vectors. A LOS vector, expressed in the camera system for some feature $\mathbf{x} = (x, y)^T$, is calculated as $\mathbf{q}^C = (x, y, f)^T$, where f is the camera focal length. As a result, three matching LOS sets are obtained: $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{123}}$, $\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}\right\}_{i=1}^{N_{12}}$ and $\left\{\mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}_{i=1}^{N_{23}}$. When handling real imagery, the camera focal length, as well as other camera parameters, are found during the camera calibration process. In addition, a radial distortion correction [13] was applied to camera-captured images, or alternatively, to the extracted feature coordinates.

Table 3.1: Initial Navigation Errors and IMU Errors

Parameter	Description	Value	Units
$\Delta\mathbf{P}$	Initial position error (1σ)	$(100, 100, 100)^T$	m
$\Delta\mathbf{V}$	Initial velocity error (1σ)	$(0.3, 0.3, 0.3)^T$	m/s
$\Delta\mathbf{\Psi}$	Initial attitude error (1σ)	$(0.1, 0.1, 0.1)^T$	deg
$\mathbf{d}$	IMU drift (1σ)	$(10, 10, 10)^T$	deg/hr
$\mathbf{b}$	IMU bias (1σ)	$(10, 10, 10)^T$	mg

3.4.2 Statistical Results based on Simulated Navigation and Synthetic Imagery

In this section, we present statistical results obtained by applying the developed algorithm to a trajectory containing a loop based on a simulated navigation system and synthetic imagery data. The assumed initial navigation errors and IMU errors are summarized in Table 3.1. The synthetic imagery data was obtained by assuming a $20^0 \times 30^0$ camera FOV, focal length of 1570 pixels, and image noise of 1 pixel. The assumed trajectory, shown in Figure 3.3(a), includes a loop that is repeated twice (see also Figure 3.3(b)).



(a) Position. The filled circle indicates the initial position.

(b) Velocity and Euler angles.

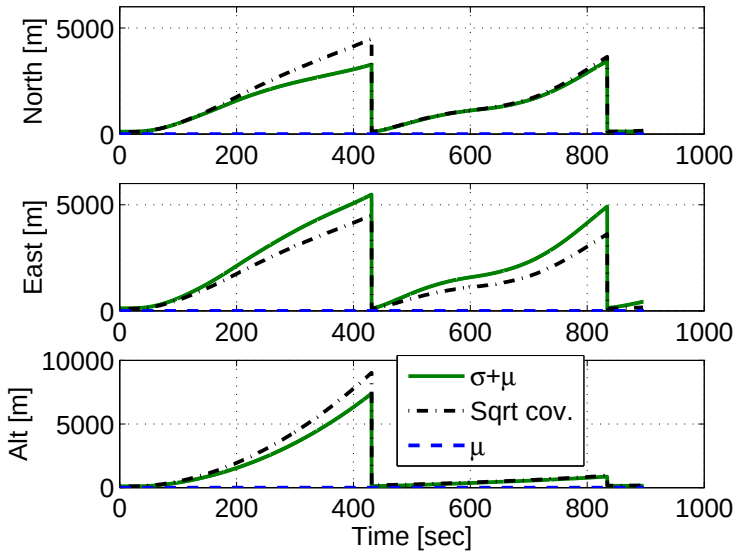
Figure 3.3: Trajectory used in the statistical study. The vehicle performs the loop twice.

The three-view navigation-aiding algorithm was applied twice, at $t = 427$ seconds and at $t = 830$ seconds; each time a specific point along the trajectory was revisited. The true translation vectors are $\mathbf{T}_{12}^L = [100 \quad 0 \quad 0]^T$ and $\mathbf{T}_{23}^L = [500 \quad 0 \quad 0]^T$. No other updates of the navigation system were performed, i. e. inertial navigation was applied elsewhere.

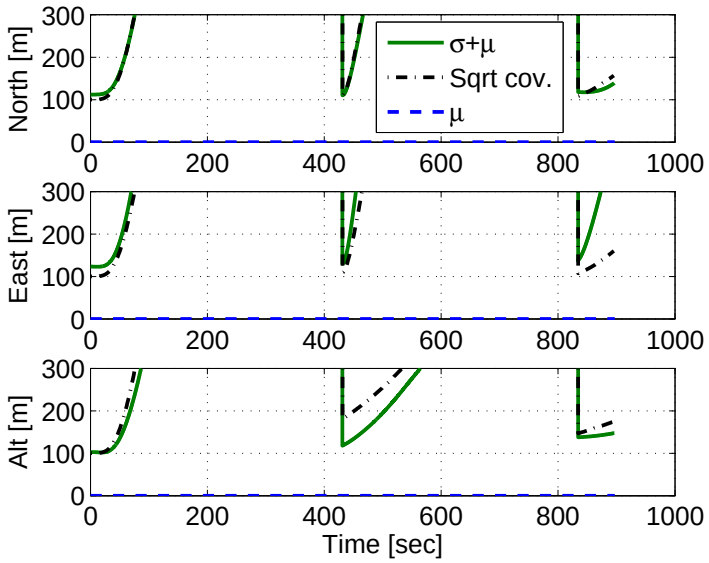
Figures 3.4-3.5 provides the Monte-Carlo results (100 runs). As seen, with the help of the three-view update, the position error (which has grown to several kilometers because

of the inertial navigation phase) is reset in *all* axes to the levels of errors at t_1 and t_2 (see Figure 3.4(b)). The velocity error is also considerably reduced in all axes as a result of the algorithm activation, while the accelerometer bias is estimated mainly in the z axis (cf. Figure 3.5(b)).

Assuming at least three matching triplets of features exist, the proposed method can be applied without using the epipolar constraints, utilizing only the constraint relating the magnitudes of translation vectors (Eq. (3.14)). In this case the accuracy of the method will degrade, mainly in a direction normal to the motion heading, as shown in Figure 3.6. The position error in the north direction, which is the motion heading at the time of the algorithm activation, is roughly the same as in the case where all the constraints in Eq. (3.17) are applied. However, in the east direction the accuracy of the position state is considerably degraded, with an error of around 900 meters, compared to an error of about 100 meters (Figure 3.4(b)), which is the initial position error (cf. Table 3.1). Observe also that although the error in the down direction has not significantly changed, the filter covariance is no longer consistent (the same filter tuning was used in both cases). The absolute reduction of position and velocity errors in all axes is not possible when applying two-view based techniques for navigation aiding, since the position and velocity along the motion direction are unobservable (cf. Section 2.6). In practical applications each of the two approaches may be applied, depending on the number of available overlapping images. Whenever a set of three images with a common overlapping area becomes available, the proposed method will reduce the navigation errors that two-view navigation aiding methods were unable to estimate (e. g. errors along motion heading) in accordance with the quality of navigation data attached to the first two images in the set.

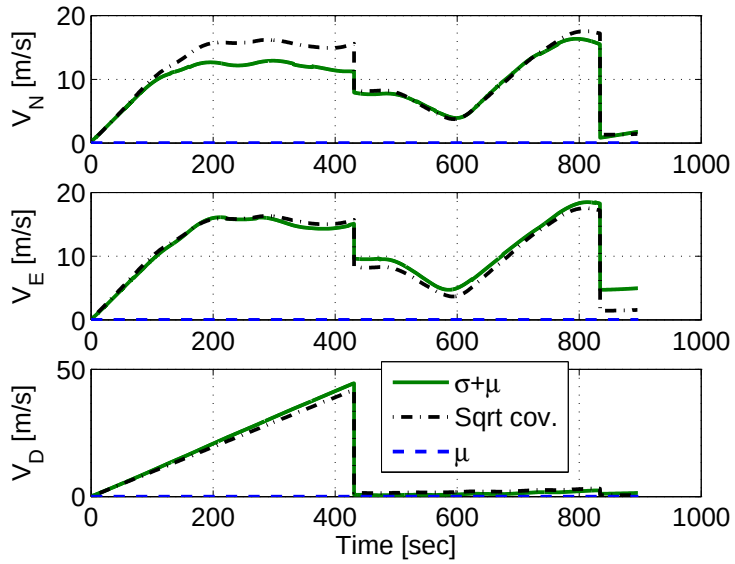


(a) Position errors.

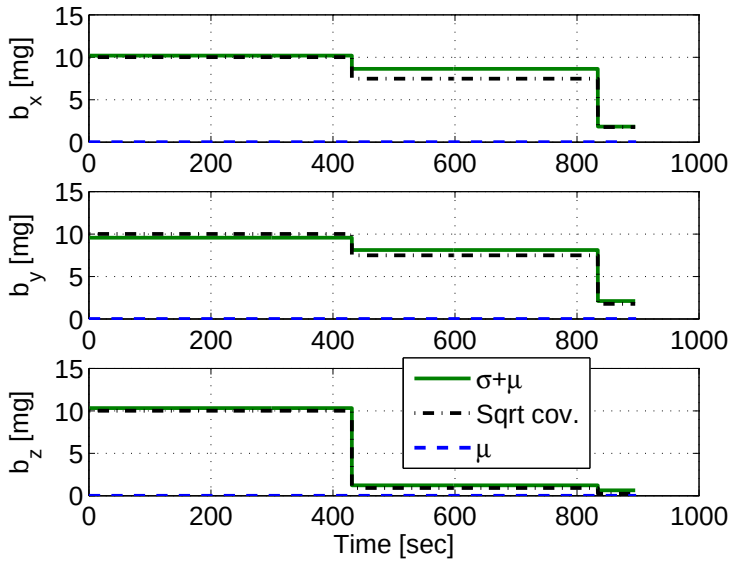


(b) Position errors - zoom.

Figure 3.4: Monte-Carlo results of the three-view navigation-aiding algorithm based on navigation simulation and synthetic imagery data.

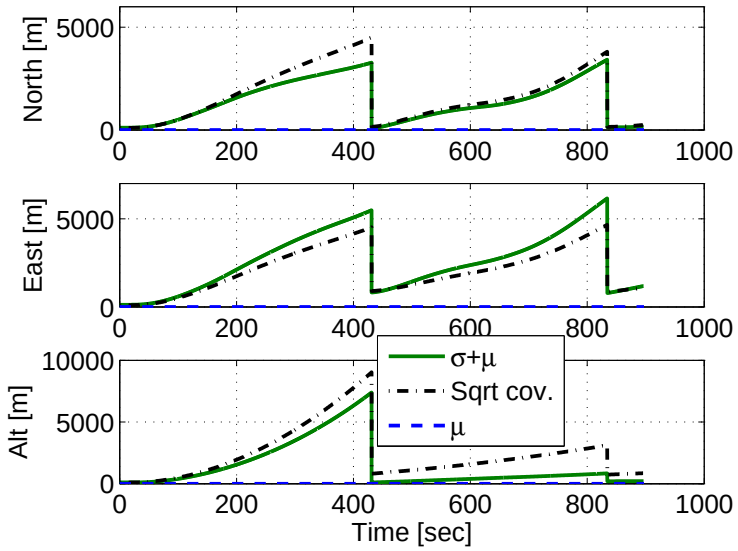


(a) Velocity errors.

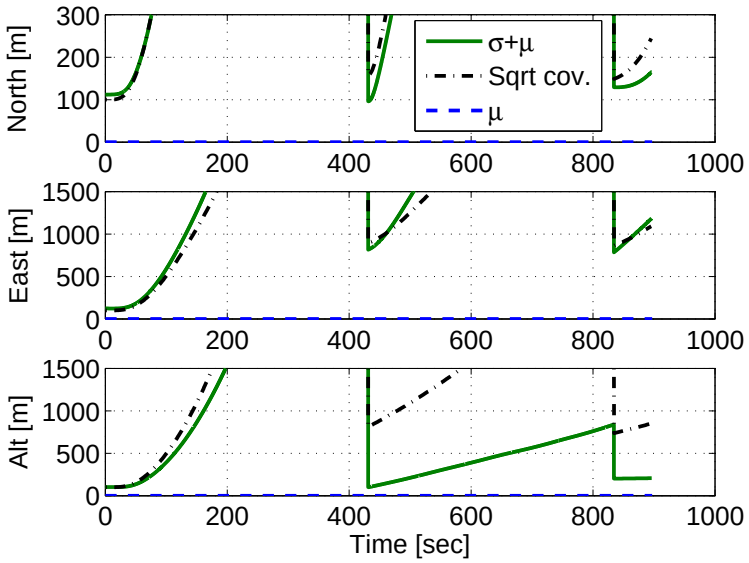


(b) Bias estimation errors.

Figure 3.5: Monte-Carlo results of the three-view navigation-aiding algorithm based on navigation simulation and synthetic imagery data.



(a) Position errors.



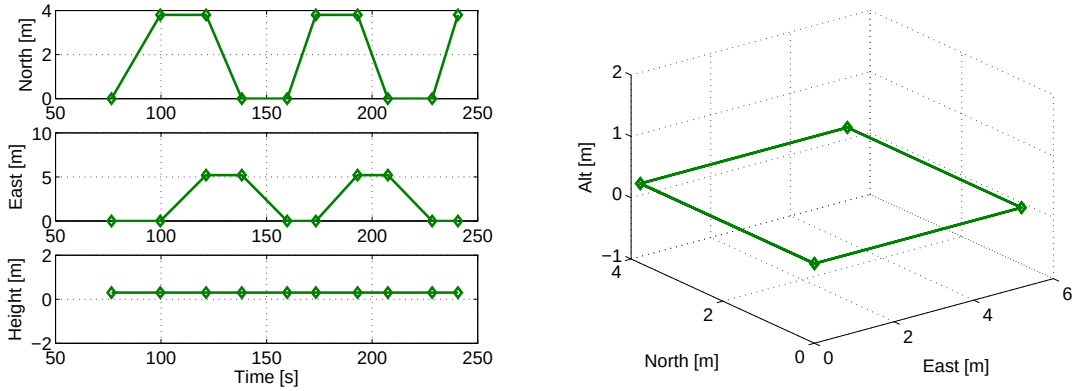
(b) Position errors - zoom.

Figure 3.6: Monte-Carlo results of the three-view navigation-aiding algorithm based on a navigation simulation and synthetic imagery data without applying epipolar constraints.

3.4.3 Experiment Results

An experiment was carried out for validating the proposed method. The experimental setup contained an MTi-G Xsens[3] IMU/INS and a 207MW Axis network camera[4] that were mounted on top of a ground vehicle. The vehicle was manually commanded using a joystick, while the camera captured images perpendicular to the motion heading. During the experiment, the inertial sensor measurements and camera images were recorded for post-processing at 100 Hz and 15 Hz, respectively. In addition, these two data sources were synchronized by associating to each image a time stamp from the navigation timeline.

Since the experiment was carried out indoors, GPS was unavailable, and therefore the MTi-G could not supply a valid navigation solution for reference. However, the true vehicle trajectory was manually measured during the experiment and associated with a timeline by post-processing the inertial sensors readings. The reference trajectory is shown in Figure 3.7. The diamond markers denote the manual measurements of the vehicle position, while the solid line represents a linear interpolation between each two markers. The vehicle began its motion at $t \approx 76$ seconds. As can be seen in Figure 3.7(a), the vehicle performed the same closed trajectory twice.



(a) True trajectory

(b) True trajectory - 3D view

Figure 3.7: Trajectory performed in the experiment.

The recorded inertial sensor measurements were processed by the strapdown block yielding an inertial navigation solution. Sets of three images with a common overlapping area were identified and chosen. The proposed algorithm was applied for each such set and used for updating the navigation system. Two different update modes are demonstrated in this experiment: a) "Sequential update", in which all the three images are acquired closely to each other, and b) "Loop update", in which the first two images are captured while the platform passes a given region for the first time, whereas the third image is

[3] http://www.xsens.com/en/general/mti-g.
[4] http://www.axis.com/products/cam_207mw/index.htm.

obtained at the second passing of the same region. The algorithm application is the same in both cases.

(a) (b)

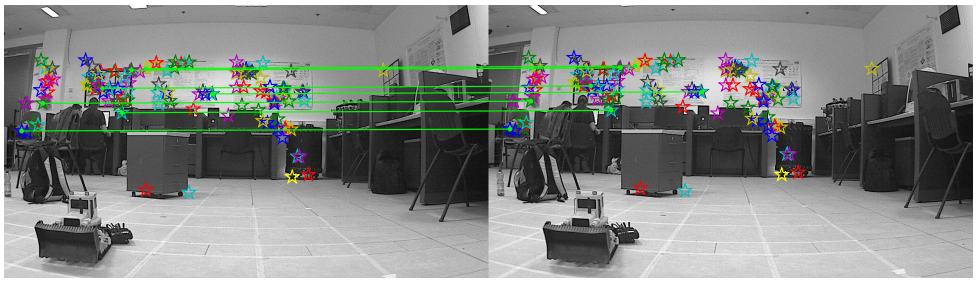
(c)

Figure 3.8: Three camera-captured images used in the first sequential update in the experiment.

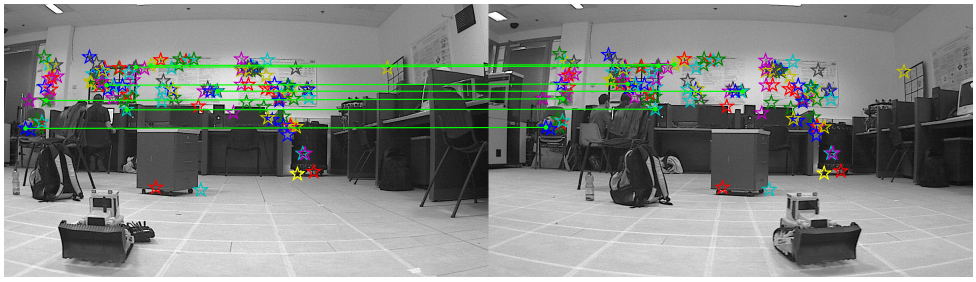
The image matching process for the first set of three overlapping images is shown in Figures 3.8 and 3.9: Figure 3.8 shows the camera-captured images, while Figure 3.9 provides the set of matching triplets $\{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$, showing matches between each pair of images. For example, Figure 3.9(a) shows the matches between the first and second image, such that $(\mathbf{x}_1, \mathbf{x}_2) \in \{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$. As seen, the three images have a significant common overlapping area, and thus it is possible to obtain a large number of matching triplets. About 140 matching triplets were found for the three images shown in Figures 3.8(a)-3.8(c); however, only a few of them are explicitly shown in Figures 3.9(a)-3.9(c), while the rest of the matches are denoted by various markers.

The localization results are shown in Figure 3.10. Figure 3.10(a) presents the estimated position compared to the true position. In addition, inertial-navigation-based position estimation is shown for comparison. Figure 3.10(b) depicts the position estimation errors (computed by subtracting the true position from the estimated position) and the square root of the filter covariance. The update mode is presented in both figures: until $t \approx 150$ seconds sequential updates were performed, while loop updates were applied after the platform has completed a loop, starting from $t \approx 158$ seconds.

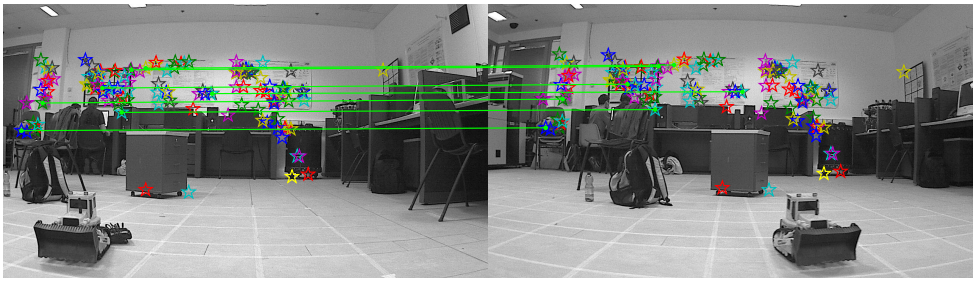
During the sequential updates phase, the time instances (t_1, t_2, t_3) were chosen such that $t_2 - t_1 \approx 1$ seconds and $t_3 - t_2 \approx 5$ seconds. As seen in Figure 3.10, while sequential



(a)



(b)



(c)

Figure 3.9: Image matching process based on images shown in Figure 3.8. (a) Matching triplets between image 1 and 2: $(\mathbf{x}_1, \mathbf{x}_2) \in \{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$; (b) Matching triplets between image 2 and 3: $(\mathbf{x}_2, \mathbf{x}_3) \in \{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$; (c) Matching triplets between image 1 and 3: $(\mathbf{x}_1, \mathbf{x}_3) \in \{\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i\}_{i=1}^{N_{123}}$. For clarity, only the first few matches are explicitly shown; the rest of the matches are denoted by marks in each image.

updates are active, the position is estimated with an accuracy of several meters, whereas the inertial solution rapidly diverges. The consistent behavior of the filter covariance indicates that the correlation between $\mathbf{X}(t_3)$ and $\mathbf{X}(t_2)$, which is not accounted for in the current filter formulation (cf. Section 3.3), is not significant.

Although the position error is significantly reduced during the sequential updates of the algorithm (until $t \approx 120$ seconds), its development is mitigated during this phase but not entirely eliminated, as clearly evident in the height error. Two main reasons for this

phenomenon are: a) Imperfect estimation of the actual IMU errors; b) In each update, the algorithm allows reducing current position errors only to the level of errors that were present while the first two images of the three, were taken. Because each update in the sequential mode uses a different set of three images, and because the development of inertial error between these images, the error – although considerably mitigated – will continue to develop.

After the vehicle had completed its first loop, it became possible to apply the algorithm in a "loop update" mode. As seen in Figure 3.10, the loop updates were applied at a varying frequency, which was typically lower than the frequency of sequential updates. Referring to Figure 3.7, the vehicle completed its first loop at $t \approx 158$ seconds and performed the same trajectory once again, completing the second loop at $t \approx 230$ and afterwards continuing the same basic trajectory for another 10 seconds. In these last 10 seconds the vehicle began performing a third loop.

Each loop update significantly reduces the inertially-accumulated position error, yielding a small error of several meters after over 150 seconds of operation. For comparison, the inertial error approaches 1100 meter (in the north axis) over this period of time, indicating the low quality of the inertial sensors. Note that the position error is reduced in *all* axes, including along the motion direction, which is not possible in two-view methods for navigation aiding.

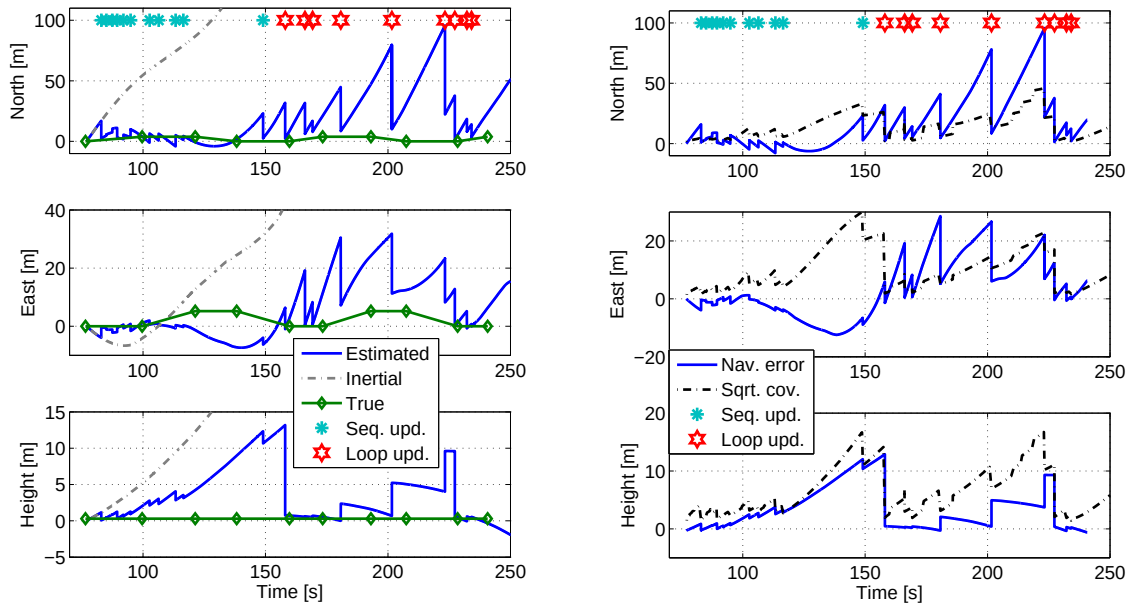
As seen in Figure 3.10, although each loop update drastically reduces the developed position error, the rate of the inertially-developing position error between each two loop updates has not been arrested compared to the pure inertial scenario (cf. Figure 3.10(a)), leading to the conclusion that the IMU errors parametrization (drift and bias) were not estimated well in the experiment.

Note also that as additional loop updates are applied and until reaching $t \approx 230$ seconds, the update accuracy deteriorates. For example, the east position error is reduced to -1.5 meters at the first loop update ($t = 158$ seconds), while in the loop update at $t = 201$ seconds the east position error was reduced only to 6 meters. The reason for this accuracy deterioration is that each loop update is performed using the current image and two images of the same scene that had been captured while the platform visited the area for the first time. As already mentioned, each update allows to reduce the current position error to the level of errors that were present while the first two images were captured. However, as can be seen from Figure 3.10, the position error in the sequential updates phase, although considerably arrested, gradually increases over time, and hence the loop updates are capable of reducing the position error to the level of errors that increase with time. For example, the first two images participating in the first loop update at $t = 158$ seconds were captured at $t = 77$ and $t = 78$ seconds, while the first two images participating in the loop update at $t = 201$ seconds were captured at $t = 131$ and $t = 132$ seconds. Since the position error at $t = 131$ and $t = 132$ seconds was larger than the position error at $t = 77$ and $t = 78$ seconds (cf. Figure 3.10(b)), the position error after the loop update at $t = 201$ seconds was accordingly larger than the position error after

the first loop update (at $t = 158$ seconds).

After $t \approx 230$ seconds, the platform began its third loop and thus the loop updates from $t \approx 230$ and on were performed using images (and the attached navigation data) captured at the beginning of the platform's trajectory (around $t = 80$ seconds). Therefore, the obtained position error at these loop updates is of accuracy comparable to the accuracy of the first loop updates (starting from $t = 158$ seconds), and hence to the accuracy of the navigation solution calculated in the beginning of the trajectory.

Analyzing the experiment results, it is also tempting to compare the performance obtained in the sequential and loop update modes. However, because these two modes of algorithm activation were not applied in the same phase, a quantitative analysis cannot be performed. Nevertheless, regardless of the sequential update mode, it is safe to state that activation of the algorithm in a loop update mode reduces the position errors in all axes to prior values while processing only three images.



(a) Estimated position.

(b) Position estimation error vs. filter uncertainty covariance

Figure 3.10: Experiment results. A small position error (several meters) is obtained while sequentially activating the algorithm. The position error is reset to its prior levels each time a loop update is applied.

3.5 Conclusions

This chapter presented a new method for vision-aided navigation based on three-view geometry. Camera-captured images were stored and associated with partial navigation data taken from the inertial navigation system. These images were used for constructing a representation of the observed environment, while some of them were also incorporated for navigation aiding. The proposed method utilized three overlapping images to formulate constraints relating between the platform motion at the time instances of the three images. A new formulation of such constraints was developed. The associated navigation data for each of the three images allowed to determine the scale ambiguity inherent to all pure computer vision techniques for motion estimation. The constraints were further reformulated and fused with an inertial navigation system using an implicit extended Kalman filter. A single activation of the method over a set of three overlapping images reduces the inertially developed position errors in all axes to the levels present while the first two images were captured. Navigation errors in other states were also reduced.

The developed method for vision-aided navigation may be used in various applications in which three overlapping images, and the required navigation data, are available. In this chapter the method was applied to maintaining small navigation errors, while operating in a GPS-denied environment, accomplished by engaging the algorithm over sequential overlapping imagery, and utilizing the overlapping images in case a loop in the trajectory occurs. In contrast to the existing methods for vision-aided navigation, which are also capable of handling loops, such as bundle adjustment and SLAM, the computational requirements of the proposed algorithm allow real-time navigation aiding, since a constant-size state vector is used, and only three images are processed at each update step of the IEKF. The refinement process of the environment representation, such as mosaic image construction, may be performed in a background process.

The method was examined based on real imagery and navigation data, obtained in an experiment, and in a statistical study using simulated navigation and synthetic imagery. The results showed that reduced position and velocity errors can be maintained over time, thus allowing operation without relying on the GPS signal. Specifically, the position errors obtained in the experiment, in which a low-grade IMU was used, were reduced to several meters each time the algorithm was applied, while the inertial position error has reached over 1000 meters in 150 seconds of operation. The implication of this result is important for various applications, in which the GPS signal is unavailable or unreliable. Among these is a holding pattern mission, in which the platform has to perform the same loop trajectory numerous times. Satellite orbit determination is another possible application.

Chapter 4

Graph-Based Cross-Covariance Calculation for a General Multi-Platform Measurement Model

Contents

This chapter addresses the problem of consistent distributed cooperative navigation. A group of collaborative platforms, capable of intercommunication, is assumed. Each platform is equipped with its own dead reckoning or inertial navigation sensors, and with additional onboard sensors. For a general multi-platform measurement model that involves both inertial navigation data and other onboard sensor readings, taken at different

time instances, the various sources of information become correlated. Thus, in the process of information fusion, this correlation should be solved for to obtain consistent state estimation.

The common approach for obtaining the correlation terms is to maintain an augmented covariance matrix. This method works for relative pose measurements (e. g., [48]), but is impractical for a general MP measurement model, because the identities of the platforms involved in generating the measurements, as well as the measurement time instances, are unknown a priori.

As mentioned in Section 1.1.5, several methods were proposed to avoid correlated updates [58], or eliminating the need in calculating the correlation terms by tracking the origins of measurements [62]. However, such methods do not utilize the full potential of the available measurements, since not all the measurements are actually incorporated.

In this chapter, it is proposed to explicitly calculate the required correlation terms based on the history of all the thus-far performed MP measurements. As common in many CN methods, including [48],[50],[52],[62], an extended Kalman filter is used for data fusion. The proposed approach relies on graph theory. The graph is locally maintained by every platform in the group, representing all the MP measurement updates. The developed method calculates the correlation terms in the most general scenarios of MP measurements while properly handling the involved process and measurement noise.

In contrast to [44] (cf. Section 1.1.5), the proposed method explicitly calculates the required correlation terms, allowing to perform navigation updates without applying smoothing over the past navigation history of the cooperative platforms, and is therefore computationally efficient.

Consequently, the main contributions of this chapter are twofold. First, a graph-based method for an explicit calculation of cross-covariance terms, required for consistent CN, is developed. The method assumes a general MP measurement model, relating any number of platforms that may contribute information from different time instances. The identities of these platforms and the time instances are a priori unknown. Second, the effect of process and measurement noise on the calculated cross covariances is analyzed and a method for incorporating these noise terms into the calculated cross-covariance terms is developed.

4.1 Problem Description

Consider a group of N cooperative platforms capable of intercommunication. Each platform is equipped with inertial navigation sensors and hence is capable of calculating its own navigation solution, comprised of position, velocity and angular orientation. Similarly to Section 1.3.2, denote by $\mathbf{x}_i$ and $\mathbf{x}_i^t$ the calculated and the (unknown) true navigation solutions of the ith platform, respectively, and let $\mathbf{y}_{i,IMU}$ represent the measurements of the platform's inertial navigation sensors. The errors in $\mathbf{y}_{i,IMU}$ are modeled by an

unknown vector of parameters $\boldsymbol{\beta}_i^t$. Denote by $\boldsymbol{\beta}_i$ the calculated model of inertial sensor errors, used for correcting the measurements $\mathbf{y}_{i,IMU}$. For instance, the vector $\boldsymbol{\beta}$ includes a collection of accelerometer and gyro biases.

Let

$$\boldsymbol{\zeta}_i(t_k) \doteq \begin{bmatrix} \mathbf{x}_i(t_k) \\ \boldsymbol{\beta}_i(t_k) \end{bmatrix} \quad , \quad \boldsymbol{\zeta}_i^t(t_k) \doteq \begin{bmatrix} \mathbf{x}_i^t(t_k) \\ \boldsymbol{\beta}_i^t(t_k) \end{bmatrix} \tag{4.1}$$

and $\mathcal{N} \doteq \{1, \ldots, N\}$. Then

$$\boldsymbol{\zeta}_i(t_{k+1}) = \mathbf{f}(\boldsymbol{\zeta}_i(t_k), \mathbf{y}_{i,IMU}(t_k)) \quad , \quad i \in \mathcal{N} \tag{4.2}$$

The following navigation error state vector is defined

$$\mathbf{X}_i(t) \doteq \begin{bmatrix} \mathbf{x}_i(t) - \mathbf{x}_i^t(t) \\ \boldsymbol{\beta}_i(t) - \boldsymbol{\beta}_i^t(t) \end{bmatrix} \equiv \boldsymbol{\zeta}_i(t) - \boldsymbol{\zeta}_i^t(t) \tag{4.3}$$

As discussed in Section 1.3.2, the evolution of the state vector $\mathbf{X}_i$ can be modeled by the linear time-varying stochastic model:

$$\dot{\mathbf{X}}_i(t) = \Phi^i(t)\mathbf{X}_i(t) + \boldsymbol{\omega}^i(t) \quad , \quad i \in \mathcal{N} \tag{4.4}$$

where Φ^i is the continuous system matrix and $\boldsymbol{\omega}^i$ is the process noise, which is assumed to be white and zero-mean Gaussian. This continuous time model can be replaced by a discrete model

$$\mathbf{X}_i(t_b) = \Phi^i_{t_a \to t_b}\mathbf{X}_i(t_a) + \boldsymbol{\omega}^i_{t_a \to t_b} \quad , \quad i \in \mathcal{N} \tag{4.5}$$

where $\Phi^i_{t_a \to t_b}$ is the discrete system matrix relating the state between any two time instances t_a and t_b, $t_b > t_a$, and $\boldsymbol{\omega}^i_{t_a \to t_b}$ is the equivalent discrete process noise.

In addition to the inertial sensors, each platform is equipped with its own set of onboard exogenous sensors. The readings of the exogenous sensors of the jth platform at some time instant t_a are denoted by $\mathbf{y}_j(t_a)$ (as opposed to $\mathbf{y}_{j,IMU}$, that denotes the IMU measurements). These measurements are corrupted by a Gaussian white noise $\mathbf{v}_j(t_a)$. Let $\mathbf{y}_j^t(t_a) \doteq \mathbf{y}_j(t_a) - \mathbf{v}_j(t_a)$.

Consider a general measurement model that relates the navigation data and onboard sensor measurements of several platforms, possibly taken at different time instances. Let j denote the identities of the platforms involved in this measurement model, $j \in \mathcal{N}$.

The considered measurement model can be formulated in an implicit form as

$$\mathbf{z}(t) = \mathbf{h}(\{\boldsymbol{\zeta}_j(t_i), \mathbf{y}_j(t_i)\}_{i=1}^r) \quad , \quad j \in \mathcal{N} \tag{4.6}$$

where $\mathbf{z}$ is the residual measurement, which is a function of $\boldsymbol{\zeta}_j(t_i)$, representing the navigation solution $\mathbf{x}_j(t_i)$ and parametrization of the inertial sensors errors $\boldsymbol{\beta}_j(t_i)$, and the onboard sensor readings $\mathbf{y}_j(t_i)$ of the jth platform at time t_i, with $t_i \leq t$ and t being the current time. The parameter r denotes the overall number of information sets $(\boldsymbol{\zeta}_j(t_i), \mathbf{y}_j(t_i))$ constituting $\mathbf{z}$. If each of the participating platforms contributes only a

single information set, r represents the number of platforms involved in the residual measurement $\mathbf{z}$. However, in the general case, each platform may contribute information from several time instances. For example, if some platform j contributes information from two time instances $t_j^1 \doteq t_1$ and $t_j^2 \doteq t_2$, then $\mathbf{z}$ will be a function of $(\boldsymbol{\zeta}_j(t_j^1), \mathbf{y}_j(t_j^1))$ and $(\boldsymbol{\zeta}_j(t_j^2), \mathbf{y}_j(t_j^2))$.

To simplify the notation, it is assumed from this point onward that the identity of the platforms forming $\mathbf{z}$ is given by $1, \ldots, r$; cases in which a platform contributes information from several time instances are treated as if this information was provided by different platforms. Thus, the residual measurement $\mathbf{z}$ can be written as:

$$\mathbf{z}(t) = \mathbf{h}(\{\boldsymbol{\zeta}_i(t_i), \mathbf{y}_i(t_i)\}_{i=1}^{r}) \tag{4.7}$$

Linearizing Eq. (4.7) about $\boldsymbol{\zeta}_i^t(t_k)$ and $\mathbf{y}_i^t(t_i)$ gives

$$\mathbf{z}(t) \approx \sum_{i=1}^{r} H_i(t_i)\mathbf{X}_i(t_i) + D_i(t_i)\mathbf{v}_i(t_i) \tag{4.8}$$

where

$$H_i(t_i) = \nabla_{\boldsymbol{\zeta}_i^t(t_i)}\mathbf{h} \quad , \quad D_i(t_i) = \nabla_{\mathbf{y}_i^t(t_i)}\mathbf{h} \tag{4.9}$$

since $\boldsymbol{\zeta}_i^t(t_k)$ and $\mathbf{y}_i^t(t_i)$ are unknown, the Jacobian matrices are approximated by

$$H_i(t_i) = \nabla_{\boldsymbol{\zeta}_i(t_i)}\mathbf{h} \quad , \quad D_i(t_i) = \nabla_{\mathbf{y}_i(t_i)}\mathbf{h} \tag{4.10}$$

The update step of the Kalman filter involves cross-covariance terms relating the different state vectors that appear in the measurement model (4.8). Denoting by $\tilde{\mathbf{X}}$ the estimation error of $\mathbf{X}$, the required cross-covariance terms are $E[\tilde{\mathbf{X}}_i(t_i)\tilde{\mathbf{X}}_j^T(t_j)]$ with $i, j = 1 \ldots r, i \neq j$. If these terms are known, a consistent measurement update can be employed.

The purpose of this chapter is to present an efficient method to compute the cross-covariance matrices on-demand while the identity of the involved platforms, i. e. the indices i and j, and the time instances t_i and t_j are unknown a priori. It is tempting to apply the common approach, used when considering relative pose measurements for CN [48], wherein an augmented covariance matrix is maintained, consisting of the covariance matrices of all the platforms in the group and of cross-covariance matrices relating any pair of platforms. However, this approach can be only applied when the measurement model involves concurrent information from different platforms, as indeed is the case with relative pose measurements.

In the case of a general measurement model (4.8), in addition to the a priori unknown identity of the r platforms contributing to the multi-platform measurement, the involved time instances are also unknown a priori. Therefore, maintaining all the possible cross-covariance terms is not a practical solution in terms of both computational load and storage requirements. Instead, it is suggested to calculate the required cross-covariance terms *on-demand* for a general MP measurement model.

4.2 Concept of Explicit Cross-Covariance Calculation

Before presenting the general concept behind the proposed approach, the calculation of cross-covariance terms is illustrated in the following basic example.

4.2.1 A Basic Example

In this example, a measurement comprised of information obtained from three different platforms, i. e. $r = 3$, is considered. The residual measurement $\mathbf{z}$ may therefore be written as

$$\mathbf{z} \approx H_3(t_3)\mathbf{X}_3(t_3) + H_2(t_2)\mathbf{X}_2(t_2) + H_1(t_1)\mathbf{X}_1(t_1) + D\mathbf{v} \tag{4.11}$$

with $D \doteq \begin{bmatrix} D_3(t_3) & D_2(t_2) & D_1(t_1) \end{bmatrix}$ and $\mathbf{v} \doteq \begin{bmatrix} \mathbf{v}_3^T(t_3) & \mathbf{v}_2^T(t_2) & \mathbf{v}_1^T(t_1) \end{bmatrix}^T$.

Figure 4.1 shows a scenario wherein information transmitted by platforms I and II, with the current information of platform III, is used for updating platform III. Circles denote a priori information, while squares denote update events. Two update events are shown in the figure. While a_1, a_2 and a_3 represent information used in the first update, b_1, b_2 and b_3 represent information used in the second update. Let t_{a_i} and t_{b_i} represent the time instances corresponding to a_i and b_i, respectively, with $i = 1, 2, 3$.

Assume that the first update was carried out and that the a priori covariance matrices of the 3 platforms and all the cross-covariance matrices between these platforms, at the time instances t_{a_1}, t_{a_2} and t_{a_3}, were stored. Assume also that the required information for the second update is available. The key question is how to calculate the cross-covariance terms required for executing the second update, i. e. $E[\tilde{\mathbf{X}}_{III}^-(t_{b_3})\tilde{\mathbf{X}}_{II}^-(t_{b_2})]$, $E[\tilde{\mathbf{X}}_{III}^-(t_{b_3})\tilde{\mathbf{X}}_{I}^-(t_{b_1})]$ and $E[\tilde{\mathbf{X}}_{II}^-(t_{b_2})\tilde{\mathbf{X}}_{I}^-(t_{b_1})]$.

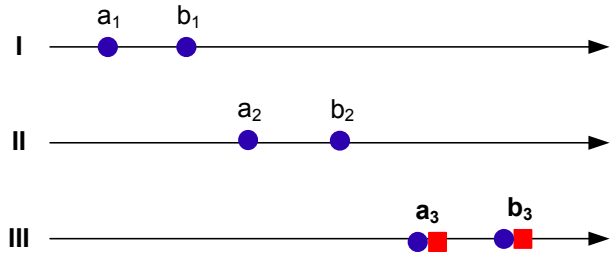


Figure 4.1: Measurement schedule example based on a measurement model that involves 3 platforms. Platform III is updated based on information transmitted by platforms I and II. The circles denote information included in the measurement, squares indicate update events.

In particular, consider the calculation of $E[\tilde{\mathbf{X}}_{III}^-(t_{b_3})\tilde{\mathbf{X}}_{II}^-(t_{b_2})]$. Since no updates of any

kind were performed between a_2 and b_2:

$$\tilde{\mathbf{X}}^{-}_{II}(t_{b_2}) = \Phi^{II}_{a_2\to b_2}\tilde{\mathbf{X}}^{-}_{II}(t_{a_2}) + \boldsymbol{\omega}^{II}_{a_2\to b_2} \tag{4.12}$$

In a similar manner, it is possible to write a transition relation between the a posteriori estimation error at a_3 and the a priori estimation error at b_3:

$$\tilde{\mathbf{X}}^{-}_{III}(t_{b_3}) = \Phi^{III}_{a_3\to b_3}\tilde{\mathbf{X}}^{+}_{III}(t_{a_3}) + \boldsymbol{\omega}^{III}_{a_3\to b_3} \tag{4.13}$$

Thus,

$$E[\tilde{\mathbf{X}}^{-}_{III}(t_{b_3})\tilde{\mathbf{X}}^{-}_{II}(t_{b_2})] = E\left[\left(\Phi^{III}_{a_3\to b_3}\tilde{\mathbf{X}}^{+}_{III}(t_{a_3}) + \boldsymbol{\omega}^{III}_{a_3\to b_3}\right)\left(\Phi^{II}_{a_2\to b_2}\tilde{\mathbf{X}}^{-}_{II}(t_{a_2}) + \boldsymbol{\omega}^{II}_{a_2\to b_2}\right)^T\right] \tag{4.14}$$

while the a posteriori estimation error at a_3 is given by

$$\begin{aligned}\tilde{\mathbf{X}}^{+}_{III}(t_{a_3}) &= \left(I - K_{a_3}H_{a_3}\right)\tilde{\mathbf{X}}^{-}_{III}(t_{a_3}) - K_{a_3}H_{a_2}\tilde{\mathbf{X}}^{-}_{II}(t_{a_2}) \\ &- K_{a_3}H_{a_1}\tilde{\mathbf{X}}^{-}_{I}(t_{a_1}) - K_{a_3}D_a\mathbf{v}_a\end{aligned} \tag{4.15}$$

where K_{a_3} is the Kalman gain matrix, calculated by platform III at the first measurement update.

Since $\boldsymbol{\omega}^{II}_{a_2\to b_2}$ is statistically independent of $\tilde{\mathbf{X}}^{-}_{III}(t_{a_3}), \tilde{\mathbf{X}}^{-}_{II}(t_{a_2}), \tilde{\mathbf{X}}^{-}_{I}(t_{a_1})$, and since $\boldsymbol{\omega}^{III}_{a_3\to b_3}$ is statistically independent of $\tilde{\mathbf{X}}^{-}_{II}(t_{a_2})$ and $\boldsymbol{\omega}^{II}_{a_2\to b_2}$ (cf. Figure 4.1):

$$E\left[\tilde{\mathbf{X}}^{+}_{III}(t_{a_3})(\boldsymbol{\omega}^{II}_{a_2\to b_2})^T\right] = 0 \tag{4.16}$$

$$E\left[\boldsymbol{\omega}^{III}_{a_3\to b_3}\left(\Phi^{II}_{a_2\to b_2}\tilde{\mathbf{X}}^{-}_{II}(t_{a_2}) + \boldsymbol{\omega}^{II}_{a_2\to b_2}\right)^T\right] = 0 \tag{4.17}$$

In addition,

$$E\left[\mathbf{v}_a\left(\Phi^{II}_{a_2\to b_2}\tilde{\mathbf{X}}^{-}_{II}(t_{a_2}) + \boldsymbol{\omega}^{II}_{a_2\to b_2}\right)^T\right] = 0 \tag{4.18}$$

Let $\tilde{\mathbf{X}}^{-}_{i}(t_{a_i})$ be represented by $\tilde{\mathbf{X}}_{a_i}$ and denote $P_{ab} \doteq E[(\tilde{\mathbf{X}}_a)(\tilde{\mathbf{X}}_b)^T]$. Incorporating Eqs. (4.15)-(4.18) into Eq. (4.14) yields

$$P^{-}_{b_3b_2} = \Phi^{III}_{a_3\to b_3}\left\{\left(I - K_{a_3}H_{a_3}\right)P^{-}_{a_3a_2} - K_{a_3}H_{a_2}P^{-}_{a_2a_2} - K_{a_3}H_{a_1}P^{-}_{a_1a_2}\right\}\left(\Phi^{II}_{a_2\to b_2}\right)^T \tag{4.19}$$

Thus, $P^{-}_{b_3b_2}$ is expressed via the filter gain matrix, the measurement matrices, covariance and cross-covariance matrices from the past MP updates, which therefore need to be stored. The other two required cross-covariance terms in this example can be calculated using the same process, yielding an equivalent expression for $P^{-}_{b_3b_1}$, while $P^{-}_{b_2b_1} = 0$.

4.2.2 A General Scenario

The approach discussed above can be generalized to any number of MP measurement updates based on the general measurement model formulated in Eq. (4.8).

The general cross-covariance term $E[\tilde{\mathbf{X}}_i(t_i)\tilde{\mathbf{X}}_j^T(t_j)]$ can be found by expressing each of the two state vectors $\tilde{\mathbf{X}}_i(t_i)$ and $\tilde{\mathbf{X}}_j(t_j)$ according to the history of the MP measurement updates, and then calculating $E[\tilde{\mathbf{X}}_i(t_i)\tilde{\mathbf{X}}_j^T(t_j)]$ based on the resulting expressions, while judiciously handling the involved noise terms. In contrast to the example from the previous section, in the general case the process and measurement noise terms are not necessarily statistically independent of the involved state vectors.

Clearly, sustaining the aforementioned approach requires storing the information involved in all the past MP measurement updates, including the filter gain, measurement, covariance and cross-covariance matrices. If this information is available for a *specific* sequence of MP measurement updates, the required cross-covariance terms can be calculated based on the process demonstrated in the previous section. In the following sections, however, a method for on-demand calculation of the cross-covariance terms for a *general* case is developed. The method uses a graph representation, locally maintained by every platform in the group, containing the information from all the past MP measurement updates.

The proposed graph topology relies upon a directed acyclic graph (DAG). Denote by t_i^{MP} the most recent time instant in which the ith platform was updated by any MP measurement. In a general MP system, the DAG topology is representative if each MP measurement is utilized for updating only the platforms $i \in \{1, \ldots, r\}$, which contributed their navigation data from the time instant $t_i > t_i^{MP}$ and assuming these platforms contributed a single information set $(\boldsymbol{\zeta}_i(t_i), \mathbf{y}_i(t_i))$ (cf. Section 4.1). In particular, the graph remains acyclic when only platforms that contributed their current navigation information, i. e. $t_i = t$, are updated. For simplicity, in this chapter we consider only *one* such platform.

It is worth noting that if some platform i contributed $l > 1$ information sets $(\boldsymbol{\zeta}_i(t_i^1), \mathbf{y}_i(t_i^1)), (\boldsymbol{\zeta}_i(t_i^2), \mathbf{y}_i(t_i^2)), \ldots, (\boldsymbol{\zeta}_i(t_i^l), \mathbf{y}_i(t_i^l))$, with $t_i^1 < t_i^2 < \cdots < t_i^l$, to the MP measurement (4.6), this platform can be updated, while sustaining an acyclic graph, at the time instant t_i^l, provided that $t_i^l > t_i^{MP}$.

Denoting by q the identity of the updated platform, its a posteriori estimation error in a general MP measurement model, formulated in Eq. (4.8), can be expressed as

$$\tilde{\mathbf{X}}_q^+(t_q) = (I - K_q H_q)\,\tilde{\mathbf{X}}_q^-(t_q) - K_q \sum_{i=1\ ,\ i\neq q}^{r} H_i \tilde{\mathbf{X}}_i^-(t_i) - K_q \sum_{i=1}^{r} D_i \mathbf{v}_i(t_i) \tag{4.20}$$

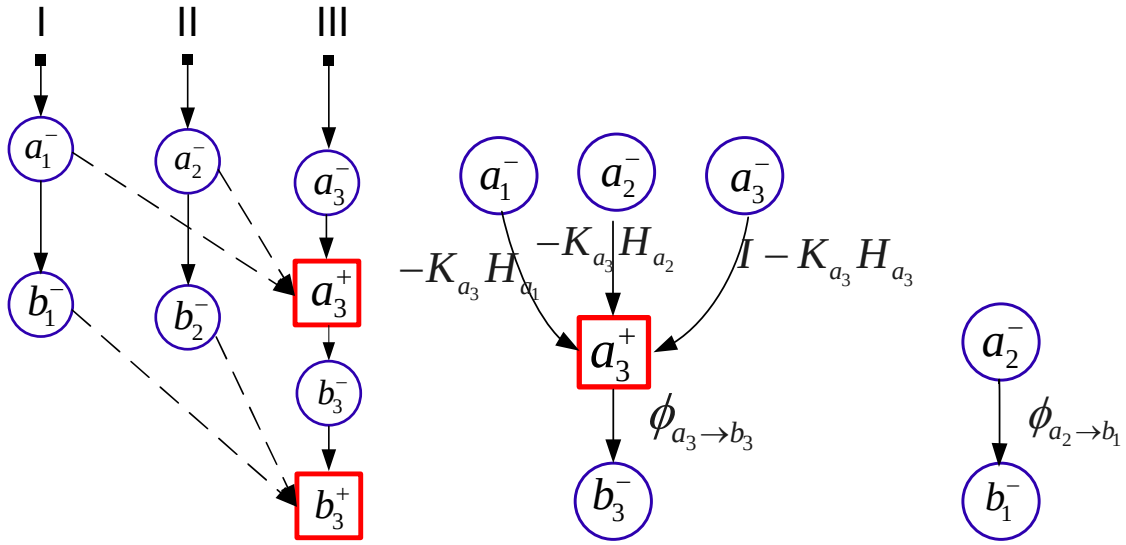
where K_q is the Kalman gain matrix computed for the qth platform. The a priori estimation error of some platform i, based on Eq. (4.5), is given by

$$\tilde{\mathbf{X}}_i^-(t_b) = \Phi_{t_a \to t_b}^i \tilde{\mathbf{X}}_i^-(t_a) + \boldsymbol{\omega}_{t_a \to t_b}^i \tag{4.21}$$

4.2.3 Graph Representation

Every platform in the group locally maintains its own copy of the DAG $G = (V, E)$, where V is the set of nodes and E is the set of directed weighted arcs. The weight of each arc reflects the information flow between the two connected nodes.

Two type of nodes exist in V. Nodes of the first type represent a priori information obtained from different platforms in the group, constituting the MP measurements. These nodes are called *a priori nodes*. A single such node represents, therefore, $\boldsymbol{\zeta}_i(t_i)$ and $\mathbf{y}_i(t_i)$ – navigation data and readings of onboard sensors of the ith platform from time instant t_i, respectively. This information is transmitted by the ith platform to the updated platform q at the current time t (cf. Eq. (4.7)). In the general case, $t_i \leq t$. Nodes of the second type represent update events, i. e. the a posteriori information of the updated platform. Such nodes are called *a posteriori nodes*. Thus, each MP measurement update is represented by $r + 1$ nodes. Figure 4.2(a) shows the graph obtained for the 3-platform measurement example considered in Section 4.2.1. A priori nodes are indicated in the graph by circles, while a posteriori nodes are designated by squares.



(a) (b)

Figure 4.2: (a) Graph representation for the scenario shown in Figure 4.1. (b) The trees $T_{b_3^-}$ and $T_{b_1^-}$ required for calculating $P^-_{b_3 b_1}$.

We proceed by presenting the following definitions.

Definition 4.2.1 *A thread of the ith platform is a sub-graph of G, containing all the nodes in V that represent information of the ith platform and arcs in E connecting between these nodes.*

Each platform in the group has its own thread in G.

Definition 4.2.2 *The transition relation is given by*

$$\tilde{\mathbf{X}}_b = \Phi^i_{t_a \to t_b} \tilde{\mathbf{X}}_a + \boldsymbol{\omega}^i_{t_a \to t_b} \tag{4.22}$$

where $a, b \in V$ are any two adjacent a priori nodes in the ith thread, representing $\tilde{\mathbf{X}}_i^-(t_a)$ and $\tilde{\mathbf{X}}_i^-(t_b)$, respectively.

The transition relation connects between the a priori estimation errors of the ith platform at two different time instances t_a and t_b, as expressed by Eq. (4.21). The nodes a and b, both located in thread i, are connected by an arc, weighted by the transition matrix $w(a,b) = \Phi^i_{t_a \to t_b}$. The noise process covariance matrix $Q^i_{t_a \to t_b} \doteq E[\boldsymbol{\omega}^i_{t_a \to t_b} (\boldsymbol{\omega}^i_{t_a \to t_b})^T]$ is associated to this arc as well. For example, the nodes a_1^- and b_1^- in Figure 4.2(a) are connected by an arc representing a transition relation.

Each thread in G can also contain a posteriori nodes. In such a case, G will contain r a priori nodes that are connected to an a posteriori node, located in the thread of the updated platform q, by an update relation, defined as follows (cf. also Eq. (4.20)).

Definition 4.2.3 *Denote by α the a posteriori node, representing $\tilde{\mathbf{X}}_q^+(t_\alpha)$, and by β_i the a priori nodes, representing $\tilde{\mathbf{X}}_i^-(t_{\beta_i})$, with $i = 1, \ldots, r$. The update relation is given by:*

$$\tilde{\mathbf{X}}_\alpha = \left(I - K_\alpha H_{\beta_q}\right) \tilde{\mathbf{X}}_{\beta_q} - K_\alpha \sum_{i=1\ ,\ i \neq q}^{r} H_{\beta_i} \tilde{\mathbf{X}}_{\beta_i} - K_\alpha \sum_{i=1}^{r} D_{\beta_i} \mathbf{v}_{\beta_i} \tag{4.23}$$

where K_α is the Kalman gain computed by the updated platform.

The transition and update relations are illustrated in Figures 4.3(a) and 4.3(b), respectively.

The arc weight $w(\beta_i, \alpha)$, connecting the a priori node β_i with the a posteriori node α is

$$w(\beta_i, \alpha) = \begin{cases} I - K_\alpha H_{\beta_q} & \text{if } i = q \\ -K_\alpha H_{\beta_i} & \text{else} \end{cases} \tag{4.24}$$

In addition, each arc is associated with a measurement noise covariance matrix $K_\alpha D_{\beta_i} R_{\beta_i} (K_\alpha D_{\beta_i})^T$, with $i = 1, \ldots, r$ and $R_{\beta_i} \doteq E[\mathbf{v}_{\beta_i} \mathbf{v}_{\beta_i}^T]$.

For instance, in Figure 4.2(a), the a priori information stored in the nodes a_1^-, a_2^- and a_3^- is connected to the node a_3^+ that represents a posteriori information.

As mentioned in Section 4.2.2, the a priori and a posteriori covariance and cross-covariance terms between the nodes, which participated in the *same* MP update in the past, are known (this information can be stored in the nodes themselves). The construction process of the graph and the communication protocol among the platforms is discussed in Chapter 5.

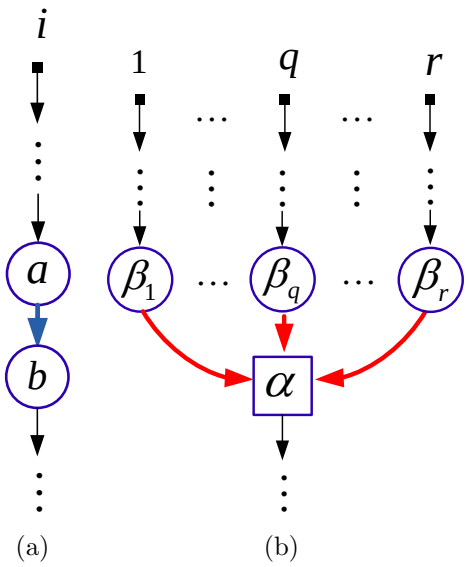


Figure 4.3: (a) The node a is connected to the node b via a transition relation. (b) The nodes β_i, with $i = 1, \ldots, r$, are connected to the node α via an update relation.

4.3 Graph-based Calculation of Cross-Covariance Terms

For a given DAG G, we wish to calculate $E[\tilde{\mathbf{X}}_i(t_i)\tilde{\mathbf{X}}_j^T(t_j)]$, the cross-covariance between the ith platform at t_i and the jth platform at t_j. In this section we use the notation $\tilde{\mathbf{X}}_a$ as an alternative to $\tilde{\mathbf{X}}_i^-(t_i)$ or $\tilde{\mathbf{X}}_i^+(t_i)$, where a is an a priori or a posteriori node in G, respectively. Let the nodes c and d in G represent $\tilde{\mathbf{X}}_i(t_i)$ and $\tilde{\mathbf{X}}_j(t_j)$, respectively. Thus, the goal here is to calculate $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$, which is equivalent to calculating $E[\tilde{\mathbf{X}}_i(t_i)\tilde{\mathbf{X}}_j^T(t_j)]$.

4.3.1 Rationale

The first step is to construct two inverse-trees $T_c = (V_{T_c}, E_{T_c})$ and $T_d = (V_{T_d}, E_{T_d})$, containing all the possible paths in G to each of the nodes c and d. This can be performed as follows. The first tree, T_c, is initialized with the node c. Each next level is comprised of the parents of the nodes that reside in the previous level, as determined from G. For example, the second level of T_c contains all the nodes in G that are directly connected to c. The same process is executed for constructing a tree T_d for the node d. Note that every node in T_c and T_d has only one child but may have one or r parents. In the latter case, the node represents an MP update event. Figure 4.2(b) shows an example of such trees, constructed based on the graph shown in Figure 4.2(a) for calculating the cross-covariance $E[\tilde{\mathbf{X}}_{b_3^-}\tilde{\mathbf{X}}_{b_1^-}^T]$, i. e. $c \equiv b_3^-$ and $d \equiv b_1^-$.

The convention used here is that if some node a_i has several parents, the jth parent

is denoted as a_{i+1}^{j}. Also, $a \equiv a_1$, as shown in Figure 4.4.

Given the two trees T_c and T_d, the cross-covariance term $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ can be computed by expressing $\tilde{\mathbf{X}}_c$ and $\tilde{\mathbf{X}}_d$ using information stored in the nodes from upper levels in the two trees. We start with the first level in the two trees, which is comprised of the node c in T_c, and the node d in T_d. Since the cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is unknown, we proceed to the parents of these nodes, i. e. to the next level in the trees, according to the relation type represented by the arc weights.

Having reached the second level, the term $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ can be expressed using information stored in nodes from the current (second) level and lower levels. For example, assuming a transition relation (4.22) connecting the first two levels in the two trees, $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ can be written, according to Eq. (4.22), in three different forms:

$$\begin{cases} E\left[\tilde{\mathbf{X}}_c \left(\Phi_{d_2 \to d} \tilde{\mathbf{X}}_{d_2} + \boldsymbol{\omega}_{d_2 \to d}\right)^T\right] \\ E\left[\left(\Phi_{c_2 \to c} \tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2 \to c}\right) \tilde{\mathbf{X}}_d^T\right] \\ E\left[\left(\Phi_{c_2 \to c} \tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2 \to c}\right) \left(\Phi_{d_2 \to d} \tilde{\mathbf{X}}_{d_2} + \boldsymbol{\omega}_{d_2 \to d}\right)^T\right] \end{cases} \tag{4.25}$$

where c_2 and d_2 are the parents of c and d, respectively.

Since the expression from the previous (first) level was already checked, it is now required to examine whether any of the expressions involving nodes from the current level are known. In other words, the question is whether any of the pairs $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_{d_2}^T]$, $E[\tilde{\mathbf{X}}_{c_2} \tilde{\mathbf{X}}_d^T]$ and $E[\tilde{\mathbf{X}}_{c_2} \tilde{\mathbf{X}}_{d_2}^T]$ are known. In addition, it is also required to know the correlation between the noise terms and the state vectors.

Since, in general, these pairs are unknown, we proceed to the next (third) level in the trees according to the relation type represented by the arc weights. Now, each of the expressions for $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ obtained while processing the previous (second) level, may be further expanded using information stored in the nodes of the current (third) level.

Continuing the previous example, assume the second and third levels are connected by a transition relation (4.22) in T_c and an update relation (4.23) in T_d, and assume the third platform is updated ($q = 3$). Then one of the possible expressions for $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ would be obtained from $\tilde{\mathbf{X}}_c = \Phi_{c_2 \to c} \tilde{\mathbf{X}}_{c_2} + \boldsymbol{\omega}_{c_2 \to c}$ and

$$\tilde{\mathbf{X}}_d = \Phi_{d_2 \to d} \left[\left(I - K_{d_2} H_{d_3^3}\right) \tilde{\mathbf{X}}_{d_3^3} - K_{d_2} \sum_{i=1\ ,\ i \neq 3}^{r} H_{d_3^i} \tilde{\mathbf{X}}_{d_3^i} - K_{d_2} \sum_{i=1}^{r} D_{d_3^i} \mathbf{v}_{d_3^i} \right] + \boldsymbol{\omega}_{d_2 \to d} \tag{4.26}$$

Note that, compared to Eq. (4.23), $\alpha \equiv d_2$ and $\beta_i = d_3^i$.

Once again, the question is whether the different cross-covariance terms that appear in the new expressions involving current and lower levels are known (had been stored in G in the past). All the expressions from the previous level (the second level) were already

analyzed. Ignoring for the moment terms that involve noise, it is obvious that less terms are to be analyzed when nodes closer to c or d are considered. Therefore, it is preferred to start analyzing from the lower level upward.

If, for example, $E[\tilde{\mathbf{X}}_{c_3}\tilde{\mathbf{X}}_{d_3^1}^T]$, is known, then the nodes $c_3 \in V_{T_c}$ and $d_3^1 \in V_{T_d}$ are either identical ($c_3 \equiv d_3^1$) or represent state vectors that had been used in the same MP measurement. Otherwise, $E[\tilde{\mathbf{X}}_{c_3}\tilde{\mathbf{X}}_{d_3^1}^T]$ would not have been stored in G. In any case, the known term $E[\tilde{\mathbf{X}}_{c_3}\tilde{\mathbf{X}}_{d_3^1}^T]$, properly weighted, is part of $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$. Having a known term also means that there is no need to proceed to nodes of higher levels related to this term.

The procedure continues to higher levels in the two trees until either all the terms required for calculating the cross-covariance $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$ are known, or the top level in both trees has been reached. In the latter case, the unknown terms of the cross-covariance are zero.

The process noise terms are assumed to be statistically independent, $E[\boldsymbol{\omega}_{i_1\to i_2}\boldsymbol{\omega}_{j_1\to j_2}^T] = 0$, if $\boldsymbol{\omega}_{i_1\to i_2}$ and $\boldsymbol{\omega}_{j_1\to j_2}$ belong to different platforms, or, if $\boldsymbol{\omega}_{i_1\to i_2}$ and $\boldsymbol{\omega}_{j_1\to j_2}$ belong to the same platform at non-coinciding time instances, i. e., $(t_{i_1}, t_{i_2}) \cap (t_{j_1}, t_{j_2}) = \phi$. The measurement noise is assumed to be statistically independent of the state vectors involved in the measurement. On the other hand, the process and measurement noise terms may be statistically *dependent* on the involved state vectors (see Section 4.3.2.3).

In the following sections, the above rationale is transformed into an algorithm for calculating the cross-covariance $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$ in a *general* scenario.

4.3.2 Algorithm for Explicit Cross-Covariance Calculation

Let $T_b = (V_{T_b}, E_{T_b})$ be a tree containing all the paths in $G = (V, E)$ to some node $b \in V$, and let $a \in V_{T_b}$ and $\alpha, \beta \in V$. The following notations are used in the remainder of this chapter:

$\pi_b(a)$	Parents of node a in tree T_b
$\mathcal{A}_b(a)$	Ancestors of node a in tree T_b
$\mathcal{D}_b(a)$	Descendants of node a in tree T_b
$a_k \overset{T_b}{\Longrightarrow} a$	Path $a_k \to \cdots \to a_2 \to a$ in tree T_b
$\{a_k \overset{T_b}{\Longrightarrow} a\}$	Group of nodes in the path $a_k \overset{T_b}{\Longrightarrow} a$

Definition 4.3.1 *A pair of nodes (α, β) is said to be known, if $E[\tilde{\mathbf{X}}_\alpha\tilde{\mathbf{X}}_\beta^T]$ is known, i. e., if it can be retrieved from the data stored in G. A known pair (α, β) is denoted by $\odot(\alpha, \beta)$.*

Definition 4.3.2 *Given the location of node a in the tree T_b, $(T_b)^a$ is defined as the sub-tree of T_b, containing all the ancestors of a in T_b and the node a itself.*

Let $T_c = (V_{T_c}, E_{T_c})$ and $T_d = (V_{T_d}, E_{T_d})$ be two trees constructed from G, and let $c_\delta, c_\rho \in V_{T_c}$ and $d_\eta, d_\zeta \in V_{T_d}$, where the indices $\delta, \rho, \eta, \zeta$ indicate the level in which each node is located.

Definition 4.3.3 *The pair (c_δ, d_η) is said to be younger than the pair (c_ρ, d_ζ) if*

$$\min(\delta, \eta) < \min(\rho, \zeta) \tag{4.27}$$

The algorithm for calculating cross covariance terms gradually processes pair permutations between nodes in $T_c = (V_{T_c}, E_{T_c})$ and nodes in $T_d = (V_{T_d}, E_{T_d})$ at different levels, starting from the first level. The permutation set of the kth level is denoted by $\mathcal{M}_k$, with $\mathcal{M}_1 \doteq \{(c, d)\}$. The next sections describe an algorithm for calculating $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ based on $\mathcal{M}_k$ from different levels. The value of $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is initialized to zero.

4.3.2.1 Processing a single member of $\mathcal{M}_k$

In the general case, when processing the permutation set $\mathcal{M}_k$ from level k, all the nodes on the path to the leaf (which is $c \in V_{T_c}$ and $d \in V_{T_d}$) should be considered, starting from the leaf and going up until reaching the current level k. For example, assume that for some member $(c_k, d_k) \in \mathcal{M}_k$, the paths to the leaf nodes are $c_k \overset{T_c}{\Longrightarrow} c$ and $d_k \overset{T_d}{\Longrightarrow} d$. Figure 4.4(a) schematically illustrates a general path $c_k \overset{T_c}{\Longrightarrow} c$. Start by checking whether (c_k, d) or (c, d_k) are known in the sense of Definition 4.3.1, i. e., whether $\odot(c_k, d)$ or $\odot(c, d_k)$. If not, then check whether $\odot(c_k, d_2)$ or $\odot(c_2, d_k)$, and so on. The procedure ends when a known pair of nodes is found, or when reaching and analyzing the pair (c_k, d_k). When a known couple of nodes is discovered, its contribution to the cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is calculated.

Denote the overall weight of the paths $c_k \overset{T_c}{\Longrightarrow} c$ and $d_k \overset{T_d}{\Longrightarrow} d$ by $W_c(c_k)$ and $W_d(d_k)$, respectively. If $\odot(c_j, d_k)$, with $1 \leq j \leq k$, then $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is updated according to:

$$E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T] \leftarrow E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T] + W_c(c_j) E[\tilde{\mathbf{X}}_{c_j} \tilde{\mathbf{X}}_{d_k}^T] W_d^T(d_k) + Q_{c_j d_k} \tag{4.28}$$

Similarly, if $\odot(c_k, d_j)$, with $1 \leq j \leq k$, then $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ is updated according to:

$$E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T] \leftarrow E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T] + W_c(c_k) E[\tilde{\mathbf{X}}_{c_k} \tilde{\mathbf{X}}_{d_j}^T] W_d^T(d_j) + Q_{c_k d_j} \tag{4.29}$$

The noise covariances $Q_{c_k d_j}$ and $Q_{c_j d_k}$ are analyzed in Section 4.3.2.3. If $w(a, b)$ is the arc weight connecting the node a to node b in G, then

$$W_c(c_k) = \Pi_{i=2}^k w(c_i, c_{i-1}) \tag{4.30}$$

$$W_d(d_k) = \Pi_{i=2}^k w(d_i, d_{i-1}) \tag{4.31}$$

After finishing analyzing the member $(c_k, d_k) \in \mathcal{M}_k$, the permutation set $\mathcal{M}_k$ is updated as follows.

$$\mathcal{M}_k \leftarrow \mathcal{M}_k \setminus \begin{cases} \{(c', d_k) \mid c' \in \pi_c(c_j)\ ,\ (c', d_k) \in \mathcal{M}_k\} & \text{if } \odot(c_j, d_k) \\ \{(c_k, d') \mid d' \in \pi_d(d_j)\ ,\ (c_k, d') \in \mathcal{M}_k\} & \text{if } \odot(c_k, d_j) \end{cases} \tag{4.32}$$

4.3.2.2 Calculation of $\mathcal{M}_{k+1}$

Having described how each level in the trees T_c and T_d is handled, the next step is to address the mechanism for advancing to the next level. After finishing processing all the members in $\mathcal{M}_k$, as discussed in Section 4.3.2.1, the only members left in $\mathcal{M}_k$ are those for whom the procedure did not find any known pair. If $\mathcal{M}_k = \phi$, the algorithm terminates.

The set of permutations in the next level, $\mathcal{M}_{k+1}$, is constructed based on the parents of each of the nodes that appear in $\mathcal{M}_k$: For each member $(a, b) \in \mathcal{M}_k$, the groups $\pi_c(a)$ and $\pi_d(b)$ are obtained. Then, a set of all the possible pair permutations between $\pi_c(a)$ and $\pi_d(b)$ is constructed and added to $\mathcal{M}_{k+1}$:

$$\mathcal{M}_{k+1} = \left\{ (c^s_{k+1}, d^t_{k+1}) \mid c^s_{k+1} \in \pi_c(a) \ , d^t_{k+1} \in \pi_d(b) \ , \ \forall (a, b) \in \mathcal{M}_k \right\} \tag{4.33}$$

where s and t distinguish between several parents.

4.3.2.3 Effect of Noise Terms

In this section, we discuss the effect of process and measurement noise terms on the cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$, when expressing $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ via $\tilde{\mathbf{X}}_{c_k}$ and $\tilde{\mathbf{X}}_{d_k}$.

Let $T_a = (V_{T_a}, E_{T_a})$ be a tree constructed for some node $a \in V$, and let $a_l, a_{l-1} \in V_{T_a}$ be some nodes from levels l and $l-1$, respectively. These nodes are connected either by a transition relation (4.22) or an update relation (4.23). In the first case, the two nodes belong to the same thread, while in the second case, the nodes may be from different threads.

Denote by $\boldsymbol{\eta}_{a_l:a_{l-1}}$ the noise related to expressing $\tilde{\mathbf{X}}_{a_{l-1}}$ via $\tilde{\mathbf{X}}_{a_l}$. Then $\boldsymbol{\eta}_{a_l:a_{l-1}}$ can be either process or measurement noise, depending on the relation type:

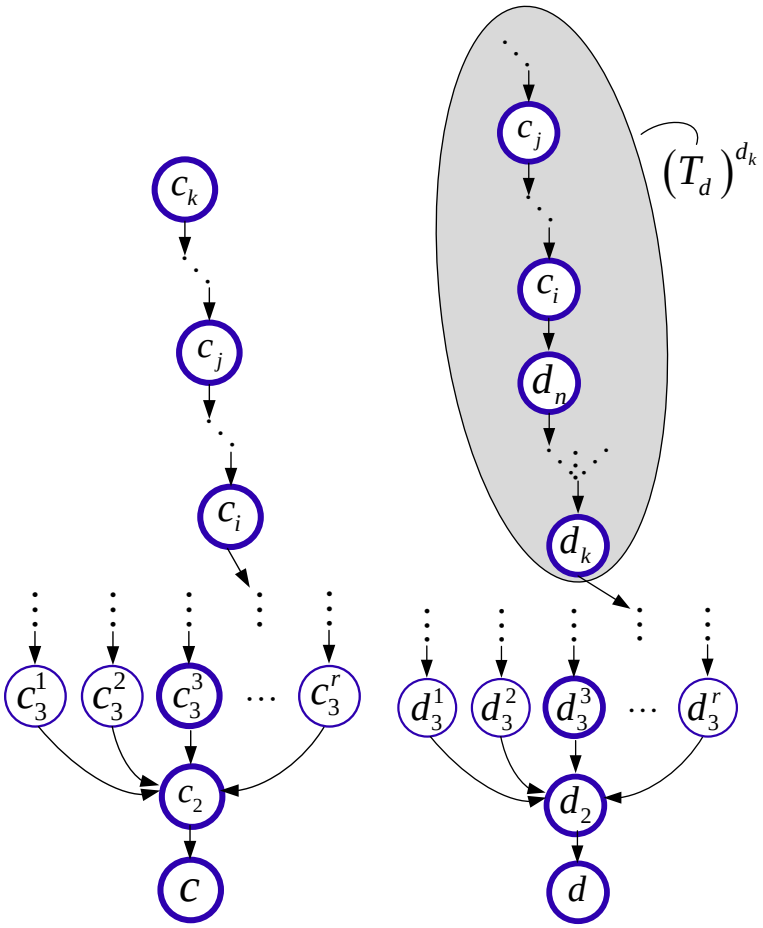
$$\boldsymbol{\eta}_{a_l:a_{l-1}} = \begin{cases} \boldsymbol{\omega}_{a_l \to a_{l-1}} & \text{transition relation} \\ -K_{a_{l-1}} D_{a_l} \mathbf{v}_{a_l} & \text{update relation} \end{cases} \tag{4.34}$$

Let c_m and d_p be some nodes in the trees T_c and T_d, respectively, and recall Definition 4.3.2.

Lemma 4.3.1 *If $(T_d)^{d_p}$ does not contain any nodes from the path $c_m \to \cdots \to c_r \to \cdots \to c$ in T_c, then $\boldsymbol{\eta}_{c_\gamma:c_{\gamma-1}}$ and $\tilde{\mathbf{X}}_{d_p}$ are statistically independent for any $\gamma \in \{1, \ldots, m\}$.*

Proof Suppose that $\boldsymbol{\eta}_{c_\gamma:c_{\gamma-1}}$ and $\tilde{\mathbf{X}}_{d_p}$ are statistically dependent for at least a single value of $\gamma \in \{1, \ldots, m\}$. Then there must exist some node c_r on the path $c_m \to \cdots \to c_r \to \cdots \to c$ in T_c, representing $\tilde{\mathbf{X}}_{c_r}$, such that $\tilde{\mathbf{X}}_{d_p}$ can be expressed in terms of $\tilde{\mathbf{X}}_{c_r}$, and perhaps other state vectors, i. e. $\tilde{\mathbf{X}}_{d_p}$ is a descendant of $\tilde{\mathbf{X}}_{c_r}$. Thus, c_r is an ancestor of d_p, and therefore will appear in $(T_d)^{d_p}$, thereby contradicting the assumption. ■

Corollary 4.3.1 *If T_d does not contain any nodes from the path $c_m \overset{T_c}{\Longrightarrow} c$, then $\boldsymbol{\eta}_{c_\gamma:c_{\gamma-1}}$ and $\tilde{\mathbf{X}}_d$ are statistically independent for any $\gamma \in \{1, \ldots, m\}$.*



(a) The tree T_c (b) The tree T_d

Figure 4.4: The node c_j in T_c has descendants that appear as ancestors of d_k in the subtree $(T_d)^{d_k}$, therefore contributing noise terms to the calculated $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$. Update-nodes are not explicitly marked.

Lemma 4.3.1 and Corollary 4.3.1 are also valid, with the proper adjustments, when considering $(T_c)^{c_m}$ and T_c, respectively.

At this point assume, without loss of generality, that in the process of analyzing the member (c_k, d_k), described in Section 4.3.2.1, the pair (c_j, d_k) was discovered as known in the sense of Definition 4.3.1. Since nodes from lower levels are analyzed first, no other known pair (c_r, d_k) or (c_k, d_r) exists with $r < j$.

Lemma 4.3.2 *The path $d_k \overset{T_d}{\Longrightarrow} d$ does not contain any node c_r from the path $c_j \to \cdots \to c_r \to \cdots \to c$ in T_c for any $1 \leq r < j$. If $r = j$, the node $c_r = c_j$ can only appear in the path $d_k \overset{T_d}{\Longrightarrow} d$ as d_k.*

Proof Suppose that the path $d_k \overset{T_d}{\Longrightarrow} d$ does contain a node c_r from the path $c_j \overset{T_c}{\Longrightarrow} c$,

with $1 \leq r < j$. Thus, there is a pair of nodes (a, b), with $a = c_r \in V_{T_c}$ and $b = c_r \in V_{T_d}$ such that $E[\tilde{\mathbf{X}}_a \tilde{\mathbf{X}}_b^T] \equiv E[\tilde{\mathbf{X}}_{c_r} \tilde{\mathbf{X}}_{c_r}^T]$ is known.[1]

However, since $r < j$, c_r is closer to c than c_j. Therefore, the pair (a, b) is younger, in the sense of Definition 4.3.3, than the pair (c_j, d_k), and thus should have been found while the algorithm processed the rth level. Consequently, this member would have been removed from the permutation set of the rth level, $\mathcal{M}_r$ (cf. Section 4.3.2.1). Hence, if such a pair indeed existed, then upon reaching the kth level, the permutation set $\mathcal{M}_k$ would not have contained the member (c_k, d_k), since $c_k \in \mathcal{A}_c(a) \equiv \mathcal{A}_c(c_r)$, and $d_k \in \mathcal{A}_d(b) \equiv \mathcal{A}_d(c_r)$ (cf. Eq. (4.33) for calculating $\mathcal{M}_k$). Since it is given that $(c_k, d_k) \in \mathcal{M}_k$, the node c_r does not exist.

Using the same reasoning, when $r = j$, the node $c_r = c_j$ cannot appear in the path $d_{k-1} \to \cdots \to d$. However, it is possible that $c_j = d_k$, since each node in G may have two children (and only one child in each of the trees). In this case, one of the children is located in T_c, while the other is located in T_d. ■

Lemmas 4.3.1 and 4.3.2 lead to the following corollary.

Corollary 4.3.2 *If $(T_d)^{d_k}$ does* not *contain any nodes from the path $c_j \overset{T_c}{\Longrightarrow} c$, then $\boldsymbol{\eta}_{c_\gamma : c_{\gamma-1}}$, for any $\gamma \in \{1, \ldots, j\}$, is statistically independent of all the states represented by the nodes $\{d_k \overset{T_d}{\Longrightarrow} d\} \cup (T_d)^{d_k}$.*

Note that $\boldsymbol{\eta}_{c_\gamma : c_{\gamma-1}}$ may still be statistically *dependent*, for at least a single value of $\gamma \in \{1, \ldots, j\}$, on states represented by the nodes in $T_d \setminus \{d_k \overset{T_d}{\Longrightarrow} d\} \setminus (T_d)^{d_k}$, if among these nodes there is at least one node from the path $c_j \overset{T_c}{\Longrightarrow} c$. This leads to the following corollary.

Corollary 4.3.3 *If for all the discovered pairs $\odot(a, b)$ with $a \in V_{T_c}$ and $b \in V_{T_d}$*

$$\mathcal{D}_c(a) \cap \mathcal{A}_d(b) = \phi \tag{4.35}$$

then all the noise terms from T_c, involved in the calculation of $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$, are statistically-independent of $\tilde{\mathbf{X}}_d$, and all the involved noise terms from T_d are statistically-independent of $\tilde{\mathbf{X}}_c$.

In other words, when the conditions of Corollary 4.3.3 are satisfied for all members in $\mathcal{M}_k$, for *all* considered k, the calculated cross-covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ will not contain any noise terms. However, when the conditions of Corollary 4.3.3 are not satisfied, $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ will contain noise covariances from different time instances and platforms. Returning to the discovered pair $\odot(c_j, d_k)$, we now assume that there are descendants of c_j in T_c that

[1] Recall that the covariance of each of the nodes in G is stored (cf. Section 4.2.3).

appear as ancestors of d_k in T_d: $\mathcal{D}_c(c_j) \cap \mathcal{A}_d(d_k) \neq \phi$. Consequently, $Q_{c_j d_k} \neq 0$ and, thus, the objective in the remainder of this section is to calculate $Q_{c_j d_k}$ (cf. Eq. (4.28)).

Among the nodes in $\mathcal{D}_c(c_j) \cap \mathcal{A}_d(d_k)$, denote by c_i, $1 < i < j$, the descendant of c_j that is closest to c, as illustrated in Figure 4.4. The child of c_i in T_d is denoted by d_n.

Lemma 4.3.3 *The path* $c_j \overset{T_c}{\Longrightarrow} c_i$ *appears in* $(T_d)^{d_k}$.

Proof $c_i \in \mathcal{A}_d(d_k)$, and therefore $c_i \in (T_d)^{d_k}$. Since c_i may be reached from any node on the path $c_j \to \cdots \to c_i$, and c_i leads to d_k, any node from $c_j \to \cdots \to c_i$ also leads to d_k. Therefore, $c_j \to \cdots \to c_i$ appears in $(T_d)^{d_k}$. ■

Observe that Lemma 4.3.3 is also valid for any sub-path $c_j \overset{T_c}{\Longrightarrow} c_{i'}$ of the path $c_j \overset{T_c}{\Longrightarrow} c_i$, with $i \leq i' < j$. Furthermore, $(T_d)^{d_k}$ might contain several appearances of the sub-paths $c_j \overset{T_c}{\Longrightarrow} c_{i'}$.

Now we analyze the correlation between the noise term $\boldsymbol{\eta}_{c_l:c_{l-1}}$, related to any two adjacent nodes c_l and c_{l-1} in the path $c_j \to \cdots \to c_l \to c_{l-1} \to \cdots \to c_i$, and $\tilde{\mathbf{X}}_{d_k}$. The term $E[\boldsymbol{\eta}_{c_l:c_{l-1}} \tilde{\mathbf{X}}_{d_k}^T]$, with $i+1 \leq l \leq j$, may be calculated as follows.

Assume for the moment that $(T_d)^{d_k}$ contains only a single appearance of $c_l \to c_{l-1}$. Then $\tilde{\mathbf{X}}_{d_k}$ is given by (cf. Figure 4.4)

$$\begin{aligned} \tilde{\mathbf{X}}_{d_k} &= W_{d_k}(c_l)\tilde{\mathbf{X}}_{c_l} + \sum_{r=k}^{n-1} W_{d_k}(d_r)\boldsymbol{\eta}_{d_{r+1}:d_r} + \\ &+ W_{d_k}(d_n)\boldsymbol{\eta}_{c_i:d_n} + \sum_{r=i}^{l-1} W_{d_k}(c_r)\boldsymbol{\eta}_{c_{r+1}:c_r} + \boldsymbol{\nu}_d \end{aligned} \tag{4.36}$$

where $\boldsymbol{\nu}_d$ is composed of state vectors and noise terms represented by nodes in $(T_d)^{d_k} \setminus \{c_l \to c_{l-1} \to \cdots \to d_k\}$. Here, $W_{d_k}(a)$ is the overall weight of the path $a \to \ldots \to d_k$ in $(T_d)^{d_k}$.
Since it was assumed that $c_l \to c_{l-1}$ appears only once in $(T_d)^{d_k}$, $(T_d)^{d_k} \setminus \{c_l \to c_{l-1} \to \cdots \to d_k\}$ does not contain $c_l \to c_{l-1}$. Therefore, according to Lemma 4.3.1, $\boldsymbol{\eta}_{c_l:c_{l-1}}$ and $\boldsymbol{\nu}_d$ are statistically independent and thus, from Eq. (4.36),

$$E[\boldsymbol{\eta}_{c_l:c_{l-1}} \tilde{\mathbf{X}}_{d_k}^T] = E[\boldsymbol{\eta}_{c_l:c_{l-1}} \boldsymbol{\eta}_{c_l:c_{l-1}}^T] W_{d_k}^T(c_{l-1}) \tag{4.37}$$

The term $E[\boldsymbol{\eta}_{c_l:c_{l-1}} \boldsymbol{\eta}_{c_l:c_{l-1}}^T]$ is equal to the process or measurement noise covariances, depending on the relation type between $\tilde{\mathbf{X}}_{c_{l-1}}$ and $\tilde{\mathbf{X}}_{c_l}$ (cf. Eq. (4.34)):

$$E[\boldsymbol{\eta}_{c_l:c_{l-1}} \boldsymbol{\eta}_{c_l:c_{l-1}}^T] = \begin{cases} Q_{c_l:c_{l-1}} & \text{transition relation} \\ K_{c_{l-1}} D_{c_l} R_{c_l:c_{l-1}} (K_{c_{l-1}} D_{c_l})^T & \text{update relation} \end{cases} \tag{4.38}$$

Recall that the matrices in Eq. (4.38) were stored as part of the arc weights (cf. Section 4.2).

In the general case, $(T_d)^{d_k}$ may contain several appearances of $c_l \to c_{l-1}$, each appearance with its own path $c_l \to c_{l-1} \to \cdots \to d_k$. Letting u distinguish between these different appearances of $c_l \to c_{l-1}$ in $(T_d)^{d_k}$, and denoting by $W_b^u(a)$ the overall weight of the uth path $a \overset{T_b}{\Longrightarrow} b$, Eq. (4.37) becomes:

$$E[\boldsymbol{\eta}_{c_l:c_{l-1}}\tilde{\mathbf{X}}_{d_k}^T] = E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T]\sum_u \left(W_{d_k}^u(c_{l-1})\right)^T \tag{4.39}$$

Furthermore, when considering the whole tree T_d, $c_l \to c_{l-1}$ may appear not only in $(T_d)^{d_k}$. According to Lemma 4.3.2, $c_l \to c_{l-1} \not\subset d_k \overset{T_d}{\Longrightarrow} d$. Thus, in addition to $(T_d)^{d_k}$, $c_l \to c_{l-1}$ may be also found only in $T_d \diagdown (T_d)^{d_k} \diagdown \{d_k \overset{T_d}{\Longrightarrow} d\}$. However, the contribution of the correlation between $\boldsymbol{\eta}_{c_l:c_{l-1}}$ and the state vectors represented by nodes in $T_d \diagdown (T_d)^{d_k} \diagdown \{d_k \overset{T_d}{\Longrightarrow} d\}$ will be calculated when processing other members in $\mathcal{M}_k$.
In a similar manner to Eq. (4.36), $\tilde{\mathbf{X}}_c$ can be expressed as (cf. Figure 4.4)

$$\tilde{\mathbf{X}}_c = W_c(c_j)\tilde{\mathbf{X}}_{c_j} + \sum_{r=i}^{l-1} W_c(c_r)\boldsymbol{\eta}_{c_{r+1}:c_r} + \boldsymbol{\nu}_c \tag{4.40}$$

where $\boldsymbol{\nu}_c$ is composed of state vectors and noise terms outside the path $c_j \overset{T_c}{\Longrightarrow} c_i \overset{T_c}{\Longrightarrow} c$. Therefore, the contribution of the noise term $\boldsymbol{\eta}_{c_l:c_{l-1}}$ to $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$, due to the nodes in $\mathcal{D}_c(c_j) \cap \mathcal{A}_d(d_k)$, is:

$$\bar{Q}_1(l) \doteq W_c(c_{l-1})E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T]\sum_u (W_d^u(c_{l-1}))^T \tag{4.41}$$

for *each* $i+1 \le l \le j$.

Yet, in addition to the above, the nodes $\mathcal{D}_d(d_k) \cap \mathcal{A}_c(c_j)$ also appear in expressions that constitute $Q_{c_jd_k}$. This situation may be handled in a similar manner. Among all the nodes in $\mathcal{D}_d(d_k) \cap \mathcal{A}_c(c_j)$, denote by d_s, $1 < s < k$, the node that is closest to d. Thus, the contribution of noise terms to $E[\tilde{\mathbf{X}}_c\tilde{\mathbf{X}}_d^T]$, due to the nodes $\mathcal{D}_d(d_k) \cap \mathcal{A}_c(c_j)$, is:

$$\bar{Q}_2(m) \doteq \sum_u W_c^u(d_{m-1})E[\boldsymbol{\eta}_{d_m:d_{m-1}}\boldsymbol{\eta}_{d_m:d_{m-1}}^T]W_d^T(d_{m-1}) \tag{4.42}$$

for *each* $s+1 \le m \le k$.

In conclusion, the noise covariance $Q_{c_jd_k}$ for a discovered $\odot(c_j, d_k)$ is:

$$Q_{c_jd_k} \doteq \sum_{l=i+1}^{j} \bar{Q}_1(l) + \sum_{m=s+1}^{k} \bar{Q}_2(m) \tag{4.43}$$

In practice, the calculation of $Q_{c_jd_k}$ requires processing all the nodes in $(T_d)^{d_k}$, checking if they appear in $c_j \overset{T_c}{\Longrightarrow} c$, and processing all the nodes in $(T_c)^{c_j}$, checking if these nodes

appear in $d_k \overset{T_d}{\Longrightarrow} d$. If such nodes were found, the contribution of the involved noise terms is computed using Eq. (4.43). A similar process should be carried out for calculating $Q_{c_k d_j}$ in case $\odot(c_k, d_j)$ is discovered (cf. Eq. (4.29)).

The above calculations are required only upon discovering a known pair. A formal algorithm for calculating $Q_{c^* d^*}$ for some discovered pair $\odot(c^*, d^*)$ is given in the next section.

4.3.3 Formal Algorithms

Algorithm 2 summarizes the developed approach for calculating the cross covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$ given the trees T_c and T_d. The notation $\text{card}(A)$ denotes the cardinality of the set A.

The process of analyzing a single permutation (c_k, d_k) from $\mathcal{M}_k$, discussed in Section 4.3.2.1, is presented in Algorithm 3, while Algorithm 4 implements the technique, developed in Section 4.3.2.3, for calculating the effect of the noise terms on the calculated cross covariance $E[\tilde{\mathbf{X}}_c \tilde{\mathbf{X}}_d^T]$.

Algorithm 2 Calculation of $E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T]$

Input: Trees T_c, T_d. $h_c \doteq \text{height}(T_c), h_d \doteq \text{height}(T_d)$
Initialization: $k = 1$, $E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T] = 0$, $\mathcal{M}_1 = \{(c, d)\}$.
while $k \leq \max(h_c, h_d)$ **do**
  for $r = 1$ to $\text{card}(\mathcal{M}_k)$ **do**
    Let $(c_k, d_k) \doteq \mathcal{M}(r)$. Execute Algorithm 3 on (c_k, d_k). Let the output be $c^*, d^*, W_{c^* d^*}, \text{flag}$.
    if flag **then**
      $E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T] = E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T] + W_{c^* d^*}$
      Update $\mathcal{M}_k$ according to Eq. (4.32)
  if $\mathcal{M}_k$ is empty **then**
    return $E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T]$
  else
    Construct $\mathcal{M}_{k+1}$ based on Eq. (4.33)
    $k = k + 1$
return $E[(\tilde{\mathbf{X}}_c)(\tilde{\mathbf{X}}_d)^T]$

Algorithm 3 Processing a single member (c_k, d_k) from $\mathcal{M}_k$

Input: Trees T_c, T_d, node c_k in T_c and node d_k in T_d
Initialization: $l = 1$, $c^* = d^* = W_{c^*d^*} = \{\}$, flag $= 0$
while $l \leq k$ **do**
  if $E[\tilde{\mathbf{X}}_{c_k}\tilde{\mathbf{X}}_{d_l}^T]$ is known, i. e., $\odot(c_k, d_l)$ **then**
    $c^* \doteq c_k$, $d^* \doteq d_l$, flag $= 1$
    break
  if $E[\tilde{\mathbf{X}}_{c_l}\tilde{\mathbf{X}}_{d_k}^T]$ is known, i. e., $\odot(c_l, d_k)$ **then**
    $c^* \doteq c_l$, $d^* \doteq d_k$, flag $= 1$
    break
  $l = l + 1$
if flag **then**
  Calculate $Q_{c^*d^*}$ by executing Algorithm 4
  $W_{c^*d^*} = W_c(c^*)E[\tilde{\mathbf{X}}_{c^*}\tilde{\mathbf{X}}_{d^*}^T]W_d^T(d^*) + Q_{c^*d^*}$
return $c^*, d^*, W_{c^*d^*}$, flag

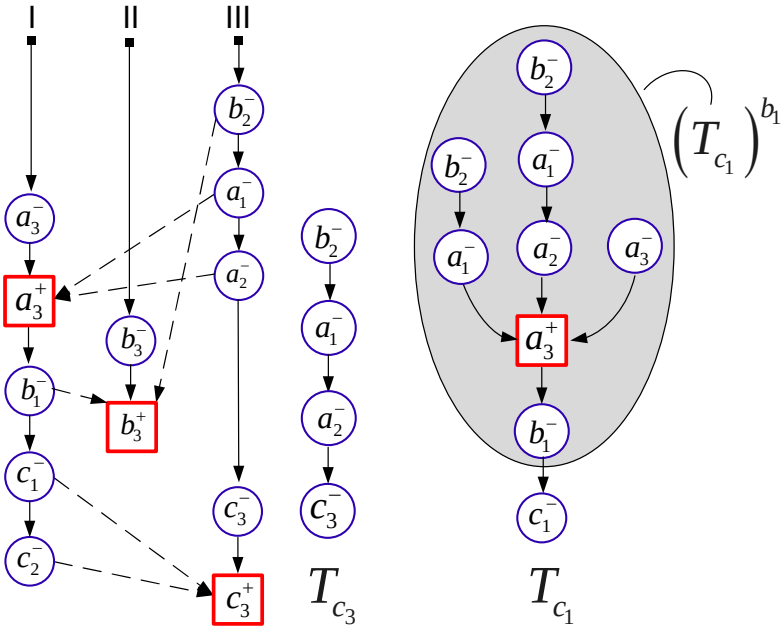
Algorithm 4 Calculation of $Q_{c^*d^*}$.

Input: T_c, T_d, c^*, d^*, s.t. $E[\tilde{\mathbf{X}}_{c^*}\tilde{\mathbf{X}}_{d^*}^T]$ is known.
Initialization: $U_{d^*} \doteq (T_d)^{d^*}$, $U_{c^*} \doteq (T_c)^{c^*}$, $Q_{c^*d^*} = 0$.
while U_{d^*} is not empty **do**
  $U_{d^*} = U_{d^*} \setminus \{l_i\}$, where $\{l_i\}$ are the leafs of U_{d^*}.
  Check if any leafs of U_{d^*} appear in $c^* \overset{T_c}{\Longrightarrow} c$.
  for each such leaf β of U_{d^*} **do**
    Denote $c^* \rightarrow \cdots \rightarrow \beta$ as $u_s \rightarrow \cdots \rightarrow u_1$, then $E[\boldsymbol{\eta}_{c^*\rightarrow\beta}\boldsymbol{\eta}_{c^*\rightarrow\beta}^T] = \sum_{\zeta=2}^{s} W_\beta(u_{\zeta-1})E[\boldsymbol{\eta}_{\zeta\rightarrow\zeta-1}\boldsymbol{\eta}_{\zeta\rightarrow\zeta-1}^T]\left(W_\beta(u_{\zeta-1})\right)^T$
    $Q_{c^*d^*} = Q_{c^*d^*} + W_c(\beta)E[\boldsymbol{\eta}_{c^*\rightarrow\beta}\boldsymbol{\eta}_{c^*\rightarrow\beta}^T]W_d^T(\beta)$
    $U_{d^*} = U_{d^*} \setminus (T_d)^\beta$
Repeat Steps 3-9, replacing: U_{d^*} by U_{c^*}; c^* by d^*; c by d; T_c by T_d; instead of Step 8 perform $Q_{c^*d^*} = Q_{c^*d^*} + W_c(\beta)E[\boldsymbol{\eta}_{d^*\rightarrow\beta}\boldsymbol{\eta}_{d^*\rightarrow\beta}^T]W_d^T(\beta)$.
return $Q_{c^*d^*}$

4.3.4 Example

In this example, the proposed method is demonstrated for an MP measurement model comprised of information obtained from three different platforms, i. e. $r = 3$. Such a measurement model was already considered in Section 4.2.1. The residual measurement $\mathbf{z}$ is given by Eq. (4.11). As will be seen in Chapter 5, this MP measurement model represents vision-based three-view MP updates. Further results involving the developed method for calculating cross-covariance terms, are therefore provided in Chapter 5.

Consider the problem of calculating the term $E[\tilde{\mathbf{X}}^-_{c_3}(\tilde{\mathbf{X}}^-_{c_1})^T]$ in the example shown in Figure 4.5(a). The trees $T_{c_3^-}$ and $T_{c_1^-}$ are shown in Figure 4.5(b). In this example, $E[\tilde{\mathbf{X}}^-_{c_3}(\tilde{\mathbf{X}}^-_{c_1})^T]$ can be calculated based on the known term $E[\tilde{\mathbf{X}}^-_{b_2}(\tilde{\mathbf{X}}^-_{b_1})^T]$, which is analyzed upon reaching the fourth level in the two trees. As can be seen, $a_1^-, a_2^- \in \mathcal{D}_{c_3^-}(b_2^-)$ and also $a_1^-, a_2^- \in \mathcal{A}_{c_1^-}(b_1^-)$. Thus, according to Section 4.3.2.3, the noise terms associated with the path $b_2^- \rightarrow a_1^- \rightarrow a_2^-$ are *not* statistically independent of $\tilde{\mathbf{X}}^-_{b_1}$.



(a) Graph (b) The trees T_{c_3} and T_{c_1}

Figure 4.5: An example assuming three-view measurements.

Applying the proposed algorithm, the term $E[\tilde{\mathbf{X}}^-_{c_3}(\tilde{\mathbf{X}}^-_{c_1})^T]$ is calculated as

$$\begin{aligned} E[\tilde{\mathbf{X}}^-_{c_3}(\tilde{\mathbf{X}}^-_{c_1})^T] &= \Phi^{III}_{b_2\rightarrow c_3} E[\tilde{\mathbf{X}}^-_{b_2}(\tilde{\mathbf{X}}^-_{b_1})^T](\Phi^{I}_{b_1\rightarrow c_1})^T + \Phi^{III}_{a_2\rightarrow c_3} Q^{I}_{a_1\rightarrow a_2} A_2^T (\Phi^{I}_{a_3\rightarrow c_1})^T + \\ &+ \Phi^{III}_{a_1\rightarrow c_3} Q^{III}_{b_2\rightarrow a_1} (A_1 + A_2\Phi^{III}_{a_1\rightarrow a_2})^T (\Phi^{I}_{a_3\rightarrow c_1})^T \end{aligned} \tag{4.44}$$

with $A_1 = -K_{a_3} H_{a_1}$ and $A_2 = -K_{a_3} H_{a_2}$.

4.3.5 Computational Complexity

As seen in Section 4.2, the computational complexity depends on the particular scenario being considered. In Appendix C, an analysis of the computational complexity is provided. It is shown that the worst-case computational complexity is bounded by $O\left(n^2 log(rn)\right)$, where n is the number of the performed MP measurement updates, represented in G. Appendix C (Section C.2) also suggests an efficient implementation method, which allows to considerably reduce the actual computational complexity.

If a platform has limited computational resources, it is possible to approximate the true cross-covariance terms by maintaining a limited history of the MP measurement updates. In this case, the graph G may be treated as a constant-size buffer, where upon reaching a maximum size, the nodes representing information contained in old MP measurement updates[2] are removed from the graph G, thereby neglecting the contribution of those updates on the cross-covariance terms to be computed in the future.

It is worth noting that in practice, specific scenarios exist in which the worst-case computational complexity is significantly less. One example is the scenario considered in Figure 4.1, which requires processing only 3 levels in each tree, assuming the efficient implementation discussed in Appendix C.

Appendix C (Section C.3) also suggests an efficient method for calculating the transition and process noise covariance matrices, required by the developed method in this chapter.

4.3.6 Incorporating Other Measurements

The proposed technique for calculating cross-covariance terms can be also applied when, in additional to the MP measurement updates, other measurements should be incorporated as well. These measurements can be produced by additional sensors, that the platforms are equipped with and using additional available information (e.g. DTM). For instance, each platform can apply epipolar-geometry constraints based on images captured by its own camera.

For simplicity, a standard measurement model is assumed for these additional measurement types:

$$\mathbf{z} = H\mathbf{X} + \mathbf{v} \tag{4.45}$$

This measurement model will be termed in this section as *basic* measurement model. Next, it is shown how the basic measurement model can be incorporated with the developed approach for calculating the cross-covariance terms.

For simplicity, the concept is demonstrated for the three-platform scenario, considered in Section 4.2.1, i. e. $r = 3$. Refer to the simple scenario shown in Figure 4.1, and assume a single basic measurement update was performed by platform III between the first update

[2]Different logic may be applied for choosing the nodes to be removed from the graph.

event, at a_3, and the second update event, at b_3. Denote by t_γ the time instant of this additional update, $t_\gamma \in (t_{a_3}, t_{b_3})$. The a posteriori estimation error[3] $\tilde{\mathbf{X}}_\gamma^+$ is given, due to Eq. (4.45), by

$$\tilde{\mathbf{X}}_\gamma^+ = (I - K_\gamma H_\gamma)\tilde{\mathbf{X}}_\gamma^- - K_\gamma \mathbf{v}_\gamma \tag{4.46}$$

where K_γ and H_γ are the Kalman gain and measurement matrices, respectively, computed for the basic measurement model (4.45) at t_γ.

Consequently, $\tilde{\mathbf{X}}_{b_3}^-$ is no longer inertially propagated from $\tilde{\mathbf{X}}_{a_3}^+$, but instead can be expressed as

$$\tilde{\mathbf{X}}_{b_3}^- = \phi_{\gamma\to b_3}\tilde{\mathbf{X}}_\gamma^+ + \boldsymbol{\omega}_{\gamma\to b_3} \tag{4.47}$$

Based on Eq. (4.46), $\tilde{\mathbf{X}}_{b_3}^-$ can be expressed as

$$\phi_{\gamma\to b_3}\left[(I - K_\gamma H_\gamma)\left(\phi_{a_3\to\gamma}\tilde{\mathbf{X}}_{a_3}^+ + \boldsymbol{\omega}_{a_3\to\gamma}\right) - K_\gamma \mathbf{v}_\gamma\right] + \boldsymbol{\omega}_{\gamma\to b_3} \tag{4.48}$$

or, alternatively:

$$\tilde{\mathbf{X}}_{b_3}^- = \phi^*_{a_3\to b_3}\tilde{\mathbf{X}}_{a_3}^+ + \boldsymbol{\omega}^*_{a_3\to b_3} \tag{4.49}$$

where

$$\phi^*_{a_3\to b_3} \doteq \phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\phi_{a_3\to\gamma} \tag{4.50}$$

is the equivalent transition matrix and

$$\boldsymbol{\omega}^*_{a_3\to b_3} \doteq \phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\boldsymbol{\omega}_{a_3\to\gamma} - \phi_{\gamma\to b_3}K_\gamma \mathbf{v}_\gamma + \boldsymbol{\omega}_{\gamma\to b_3} \tag{4.51}$$

is the equivalent noise term with noise covariance $Q^*_{a_3:b_3}$ given by

$$Q^*_{a_3:b_3} = \phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)Q_{a_3:\gamma}\left[\phi_{\gamma\to b_3}(I - K_\gamma H_\gamma)\right]^T + \phi_{\gamma\to b_3}K_\gamma R K_\gamma^T \phi^T_{\gamma\to b_3} + Q_{\gamma:b_3} \tag{4.52}$$

where $R \doteq E[\mathbf{v}_\gamma \mathbf{v}_\gamma^T]$.

Thus, for example, $P^-_{b_3b_2}$ is given by (cf. Eq. (4.19)):

$$P^-_{b_3b_2} = \Phi^*_{a_3\to b_3}\left\{(I - K_{a_3}H_{a_3})\,P^-_{a_3a_2} - K_{a_3}H_{a_2}P^-_{a_2a_2} - K_{a_3}H_{a_1}P^-_{a_1a_2}\right\}\Phi^T_{a_2\to b_2}$$

In the general case, there might be a number of basic updates in each of the platforms. However, these updates are treated in a similar manner, by calculating the equivalent transition matrix Φ^* and noise covariance matrix Q^* between the time instances that participate in the MP measurement. The repositories maintained by the platforms (cf. Section C.3 in Appendix C) should be also accordingly updated.

[3]Explicit identities of the involved platforms are not indicated.

4.4 Conclusions

In this chapter, a new method was proposed for on-demand, explicit calculation of correlation terms, required for consistent extended Kalman filter-based data fusion in distributed cooperative navigation. The method assumed a general multi-platform model, involving navigation information and readings of onboard sensors of any number of platforms, possibly obtained at different time instances.

Each platform in the group maintained a state vector comprised only of its own navigation parameters, while the required correlation terms with other platforms were calculated based on a graph, representing all the multi-platform measurement updates performed thus far. This graph was locally maintained by every platform in the group. The developed method is capable of handling the most general scenarios of multi-platform measurements by properly taking into account the involved process and measurement noise terms.

The proposed method was demonstrated in a synthetic example in which the multi-platform measurement is constituted upon information obtained from three different platforms. Such a measurement model is used in Chapter 5, in which the three-view navigation aiding method, that was developed in Chapter 3 for a single platform, is extended to cooperative navigation. Additional discussion and results regarding the proposed method for calculating cross-covariance terms are given in Section 5.5 and in Appendix C. In particular, it is shown that applying the proposed method for calculating the correlation terms allows to obtain consistent and unbiased estimation, which becomes biased and inconsistent when these terms are neglected.

Chapter 5

Distributed Vision-Aided Cooperative Navigation based on Three-View Geometry

Contents

This chapter presents a new method for vision-aided cooperative navigation, which is based on three-view geometry. This result is an extension of the method developed in Chapter 3 for cooperative navigation of multiple autonomous platforms. A group of cooperative platforms is assumed, each platform is equipped with a standard inertial navigation system and an on-board, possibly gimbaled, camera. The platforms are also assumed to be capable of intercommunicating.

In contrast to the common approach for cooperative localization that is based on relative pose measurements between pairs of platforms (cf. Section 1.1.5), the method developed in this chapter formulates a measurement whenever the same scene is observed by different platforms. The camera, therefore, is no longer required to be aimed towards other platforms. Each measurement is constituted upon identifying three images with a common overlapping area. These images can be captured by different platforms, not necessarily at the same time. The three-view constraints, developed in Chapter 3, are reformulated into a measurement, which is then used for performing navigation aiding.

A similar concept has been already proposed in Refs. [44] and [61] regarding two-view measurements between pairs of platforms. However, in contrast to these works, application of the three-view geometry constraints allows to reduce position and velocity errors in all axes without assuming a range sensor (cf. also Chapter 3). This is not possible with relative pose measurements and two-view measurements.

The three-view measurement is a function of imagery and navigation information belonging to different platforms. In the general case, these different sources of information can be statistically dependent. Ignoring this dependence can result in inconsistent and overconfident estimation [62]. In this chapter, it is proposed to explicitly calculate the correlation terms required in the Kalman-based information fusion phase. This is performed by adjusting the approach, developed in Chapter 4 for a general multi-platform measurement model, to the specific three-view measurement model considered herein. Consequently, only platforms that contribute their current image and navigation solution to the three-view measurement can be actually updated (as discussed in Chapter 4). In contrast to [44], explicit calculation of the cross-covariance terms eliminates the need in the smoothing phase each time a new measurement is to be executed.

5.1 Method Overview

Figure 5.1 shows the overall concept of the proposed method for multi-platform vision-aided navigation. The proposed method assumes a group of cooperative platforms capable of inter-communication. Each platform is equipped with a standard inertial navigation system and an onboard camera, which may be gimbaled. Some, or all, of the platforms maintain a local repository comprised of images captured along the mission. These images are attached with navigation data when they are captured. The INS is comprised of an inertial measurement unit whose measurements are integrated into a navigation solution.

In a typical scenario, a platform captures an image and broadcasts it, along with its current navigation solution, to other platforms in the group, inquiring if they have previously captured images containing the same region. Upon receiving such a query, each platform performs a check in its repository looking for appropriate images. Among these images, only images with a smaller navigation uncertainty compared to the uncertainty in the navigation data of the query image, are transmitted back. Platforms that do not

maintain a repository, perform the check only on the currently-captured image.

The process of the querying platform is schematically described in Figure 5.1. After receiving the images and the attached navigation data from other platforms in the group, two best images are chosen and, together with the querying image, are used for formulating the three-view constraints (Section 5.2). These constraints are then transformed into a measurement and are used for updating the navigation system of the querying platform, as described in Section 5.3. Since the navigation data attached to the chosen three images can be correlated, a graph-based method is applied for calculating the required cross-covariance terms for the fusion process. This method was developed in Chapter 4 for a general MP measurement model. Details regarding the specific implementation of this method for the considered three-view MP measurement model are given in Section 5.3.2. The overall protocol for information sharing among the platforms in the group is discussed in Section 5.4.

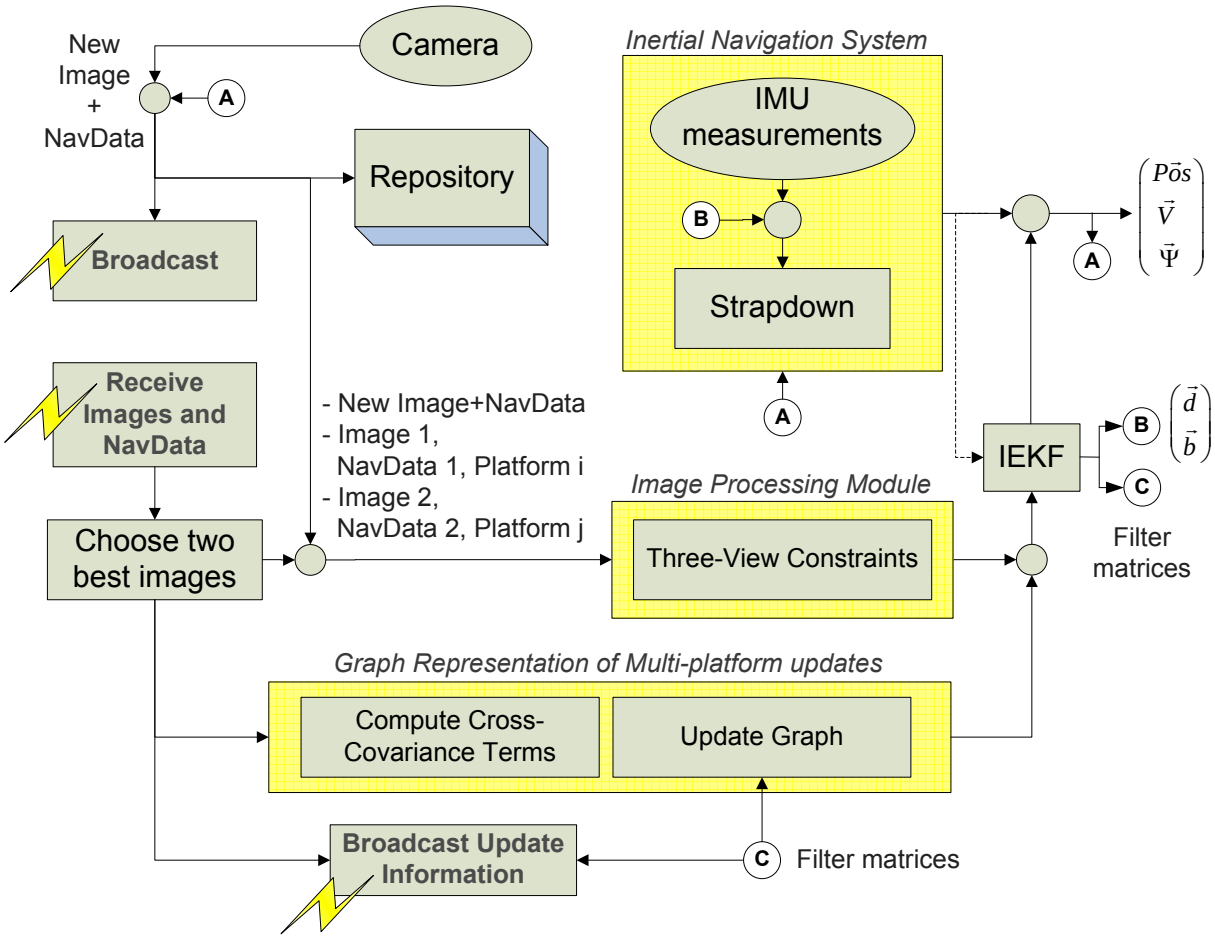


Figure 5.1: Multi-platform navigation aiding - querying platform scheme

5.2 Three-View Geometry Constraints

Assume some general scene is observed from three different views, captured by different platforms. Figure 5.2 depicts such a scenario, in which a static landmark $\mathbf{p}$ is observed in the three images I_1, I_2 and I_3. The image I_3 is the currently-captured image of the third platform, while I_1 and I_2 are two images captured by the first two platforms. These two images can be the currently-captured images of these platforms, but they could also be captured in the past and stored in the repository of each platform, as is indeed illustrated in the figure. Alternatively, I_1 and I_2 could also be captured by the same platform.

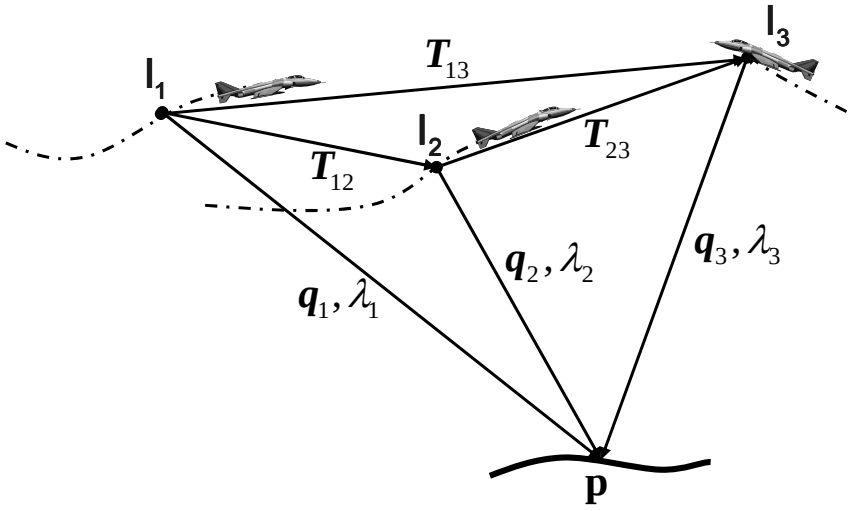


Figure 5.2: Three-view geometry: a static landmark $\mathbf{p}$ observed by three different platforms. Images I_1 and I_2 were captured by the first two platforms in the past, while image I_3, is the currently-acquired image by the third platform.

The notations in Figure 5.2 are similar to those defined for a single platform (cf. Figure 3.2). The constraints resulting from observing the same landmark from three different views are given by Theorem 3.2.1 in Section 3.2:

$$\mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) = 0 \tag{5.1a}$$

$$\mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) = 0 \tag{5.1b}$$

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \tag{5.1c}$$

All the parameters in Eqs. (5.1) should be expressed in the same coordinate system using the appropriate rotation matrices taken from navigation systems of the involved platforms. It is assumed that this coordinate system is the LLLN system of the platform that captured the second image at t_2.

Taking into account that, in practice, multiple matching features will be obtained, we

use the formulation derived while considering a single platform:

$$\begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N\times 3} \mathbf{T}_{23} = \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N\times 3} \mathbf{T}_{12} \tag{5.2}$$

where N and the matrices U, F, G and W are defined in Section 3.2.1.

5.3 Three-View-Based Navigation Update

Define the state vector of *each* platform to be its own navigation errors and IMU error parameterization, as given by Eq. (1.10):

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\boldsymbol{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T$$

The process equation for the ith platform is given by

$$\mathbf{X}_i(t_b) = \Phi^i_{t_a\to t_b}\mathbf{X}_i(t_a) + \boldsymbol{\omega}^i_{t_a\to t_b} \tag{5.3}$$

where the transition matrix $\Phi^i_{t_a\to t_b}$ and the discrete process noise $\boldsymbol{\omega}_{t_a\to t_b}$ are discussed in Section 1.3.2.

When real navigation and imagery data are considered, the constrains in Eq. (5.2) will not be satisfied. In a similar manner to Section 3.3, a residual measurement $\mathbf{z}$ is defined:

$$\mathbf{z} \doteq \begin{bmatrix} U \\ F \\ 0 \end{bmatrix} \mathbf{T}_{23} - \begin{bmatrix} W \\ 0 \\ G \end{bmatrix} \mathbf{T}_{12}$$

which is a nonlinear function of the position and attitude of the involved platforms, and of the LOS vectors from the three views. The involved information can be taken, in the general case, from different time instances. Note, that the identity of the involved platforms, *and* the time instances are unknown a priori. Denoting the identity of the involved platforms in the current measurement by the indices $1, 2$ and 3, and the time instances by t_1, t_2 and t_3, the residual measurement is given by:

$$\mathbf{z}(t) = \mathbf{h}\left(\mathbf{Pos}_3(t_3), \boldsymbol{\Psi}_3(t_3), \mathbf{Pos}_2(t_2), \boldsymbol{\Psi}_2(t_2), \mathbf{Pos}_1(t_1), \boldsymbol{\Psi}_1(t_1), \left\{\mathbf{q}^{C_1}_{1_i}, \mathbf{q}^{C_2}_{2_i}, \mathbf{q}^{C_3}_{3_i}\right\}\right) \tag{5.4}$$

Linearizing Eq. (5.4) we obtain, similarly to Eq. (3.23):

$$\mathbf{z} \approx H_3(t_3)\mathbf{X}_3(t_3) + H_2(t_2)\mathbf{X}_2(t_2) + H_1(t_1)\mathbf{X}_1(t_1) + D\mathbf{v} \tag{5.5}$$

where the Jacobian matrices H_3, H_2, H_1 and D in the above equation are defined as

$$H_3 \doteq \nabla_{\boldsymbol{\zeta}_3(t_3)}\mathbf{h} \quad , \quad H_2 \doteq \nabla_{\boldsymbol{\zeta}_2(t_2)}\mathbf{h} \quad , \quad H_1 \doteq \nabla_{\boldsymbol{\zeta}_1(t_1)}\mathbf{h} \quad , \quad D \doteq \nabla_{\left\{\mathbf{q}^{C_1}_{1_i}, \mathbf{q}^{C_2}_{2_i}, \mathbf{q}^{C_3}_{3_i}\right\}}\mathbf{h} \tag{5.6}$$

with $\boldsymbol{\zeta}_i(t_i)$ being the navigation solution and IMU errors parametrization (cf. Sections 1.3.2 and 3.3), and $\mathbf{v}$ is the image noise associated with the LOS vectors appearing in Eq. (5.4).

From now on, we will use the notation $\mathbf{X}_i$ to denote $\mathbf{X}_i(t_i)$. Thus, $\mathbf{X}_i$ is the state vector of the appropriate platform at the capture-time of the ith image, as defined by Eq. (5.3). This state vector models the errors in the navigation data attached to this image.

As can be seen, the residual measurement is a function of all the three state vectors, which in the general case can be correlated. Assuming the correlation terms relating the three state vectors are known, all the involved platforms in the measurement can be updated by applying standard equations of the IEKF, as detailed in Section 5.3.1[1].

However, the calculation of the correlation terms is not trivial for the considered three-view measurement model. The difficulty comes from the fact that it is a priori unknown which platforms and what time instances will participate in a three-view measurement.

The common approach for calculating the cross-covariance terms in CN is to use an augmented covariance matrix, which contains cross-covariance terms between all the possible pairs of platforms in the group. As discussed in Chapter 4, this is indeed an approach used in some works (e. g. [48]), in which the measurement is a function of the navigation parameters at the *current* time of several platforms (as in relative pose measurements between pairs of platforms). Assuming M platforms in the group, and an $m \times m$ covariance matrix P_i for each platform i, the total covariance matrix of the group, containing also all the cross-covariance terms among platforms in the group is an $Mm \times Mm$ matrix

$$\mathcal{P}_{\text{Total}} = \begin{bmatrix} P_1 & P_{12} & \cdots & P_{1M} \\ P_{21} & P_2 & \cdots & P_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ P_{M1} & P_{M2} & \cdots & P_M \end{bmatrix}$$

where $P_i = E[\tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i^T]$ and $P_{ij} = E[\tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_j^T]$. The matrix $\mathcal{P}_{\text{Total}}$ can be efficiently calculated in a distributed manner (i. e. calculated by every platform in the group) [48].

Yet, the measurement model in Eq. (5.5) involves data from different platforms and from different, unknown time instances. Maintaining a total covariance matrix $\mathcal{P}_{\text{Total}}$ containing a covariance for every platform and cross-covariance terms between each pair of platforms in the group for any two time instances along the mission duration is not practical. Thus, an alternative technique should be used.

As a solution to the aforementioned problem, it is proposed to explicitly calculate the required cross-covariance terms based on an approach developed in Chapter 4 for a general MP measurement model. This approach represents the MP updates in a directed acyclic

[1] As shall be seen in the sequel, the actual update equations are, in the general case, different (cf. Eqs. (5.7)-(5.10)).

graph, locally maintained by every platform in the group. The required cross-covariance terms are computed based on this graph representation, going back and forth in the time domain according to the history of the so-far performed MP updates.

Another possible approach for consistent data fusion has been recently developed in [44], considering relative pose and two-view measurements between pairs of platforms. However, in this approach a smoothing over the navigation solutions of all the platforms is performed whenever any kind of measurement is received. In contrast to this, the approach proposed herein allows on-demand calculation of the required cross-covariance terms, without refining the navigation history, thereby being computationally efficient.

The graph needs to be acyclic, since otherwise, a measurement might trigger recursive updates in past measurements. In a general scenario involving three-view measurements among different platforms at different time instances, the graph is guaranteed to be acyclic, particularly if only platforms that contributed their *current* (and not past) image and navigation data are updated (see a further discussion in Section 4.2.2, page 101). For simplicity, we consider only one such platform, which is the querying platform, i. e. the platform that broadcasted the query image to the rest of the platforms in the group. Moreover, without loss of generality, it is assumed that the querying platform captures the *third* image, as illustrated in Figure 5.2. Thus, referring to Eq. (5.5), only $\mathbf{X}_3(t_3)$ is estimated, while $\mathbf{X}_2(t_2)$ and $\mathbf{X}_1(t_1)$ are modeled as random parameters.

An IEKF is applied, whereby the Kalman gain matrix is computed as (cf. Section 3.3)

$$K_3 = P_{\mathbf{X}_3\mathbf{z}} P_{\mathbf{z}}^{-1} \tag{5.7}$$

where

$$P_{\mathbf{X}_3\mathbf{z}} = P_3^- H_3^T + P_{32}^- H_2^T + P_{31}^- H_1^T \tag{5.8}$$

$$P_{\mathbf{z}} = H_3 P_3^- H_3^T + \begin{bmatrix} H_2 & H_1 \end{bmatrix} \begin{bmatrix} P_2 & P_{21} \\ P_{21}^T & P_1 \end{bmatrix} \begin{bmatrix} H_2 & H_1 \end{bmatrix}^T + DRD^T \tag{5.9}$$

The update equations are the IEKF standard equations. In particular, the a posteriori estimation error of the querying platform is given by:

$$\tilde{\mathbf{X}}_3^+ = [I - K_3 H_3]\, \tilde{\mathbf{X}}_3^- - K_3 H_2 \tilde{\mathbf{X}}_2^- - K_3 H_1 \tilde{\mathbf{X}}_1^- - K_3 D\mathbf{v} \tag{5.10}$$

where $\tilde{\mathbf{X}}$ denotes the estimation error of $\mathbf{X}$.

It should be noted that the remark regarding the ad-hoc approach in the above calculation of the Kalman gain, which appears in Section 3.3 (page 80) in the context of a single platform, is relevant in the current case as well.

As can be seen from Eqs. (5.8) and (5.9), the cross-covariance terms P_{21}, P_{31} and P_{32} indeed participate in the update process, and therefore these terms need to be calculated. Additional discussion regarding calculation of these terms using the approach developed in Chapter 4 is given in Section 5.3.2.

It is worth mentioning that there are specific cases, in which *all* the platforms participating in the measurement can be updated, since it is guaranteed that the graph will always be acyclic. In these cases, the filter formulation changes as described next. An example of such a scenario is given in Section 5.5.2.

5.3.1 All the Involved Platforms are Updated

The following augmented state vector is defined:

$$\boldsymbol{\mathcal{X}} \doteq \begin{bmatrix}\mathbf{X}_3^T & \mathbf{X}_2^T & \mathbf{X}_1^T\end{bmatrix}^T \tag{5.11}$$

with an augmented covariance matrix $\mathcal{P} \doteq E[\tilde{\boldsymbol{\mathcal{X}}}\tilde{\boldsymbol{\mathcal{X}}}^T]$, where $\tilde{\mathbf{a}}$ denotes the estimation error of $\mathbf{a}$. Let also $\mathcal{H} \doteq \begin{bmatrix}H_3 & H_2 & H_1\end{bmatrix}$ and $\mathcal{K} \doteq \begin{bmatrix}\breve{K}_3^T & \breve{K}_2^T & \breve{K}_1^T\end{bmatrix}^T$. The augmented Kalman gain matrix $\mathcal{K}$ is computed as

$$\mathcal{K} = \mathcal{P}^-\mathcal{H}^T\left(\mathcal{H}\mathcal{P}^-\mathcal{H}^T + DRD^T\right)^{-1} \tag{5.12}$$

The a posteriori estimation error of the augmented state vector $\boldsymbol{\mathcal{X}}$ is

$$\tilde{\boldsymbol{\mathcal{X}}}^+ = \left[I - \breve{K}\breve{H}\right]\tilde{\boldsymbol{\mathcal{X}}}^- - \breve{K}D\mathbf{v} \tag{5.13}$$

while the augmented covariance matrix is updated according to

$$\mathcal{P}^+ = [I - \mathcal{K}\mathcal{H}]\,\mathcal{P}^-\,[I - \mathcal{K}\mathcal{H}]^T + [\mathcal{K}D]\,R\,[\mathcal{K}D]^T \tag{5.14}$$

The a posteriori estimation errors of the three state vectors in $\boldsymbol{\mathcal{X}}$ can be explicitly written based on Eq. (5.13) as:

$$\tilde{\mathbf{X}}_3^+ = \left[I - \breve{K}_3H_3\right]\tilde{\mathbf{X}}_3^- - \breve{K}_3H_2\tilde{\mathbf{X}}_2^- - \breve{K}_3H_1\tilde{\mathbf{X}}_1^- - \breve{K}_3D\mathbf{v} \tag{5.15}$$

$$\tilde{\mathbf{X}}_2^+ = \left[I - \breve{K}_2H_2\right]\tilde{\mathbf{X}}_2^- - \breve{K}_2H_3\tilde{\mathbf{X}}_3^- - \breve{K}_2H_1\tilde{\mathbf{X}}_1^- - \breve{K}_2D\mathbf{v} \tag{5.16}$$

$$\tilde{\mathbf{X}}_1^+ = \left[I - \breve{K}_1H_1\right]\tilde{\mathbf{X}}_1^- - \breve{K}_1H_3\tilde{\mathbf{X}}_3^- - \breve{K}_1H_2\tilde{\mathbf{X}}_2^- - \breve{K}_1D\mathbf{v} \tag{5.17}$$

5.3.2 Calculation of the Cross-Covariance Terms P_{32}, P_{31} and P_{21}

The required cross-covariance terms in each three-view update are P_{32}, P_{31} and P_{21}. These terms can be calculated by adjusting the method for calculating the cross-covariance terms for a general MP measurement model, which was developed in Chapter 4, to the specific measurement model of three-view measurements (5.5).

Since the residual measurement $\mathbf{z}$ is constituted upon navigation and imagery data of three views, Eq. (5.4) can be expressed in a similar manner to Eq. (4.7) with $r = 3$ (cf. Section 4.1):

$$\mathbf{z}(t) = \mathbf{h}\left(\mathbf{x}_1(t_1), \mathbf{y}_1(t_1), \mathbf{x}_2(t_2), \mathbf{y}_2(t_2), \mathbf{x}_3(t_3), \mathbf{y}_3(t_3)\right) \tag{5.18}$$

In the general case, $\mathbf{y}_i(t_i)$ denotes the external sensors readings of the ith platform at some time instant t_i. These readings are corrupted by a Gaussian white noise $\mathbf{v}_i(t_i)$. In the current case, the camera is the only required external sensor, and thus, $\mathbf{y}_i(t_i)$ represents the pixel coordinates of the image captured by the ith platform at time t_i. Consequently, the linearized measurement equation (5.5)

$$\mathbf{z} \approx H_3(t_3)\mathbf{X}_3(t_3) + H_2(t_2)\mathbf{X}_2(t_2) + H_1(t_1)\mathbf{X}_1(t_1) + D\mathbf{v} \tag{5.19}$$

can be interpreted in terms of Eq. (4.7), so that

$$H_i(t_i) = \nabla_{\boldsymbol{\zeta}_i(t_i)}\mathbf{h} \quad , \quad i = 1, 2, 3 \tag{5.20}$$

and $D = \begin{bmatrix} D_1(t_1) & D_2(t_2) & D_3(t_3) \end{bmatrix}$, $\mathbf{v} = \begin{bmatrix} \mathbf{v}_1^T(t_1) & \mathbf{v}_2^T(t_2) & \mathbf{v}_3^T(t_3) \end{bmatrix}^T$ with

$$D_i(t_i) = \nabla_{\mathbf{y}_i(t_i)}\mathbf{h} \tag{5.21}$$

Now, the only missing part for applying the graph-based method for calculating the cross-covariance terms (cf. Chapter 4) is calculation of the noise covariance between some two adjacent nodes c_{l-1} and c_l in any of the two trees that are constructed from the graph G (cf. Section 4.3.2.3): $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T]$. This term is given in the general case by Eq. (4.38):

$$E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T] = \begin{cases} Q_{c_l:c_{l-1}} & \text{transition relation} \\ K_{c_{l-1}} D_{c_l} R_{c_l:c_{l-1}} (K_{c_{l-1}} D_{c_l})^T & \text{update relation} \end{cases} \tag{5.22}$$

While in case of a transition relation, $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T] = Q_{c_l:c_{l-1}}$ is calculated from the process noise $\boldsymbol{\omega}$ of the appropriate platform, a clarification is required for calculating $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T]$ in the case of an update relation. For an update relation, c_{l-1} is an a posteriori node, while c_l is an a priori node representing one of the three images that participate in a given three-view measurement. Thus, the node c_{l-1} has three parents, one of which is the node c_l.

Recall the term DRD^T which participates in the update step of the querying platform (cf. Eq. (5.9)). This term represents the noise covariance of *all* the three images constituting a given three-view measurement. An explicit expression for DRD^T is given by Eq. (B.90) (cf. Appendix B):

$$\begin{aligned} DRD^T \;=\; & \sum_{i=1}^{N_{123}+\Delta N_{12}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{1_i}^{C_1}} + \sum_{i=1}^{N_{123}+\Delta N_{12}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{2_i}^{C_2}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{2_i}^{C_2}} + \\ & + \sum_{i=1}^{N_{123}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{3_i}^{C_3}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{3_i}^{C_3}} \end{aligned} \tag{5.23}$$

However, the term $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}_{c_l:c_{l-1}}^T]$ represents the noise covariance of only *one* of the three images. Therefore, only part of the ingredients that appear in Eq. (5.23) should be

taken when calculating $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}^T_{c_l:c_{l-1}}]$. Let the contribution of the sth view, $s=1,2,3$, to the term DRD^T be denoted as $\left(DRD^T\right)_{\text{view } s}$:

$$\left(DRD^T\right)_{\text{view } 1} \doteq \sum_{i=1}^{N_{123}+\Delta N_{12}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{1_i}^{C_1}} \tag{5.24}$$

$$\left(DRD^T\right)_{\text{view } 2} \doteq \sum_{i=1}^{N_{123}+\Delta N_{12}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{2_i}^{C_2}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{2_i}^{C_2}} \tag{5.25}$$

$$\left(DRD^T\right)_{\text{view } 3} \doteq \sum_{i=1}^{N_{123}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{3_i}^{C_3}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{3_i}^{C_3}} \tag{5.26}$$

Then, $E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}^T_{c_l:c_{l-1}}]$ is given in the case of an update relation by

$$\begin{aligned} E[\boldsymbol{\eta}_{c_l:c_{l-1}}\boldsymbol{\eta}^T_{c_l:c_{l-1}}] &= K_{c_{l-1}} D_{c_l} R_{c_l:c_{l-1}} (K_{c_{l-1}} D_{c_l})^T = \\ &= K_3 \left(DRD^T\right)_{\text{view } s} K_3^T \quad , \quad s \in \{1,2,3\} \end{aligned} \tag{5.27}$$

where $K_{c_{l-1}} = K_3$ is the Kalman gain computed by the querying platform according to Eqs. (5.7)-(5.9).

5.4 Overall Distributed Scheme

Assume a scenario of M cooperative platforms. Each, or some, of these platforms maintain a repository of captured images attached with navigation data. All the platforms maintain a local copy of the graph, that is updated upon every multi-platform update event. This graph contains M threads, one thread for each platform in the group. The graph is initialized to M empty threads. The formulation of a single multi-platform update event is as follows.

The querying platform broadcasts its currently-captured image and its navigation solution to the rest of the platforms. A platform that receives this query, performs a check in its repository whether it has previously captured images of the same region. Platforms that do not maintain such a repository perform this check over the currently captured image only. Different procedures for performing this query may be devised. One possible alternative is to check only those images in the repository, which have a reasonable navigation data attached, e. g. images that were captured from a vicinity of the transmitted position of the querying platform.

Among the chosen images, only images that have a smaller uncertainty in their attached navigation data, compared to the uncertainly in the transmitted navigation data of the querying platform, are transmitted back to the querying platform. More specifically, denote by P_Q the covariance matrix of the querying platform, and P the covariance matrix attached to one of the chosen images from a repository of some other platform in

the group. Then, in the current implementation, this image is transmitted back to the querying platform only if its position uncertainty is smaller than the position uncertainty of the querying platform, i. e.:

$$(P)_{ii} < \alpha(P_Q)_{ii} \quad , \quad i = 1, 2, 3 \tag{5.28}$$

where $(A)_{ij}$ is the member from the ith row and jth column of some matrix A, and α is a constant satisfying $0 < \alpha \leq 1$. Naturally, other criteria can be applied as well.

The chosen images, satisfying the above condition are transmitted to the querying platform, along with their attached navigation data. In addition, a transition matrix between the transmitted images, should more then one image is transmitted by the same platform, is sent. In case the replying platform has already participated in at least one multi-platform update of any platform in the group, its thread in the graph will contain at least one node. Therefore, transition matrices bridging the navigation data attached to the images being transmitted in the current multi-platform update to the closest nodes in this thread are also sent.

As an example, consider the scenario shown in Figure 5.3. Figure 5.4 presents the construction details of the graph for this scenario, for each of the executed three-view measurement updates. Assume the first update, a_3^+, was executed, and focus on the second update, b_3^+. As shown in Figure 5.4(b), platform I transmits two images and navigation data, denoted by the nodes b_1^- and b_2^- in the graph. However, in addition to the transmitted transition matrix and process noise covariance matrix between these two nodes, $\phi_{b_1 \to b_2}$ and $Q_{b_1:b_2}$, the transition matrix and noise covariance matrix between the nodes b_2 and a_3, $\phi_{b_2 \to a_3}$ and $Q_{b_2:a_3}$, are transmitted as well.

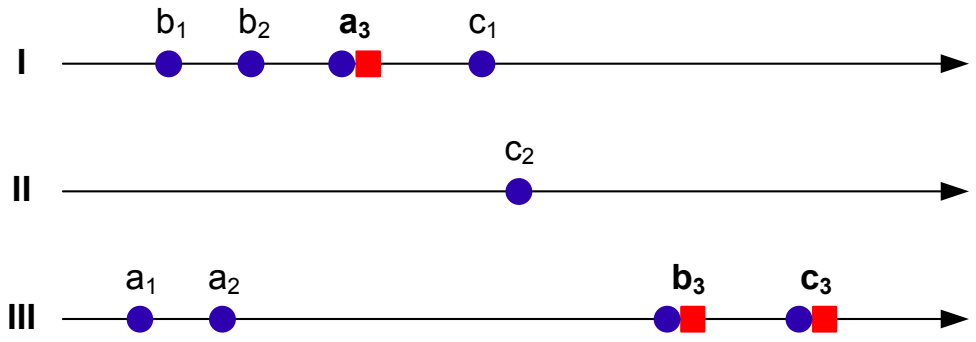


Figure 5.3: Three-view MP scenario

Upon receiving the transmitted images and the navigation data, two best images are selected[2], the cross-covariance terms are calculated based on the local graph, as discussed in Chapter 4, followed by computation of all the relevant filter matrices: $H_3, H_2, H_1, \mathcal{A}, \mathcal{B}, D$.

[2]The selection is according to some criteria, e. g., Eq. (5.28). Alternatively, the proposed approach may be also applied on more than three images.

Next, the update of the querying platform is carried out based on Section 5.3. Now, it is only required to update the local graphs of all the platforms in the group by the performed update event. The querying platform broadcasts the following information: *a)* identity of the involved platforms in the current update; *b)* time instances (or some other identifiers) of the involved images; required transition matrices of the involved images; *c)* a priori and a posteriori covariance and cross-covariance matrices involved in the current update event; *d)* filter matrices K_3, H_3, H_2 and H_1. Then, each platform updates its own graph representation.

The process described above is summarized in Algorithms 5 and 6. Algorithm 5 contains a protocol of actions carried out by the querying platform, while Algorithm 6 provides the protocol of actions for the rest of the platforms in the group.

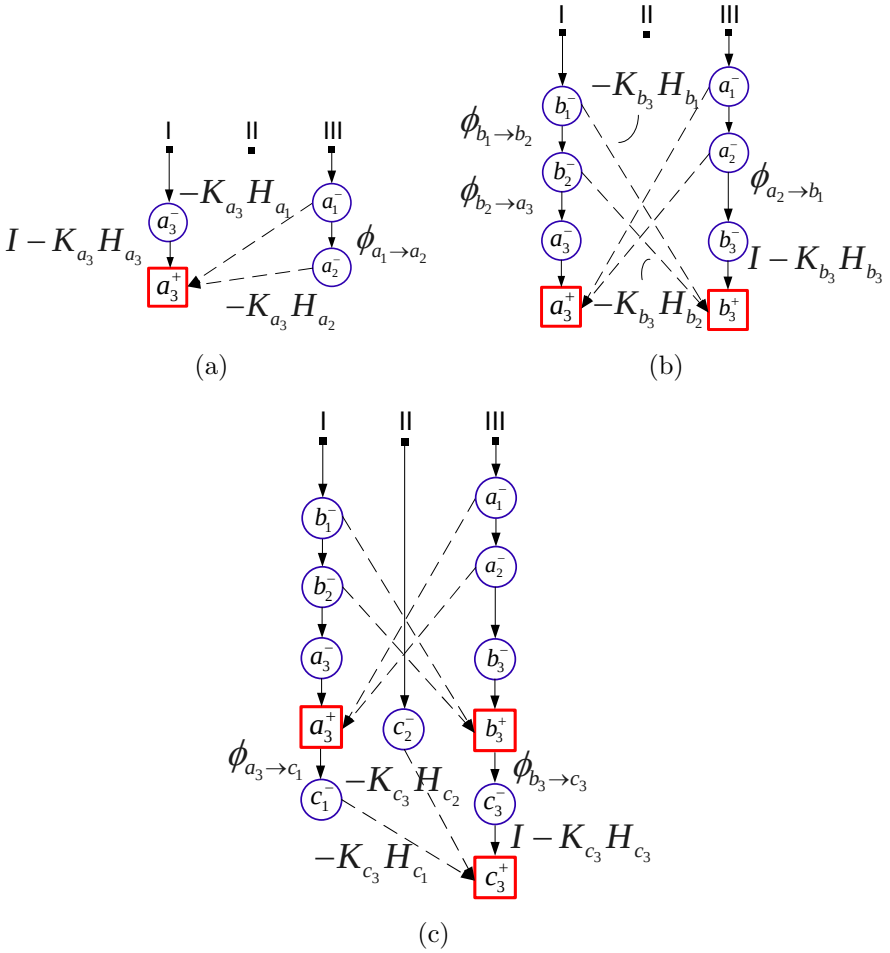


Figure 5.4: Graph update process: **a)** update event a_3^+; **b)** update event b_3^+; **c)** update event c_3^+.

Algorithm 5 Querying Platform Protocol

1: Notations: Q - Querying platform; A, B - two other platforms.
2: Broadcast current image I^Q and current navigation data.
3: Receive a set of images and associated navigation data from other platforms. See steps 2-9 in Algorithm 6.
4: Choose two best images I^A, I^B transmitted by platforms A and B, respectively.
5: **First graph update:**

- Add a new node for each image in the appropriate thread (A, B and Q).
- Denote these three new nodes in threads A, B and Q as β_1, β_2 and β_3, respectively.
- Connect each such node to previous and next nodes (if exist) in its thread by directed arcs associated with the transition matrices and with the process noise covariance matrices.

6: Calculate cross-covariance terms based on the local graph.
7: Calculate the measurement $\mathbf{z}$ and the filter matrices K_3, H_3, H_2, H_1, D based on the three images I^A, I^B, I^Q and the attached navigation data.
8: Perform navigation update on platform Q.
9: **Final graph update:**

- Add an update-event node, denoted by α, in the thread Q.
- Connect the nodes β_1, β_2 and β_3 to the update-event node α by directed arcs weighted as $-K_3H_1, -K_3H_2$ and $I - K_3H_3$, respectively. Associate also measurement noise covariance matrix to each arc (cf. Eqs. (5.24)-(5.26)).
- Store a priori and a posteriori covariance and cross-covariance terms (e. g. in the nodes $\beta_1, \beta_2, \beta_3$ and α).

10: Broadcast update event information.

5.4.1 Handling Platforms Joining or Leaving the Group

Whenever a platform joins an existing group of cooperative platforms, it must obtain the graph describing the history of multi-platform updates among the platforms in the group. This graph can be transmitted to the joining platform by one one of the platforms in the group. Departure of a platform from the group does not require any specific action.

An interesting scenario is one in which there are *several* groups of cooperative platforms, and a platform has to migrate from one group to another. Refer to the former and the latter groups as the source and destination groups, respectively. For example,

Algorithm 6 Replying Platform Protocol

1: Notations: Q - Querying platform; A - current platform.
2: **if** a query image and its navigation data are received from platform Q **then**
3: Search repository for images containing the same scene.
4: Choose images that satisfy the navigation uncertainty criteria (5.28).
5: For each chosen image, captured at some time instant k, look among all the nodes in thread A in the local graph, for two nodes with time l and m that are closest to k, such that $l < k < m$.
6: Calculate transition matrices $\phi_{l\to k}$ and $\phi_{k\to m}$ and noise covariance matrices $Q_{l:k}$ and $Q_{k:m}$.
7: **if** more than one image was chosen in step 4 **then**
8: Calculate transition matrices and noise covariance matrices between the adjacent chosen images.
9: Transmit the chosen images, their navigation data and the calculated transition and noise covariance matrices to the querying platform Q.
10: **if** update message is received from Q **then**
11: Update local graph following steps 5 and 9 in Algorithm 5.

this might be the case when each cooperative group operates in a distinct location and there is a need to move a platform within these groups. In these scenarios the migrating platform has already a local graph representing the multi-platform events of the source group, while the destination group has its own graph.

These two graphs have no common threads only when each platform is assigned only to one group, and, in addition, no migration between the groups have occurred in the past. In any case, upon receiving the graph of the destination group, the joining platform must fuse the two graphs and broadcast the updated graph to all the platforms in the destination group.

5.5 Simulation and Experimental Results

In this section the developed method for vision-aided cooperative navigation is studied in two different scenarios. First, a formation of two platforms is considered. Statistical results, based on simulated navigation data and synthetic imagery are presented (Section 5.5.2). Next, a holding pattern scenario is demonstrated in an experiment using real imagery and navigation data (Section 5.5.3).

5.5.1 Implementation Details

The navigation simulation for *each* of the two platforms consists of the steps described in Section 3.4.1.1.

Table 5.1: Initial Navigation Errors and IMU Errors in a Formation Scenario

Parameter	Description	Platform I	Platform II	Units
$\mathbf{\Delta P}$	Initial position error (1σ)	$(10, 10, 10)^T$	$(100, 100, 100)^T$	m
$\mathbf{\Delta V}$	Initial velocity error (1σ)	$(0.1, 0.1, 0.1)^T$	$(0.3, 0.3, 0.3)^T$	m/s
$\mathbf{\Delta \Psi}$	Initial attitude error (1σ)	$(0.1, 0.1, 0.1)^T$	$(0.1, 0.1, 0.1)^T$	deg
$\mathbf{d}$	IMU drift (1σ)	$(1, 1, 1)^T$	$(10, 10, 10)^T$	deg/hr
$\mathbf{b}$	IMU bias (1σ)	$(1, 1, 1)^T$	$(10, 10, 10)^T$	mg

Each platform is handled independently based on its own trajectory. Once a platform obtains three images with a common overlapping area, the developed algorithm is executed: cross-covariance terms are computed, followed by estimation of the state vector. The estimated state vector is then used for updating the navigation solution and the IMU measurements (cf. Figure 5.1). Next, the update information is stored and delivered to the second platform. The image processing module is identical to the one described in Section 3.4.1.1.

5.5.2 Formation Flying Scenario - Statistical Results

In this section the proposed method for vision-aided cooperative navigation is applied on a formation flying scenario, comprised of two platforms. Each of the platforms is equipped with its own navigation system and onboard camera. The navigation system of Platform I is of a better quality, compared to the navigation system of Platform II. Table 5.1 presents the assumed initial navigation errors and the errors of the IMU of the two platforms.

The two platforms performed the same straight and level north-heading trajectory at a velocity of 100 m/s, with Platform I being 20 seconds ahead of platform II. The synthetic imagery data was obtained by assuming a $20^0 \times 30^0$ field of view, focal length of 1570 pixels, and image noise of 0.5 pixel. The ground landmarks were randomly drawn with a height variation of ± 200 meters relative to the mean ground level. In the considered scenario, Platform I transmitted information (images and navigation data) to Platform II, thereby allowing to update the navigation system of Platform II using the three-view measurement model. Platform II was updated using the proposed method every 10 seconds, with the first update carried out after 27 seconds of inertial flight. The navigation system of Platform I is not updated, and it therefore develops inertial navigation errors over time.

Figure 5.5 summarizes the assumed measurement schedule. The true translation motion between any three views participating in the same measurement is $T_{12} = 200$ meters and $T_{23} = 400$ meters, in the north direction. In each update, two of the three images[3]

[3]Since in this section a synthetic imagery data is used, the term "image" refers to a synthetic data,

that participate in the measurement, were taken from Platform I. Since the two platforms perform the same trajectory, with a 20 seconds time delay, these two images have been acquired by Platform I 20 seconds before the measurement. Therefore they were stored in Platform I's repository and retrieved upon request.

Figure 5.6(a) shows the equivalent graph that was used for calculating the cross-covariance terms in each update event of Platform II, based on the graph-based method proposed in Chapter 4 (cf. Algorithm 2). For example, the two trees $T_{b_3^-}$ and $T_{b_1^-}$, constructed for calculating $P_{b_3^- b_1^-}$ are given in Figure 5.6(b). In the considered scenario, the conditions of Corollary 4.3.3 (cf. Section 4.3.2.3) are satisfied, as can be seen in Figure 5.6(b). Therefore, the computed cross covariances do *not* involve any noise terms. The obtained cross-covariance terms in the considered scenario maintain a constant structure regardless of how many MP updates were performed so far. For example, the cross-covariance term $P^-_{b_3 b_2}$, required for the second MP update, is similar to Eq. (4.19):

$$P^-_{b_3 b_2} = \Phi^{II}_{a_3 \to b_3} \left\{ (I - K_{a_3} H_{a_3}) P^-_{a_3 a_2} - K_{a_3} H_{a_2} P^-_{a_2 a_2} - K_{a_3} H_{a_1} P^-_{a_1 a_2} \right\} (\Phi^I_{a_2 \to b_2})^T \quad (5.29)$$

while the terms $P^-_{b_3 b_1}$ and $P^-_{b_2 b_1}$ are given by

$$P^-_{b_3 b_1} = \Phi^{II}_{a_3 \to b_3} \left\{ (I - K_{a_3} H_{a_3}) P^-_{a_3 a_2} - K_{a_3} H_{a_2} P^-_{a_2 a_2} - K_{a_3} H_{a_1} P^-_{a_1 a_2} \right\} (\Phi^I_{a_2 \to b_1})^T \quad (5.30)$$

$$P^-_{b_2 b_1} = \Phi^I_{b_1 \to b_2} P^-_{b_1 b_1} \quad (5.31)$$

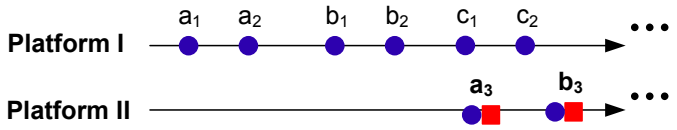


Figure 5.5: Three-view measurements schedule assumed in the simulation runs.

Figures 5.7 and 5.8 present the Monte-Carlo results (1000 runs) for Platform II, in terms of the mean navigation error (μ), standard deviation (σ) and square root covariance of the filter. As seen, the position and velocity errors (Figures 5.7(a) and 5.7(b)) are significantly reduced, compared to the inertial scenario, in *all* axes. The bias state is estimated also in *all* axes, while the drift state is only partially estimated. The updates yielded a mild reduction in Euler angle errors as well.

Figures 5.9 and 5.10 compare between the navigation errors of Platform II and Platform I. Although Platform II is equipped with an inferior navigation system, its performance is not inferior to the performance of Platform I. After several updates, Platform II actually outperforms Platform I. For example, the position errors of Platform II are smaller than the position errors of Platform I. The reason for this phenomenon is that

i. e. (noisy) features coordinates.

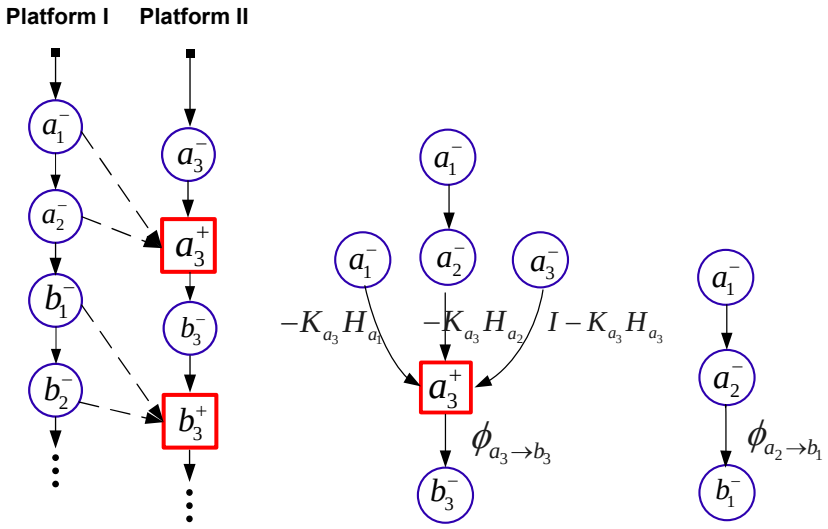
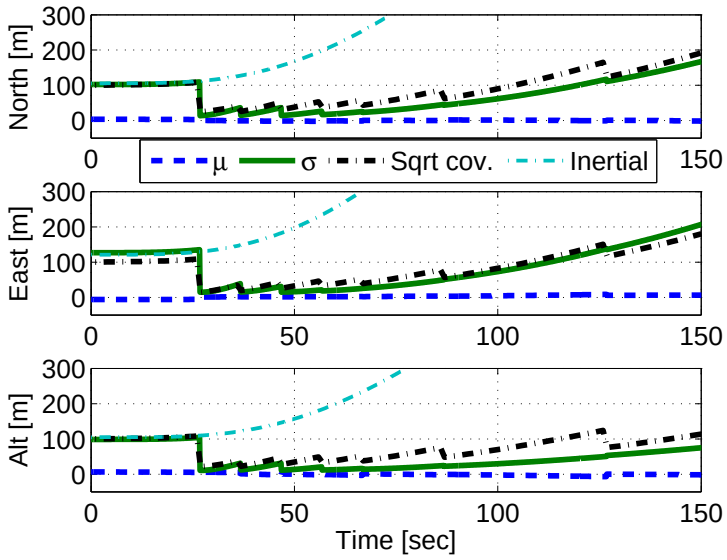


Figure 5.6: (a) Equivalent graph for the scenario shown in Figure 5.5. (b) The trees $T_{b_3^-}$ and $T_{b_1^-}$ required for calculating $P_{b_3^- b_1^-}$.

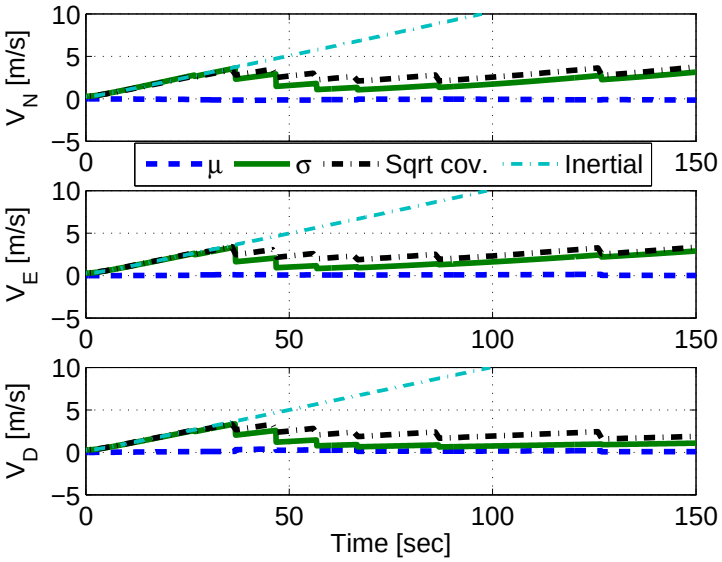
while the measurement is based upon three images, which were obtained from two platforms, only one of the platforms is actually updated. Updating both platforms would yield an improvement in both platforms [48]. Referring to Section 4.2.2, since Platform I contributes two sets of information to each MP measurement (e. g., at t_{a_1} and t_{a_2}), the graph will remain acyclic, if Platform I is updated at t_{a_2} and t_{b_2} (and not at t_{a_1} and t_{b_1}).

The importance of incorporating the cross-covariance terms in the update process is clearly evident when comparing the results of Figures 5.7 and 5.8 with Figure 5.11, that presents Monte-Carlo results when the cross-covariance terms are neglected. As seen in Figure 5.11, neglecting the cross-covariance terms results in a biased and inconsistent estimation of position and velocity errors along the motion heading.

It is also worth mentioning that should the leader perform self-updates based on the available sensors and information (e. g. epipolar constraints, GPS, DTM), improved navigation errors will be obtained not only in the leader but also in the follower navigation system.

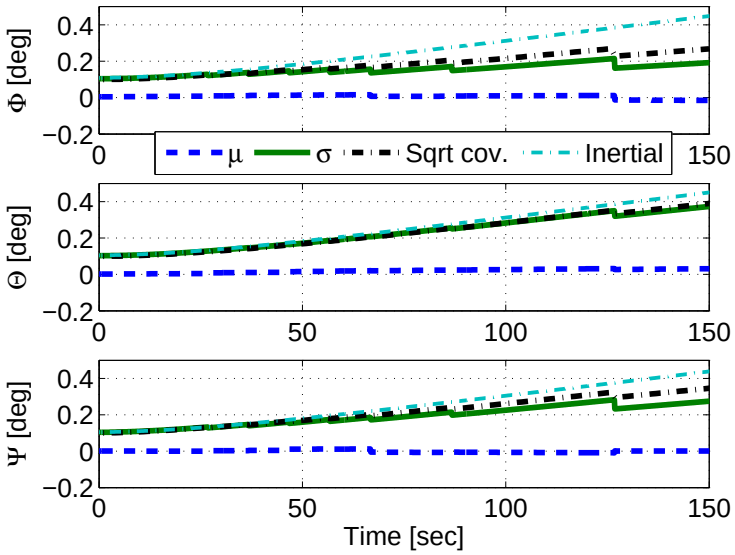


(a) Position errors.

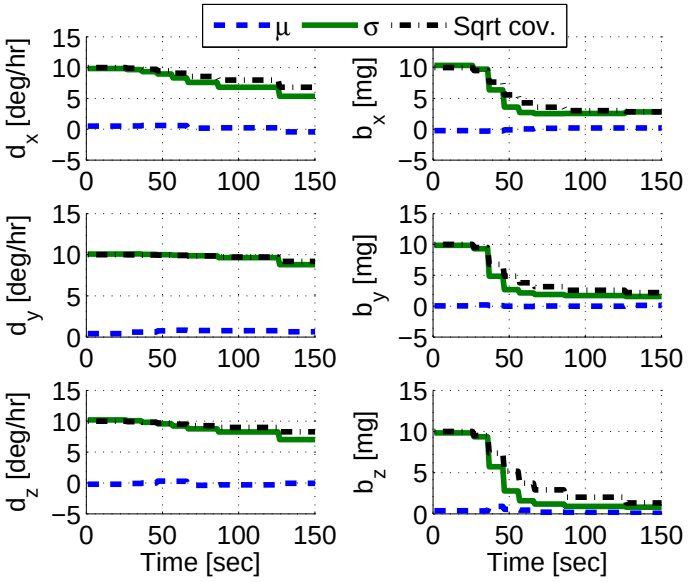


(b) Velocity errors.

Figure 5.7: Formation scenario - Monte Carlo (1000 runs) results; Platform II navigation errors compared to inertial navigation: Reduced position and velocity errors in *all* axes.

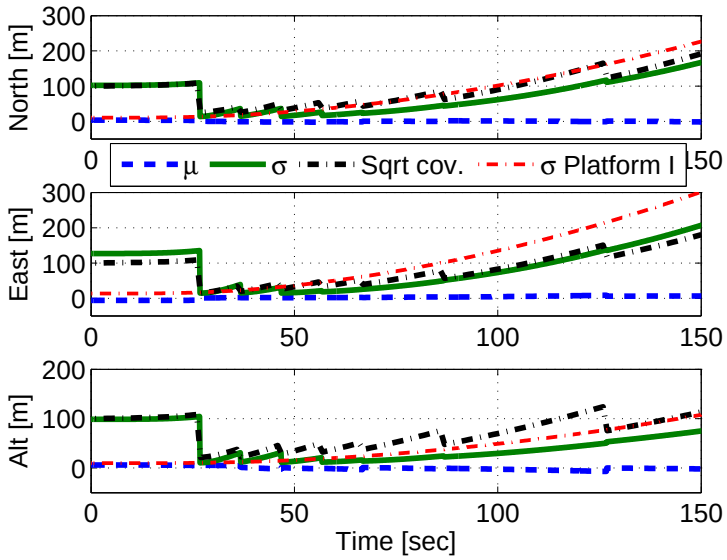


(a) Euler angles errors.

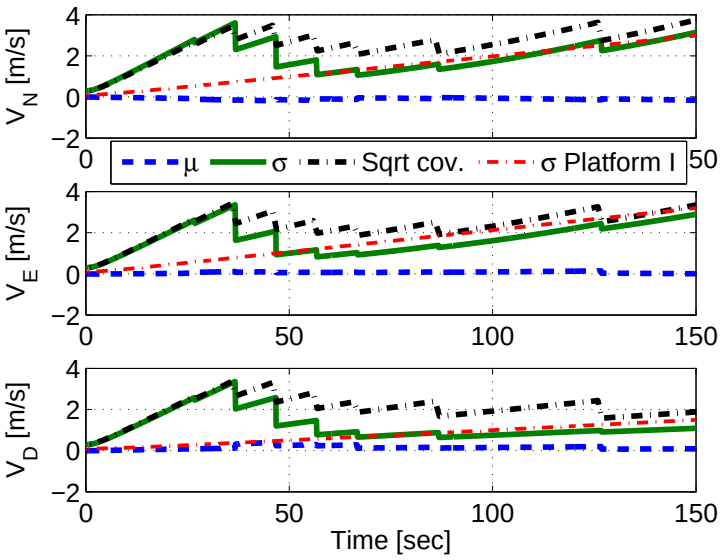


(b) Drift and bias estimation errors.

Figure 5.8: Formation scenario - Monte Carlo (1000 runs) results; Platform II navigation errors compared to inertial navigation: Bias estimation to the bias levels of Platform I (see also Figures 5.9 and 5.10).

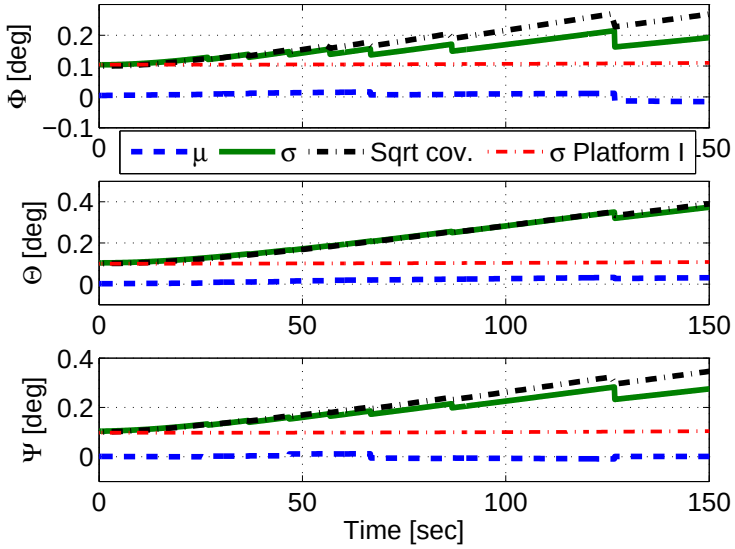


(a) Position errors.

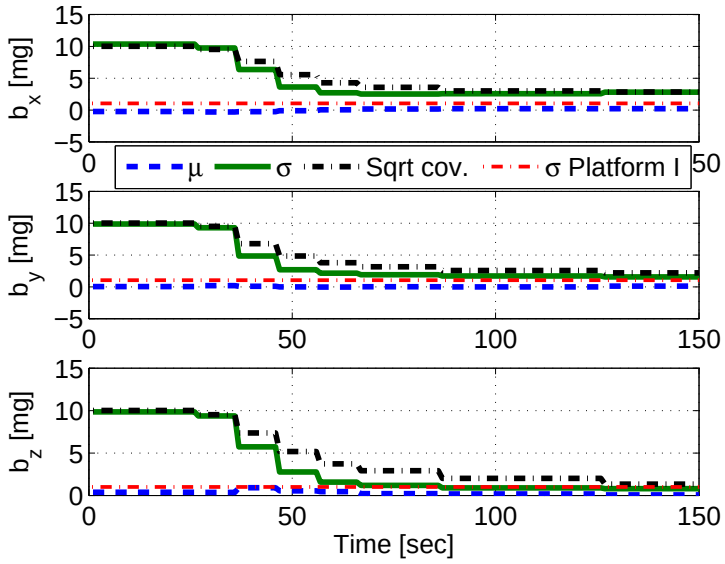


(b) Velocity errors.

Figure 5.9: Formation scenario - Monte Carlo (1000 runs) results; Platform II navigation errors compared to navigation errors of Platform I: Position and velocity errors are reduced below the level of errors of Platform I.

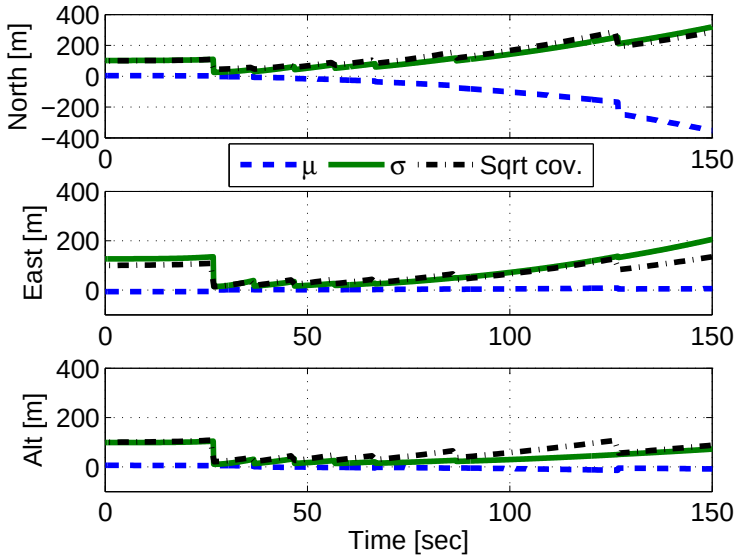


(a) Euler angles errors.

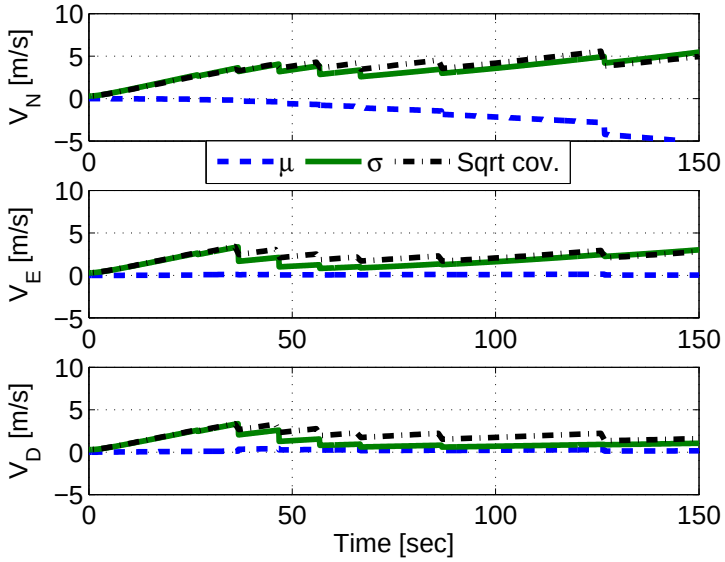


(b) Bias estimation errors.

Figure 5.10: Formation scenario - Monte Carlo (1000 runs) results; Platform II navigation errors compared to navigation errors of Platform I: Bias estimation to Platform I bias levels (1 mg). Euler angles are also reduced, however do not reach Platform I levels due to poor estimation of the drift state (cf. Figure 5.8(b)).



(a) Position errors.



(b) Velocity errors.

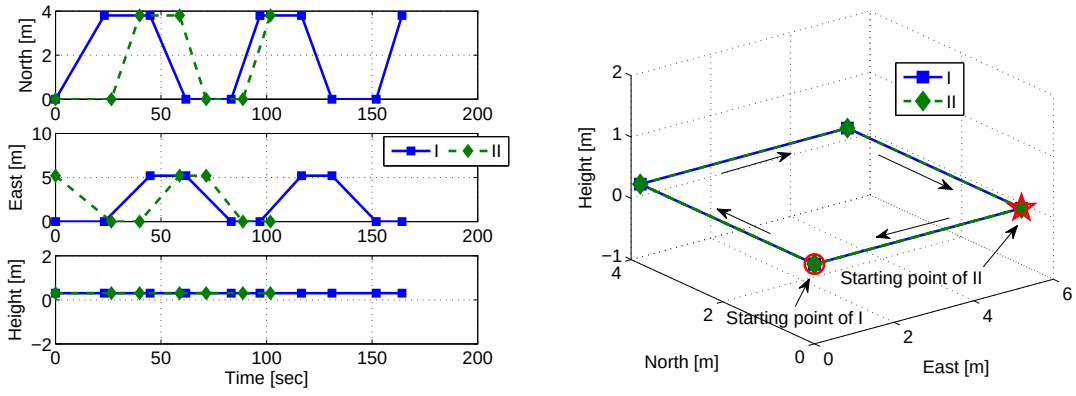
Figure 5.11: Formation scenario - Monte Carlo (1000 runs) results; Platform II navigation errors when cross-covariance terms are neglected: Biased estimation along the motion heading.

5.5.3 Holding Pattern Scenario - Experiment Results

In this section the proposed method for vision-aided cooperative navigation is demonstrated in an experiment. The experiment setup is identical to the setup that was used in Chapter 3: A single manually-driven ground vehicle attached with a wireless camera and an IMU. Refer to Section 3.4.3 for additional details.

The vehicle performed two different trajectories. The IMU and the camera were turned off between these two trajectories, thereby allowing to treat each trajectory as if it were performed by a different vehicle, equipped with a similar hardware (IMU and camera), as opposed to Section 5.5.2, where one of the vehicles was assumed to be equipped with a better navigation system. Thus, we have two ground vehicles, each performing its own trajectory and recording its own IMU and imagery data.

The two trajectories represent a holding pattern scenario. Each platform performs the same basic trajectory: Vehicle I performs this basic trajectory twice, while vehicle II performs the basic trajectory once, starting from a different point along the trajectory, and reaching the starting point of vehicle I after about 26 seconds. The reference trajectories of vehicle I and II are shown in Figure 5.12. These ground-truth trajectories were measured manually as the GPS was unavailable in the experiment (cf. Section 3.4.3). The diamond and square marks denote the manual measurements of the vehicles position. Each two adjacent marks of the same platform are connected using a linear interpolation.



(a) (b)

Figure 5.12: Trajectories of vehicles I and II in the experiment. Diamond and square marks indicate manually-measured vehicle locations. Circle and star marks in (b) denote the starting point of each platform.

The proposed method for multi-platform three-view based updates was applied several times in the experiment. In addition, the method was executed in a self-update mode, in which all the images are captured by the same vehicle (cf. Chapter 3). The required cross-covariance terms were calculated using the graph-based method developed in Chapter 4.

A schematic sketch of the measurements schedule is given in Figure 5.13. Table 5.2 provides further information, including the time instances of each participating triplet of images in the applied measurements. As seen, vehicle I is updated twice using data obtained from vehicle II (measurements **c** and **e**), and four times based on its own images (measurements $\mathbf{f}, \mathbf{g}, \mathbf{h}$ and **i**). Vehicle II is updated three times utilizing the information received from vehicle I (measurements $\mathbf{a}, \mathbf{b}$ and **d**). The vehicles performed inertial navigation elsewhere, by processing the recorded IMU data.

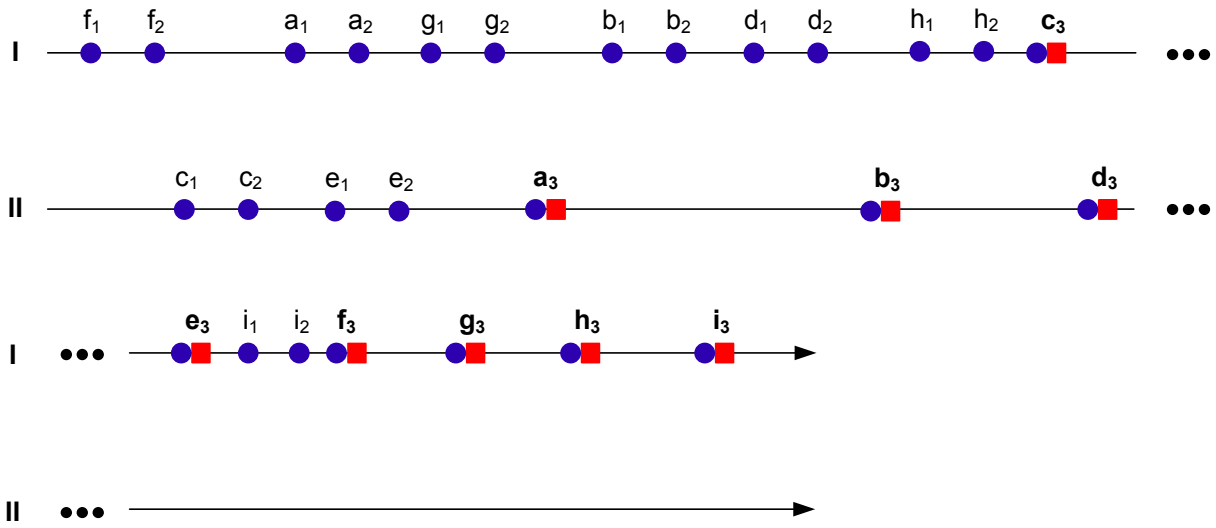


Figure 5.13: Schematic sketch of the measurement schedule in the experiment. Further information regarding each measurement is given in Table 5.2.

The images participating in each three-view update were manually identified and chosen. Figure 5.14 shows, for example, the three images of measurement **a**: images 5.14(a) and 5.14(b) were captured by vehicle I, while image 5.14(c) was captured by vehicle II. Features that were found common to all the three images (triplets) are also shown in the figure. Note that two objects (a bottle, and a bag) that appear in images 5.14(a) and 5.14(b) are missing in image 5.14(c). These two objects were removed between the two trajectories. Therefore, as seen in Figure 5.14, these two objects are not represented by matched triplets of features (but can be represented by matched pairs of features between the first two views). Additional details regarding the image processing phase in the experiment can be found in Section 3.4.3.

The experiment results are given in Figures 5.15 and 5.16: Figures 5.15(a) and 5.15(b) show the position errors for vehicle I and II, while Figures 5.16(a) and 5.16(b) show the velocity errors. Each figure consists of three curves: navigation error, square root covariance of the filter, and navigation error in an inertial scenario (given for reference). The measurement type (MP-update or self-update) is also denoted in the appropriate locations.

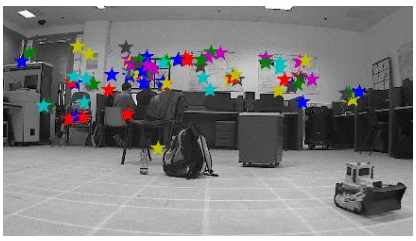
Table 5.2: Measurement details in the experiment.

Notation	Type	Querying vehicle	t_3 [sec]	Replying vehicle	t_1, t_2 [sec]
a	MP update	II	32.6	I	8.4, 14.2
b	MP update	II	53.2	I	35.9, 39.1
c	MP update	I	60.0	II	2.3, 5.6
d	MP update	II	60.6	I	47.9, 49.2
e	MP update	I	66.8	II	10.3, 12.1
f	Self update	I	81.1	I	0.3, 1.3
g	Self update	I	97.0	I	22.8, 24.3
h	Self update	I	124.7	I	54.3, 55.6
i	Self update	I	142.0	I	70.8, 72.1

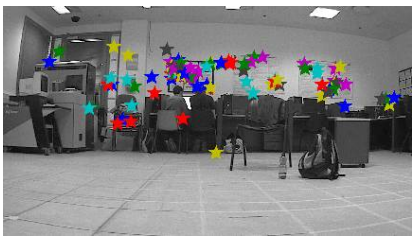
The position error was calculated by subtracting the navigation solution from the true trajectories (cf. Figure 5.12). In a similar manner, the velocity error was computed by subtracting the navigation solution from the true velocity profiles. However, since velocity was not measured in the experiment, it was only possible to obtain an approximation of it. The approximated velocity was calculated assuming that the vehicles moved with a constant velocity in each phase[4].

As seen from Figures 5.15(a), the position error of Vehicle I was nearly nullified in all axes as the result of the first update, which was of MP type. The next update (also MP) caused to another reduction in the north position error. After completing a loop in the trajectory, it became possible to apply the three-view updates in a self-update mode for Vehicle I, i. e. all the three images were captured by Vehicle I. Overall, due to the applied 6 three-view updates, the position error of Vehicle I has been confined to around 50 meters in north and east directions, and 10 meters in altitude. As a comparison, the position error of vehicle I in an inertial scenario reaches, after 150 seconds of operation, $900, 200$ and 50 meters in north, east and down directions, respectively. The position error of Vehicle II (cf. Figure 5.15(b)) has been also dramatically reduced as the result of the three-view multi-platform updates. For example, after the third update ($t \approx 60$ seconds), the position error was nearly nullified in north direction and reduced from 50 to 20 meters in east direction. One can observe that the velocity errors are also considerably reduced in all axes (cf. Figures 5.16(a) and 5.16(b)).

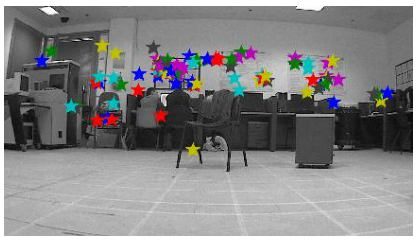
[4]The phase duration and the translation that each vehicle has undergone in each phase are known from analyzing the IMU measurements and from the true trajectories.



(a)



(b)



(c)

Figure 5.14: Images participating in measurement **a** and matched triplets of features. Images (a) and (b) were captured by vehicle I; Image (c) was captued by vehicle II. The images (a),(b) and (c) are represented in Figure 5.13 as a_1, a_2 and a_3.

5.6 Conclusions

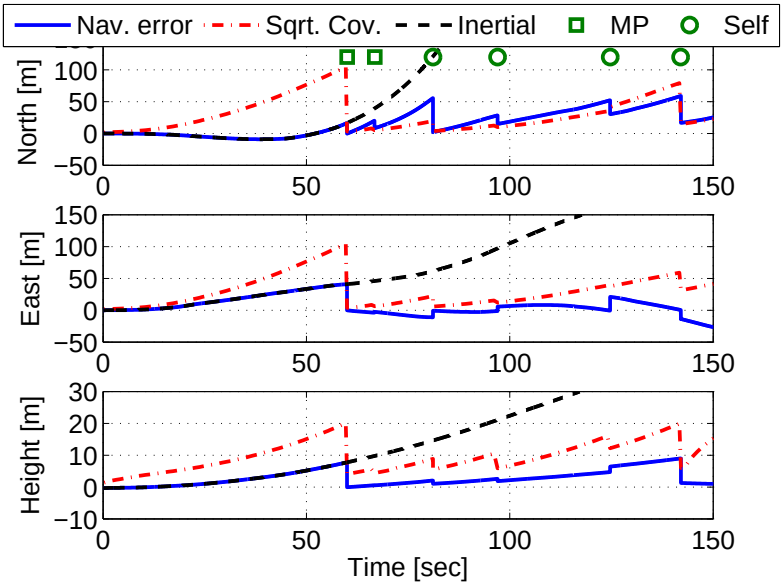
This chapter presented a new method for distributed vision-aided cooperative navigation based on the three-view geometry constraints that were developed in Chapter 3. A group of collaborative platforms was assumed. Each platform was equipped with an INS and a camera. The platforms were also assumed to be capable of communicating between themselves.

In the proposed method, a measurement was formulated whenever the same general scene was observed by different platforms. Three images of a common region were required for each measurement. These images were not necessarily captured at the same time. All, or some, of the platforms maintained a local repository of captured images, that were associated with some navigation parameters. In a typical scenario, a platform captured an image and broadcasted it, along with its current navigation solution, to other platforms in the group, inquiring if they had previously captured images containing the same region. Upon receiving such a query, each platform performed a check in its repository looking for appropriate images. Among these images, only images with a smaller navigation uncertainty compared to the uncertainty in the navigation data of the query image, were

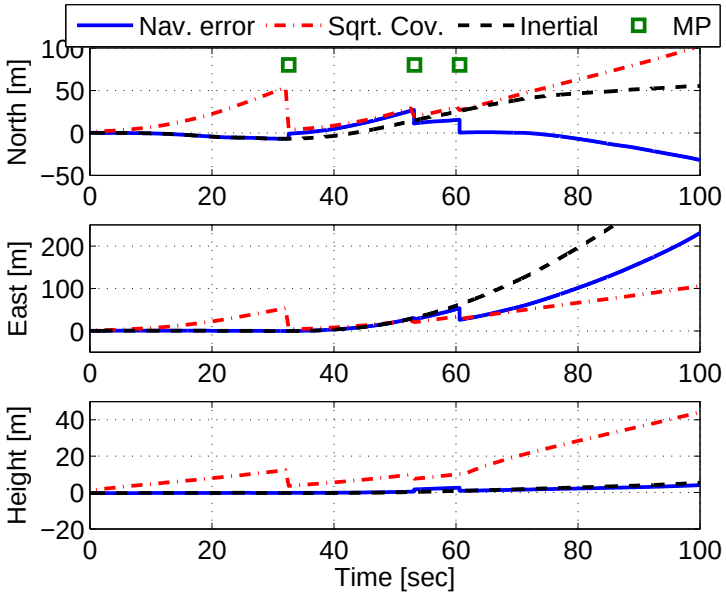
transmitted back.

The three-view geometry constraints, formulated based on the currently-captured image by the broadcasting platform and the imagery and navigation data obtained from different platforms, allowed reducing navigation errors of the broadcasting platform. In particular, position and velocity errors were reduced in all axes, without assuming a range sensor. Since the navigation parameters associated with the three images participating in the same measurement can be correlated, the required correlation terms were computed using the approach developed in Chapter 4.

The proposed method was studied in a simulated environment and in an experiment. Statistical results were presented, based on simulated navigation and synthetic imagery data, for a formation scenario of two platforms, in which Platform I was equipped with a higher quality INS compared to the INS of Platform II. The developed method allowed to reduce the rapidly-developing navigation errors of Platform II to the level of errors of Platform I. Applying the method for calculating the correlation terms allowed to obtain a consistent and unbiased estimation. It was shown that neglecting the correlation terms yields biased and inconsistent estimations of position and velocity. A holding pattern scenario was demonstrated in an experiment, involving two ground vehicles, equipped with identical inertial measurement units and cameras. Significant reduction in the navigation errors of both of the vehicles was obtained as a result of activating the proposed method.

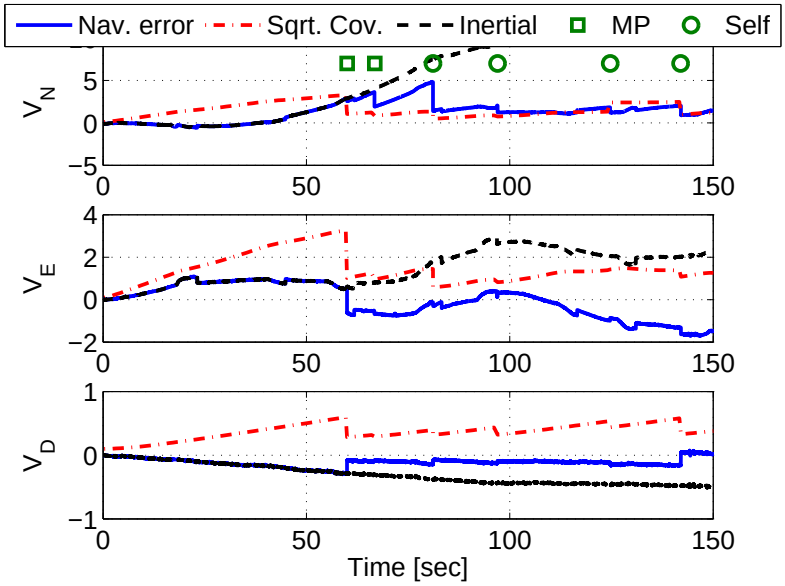


(a) Position errors - vehicle I

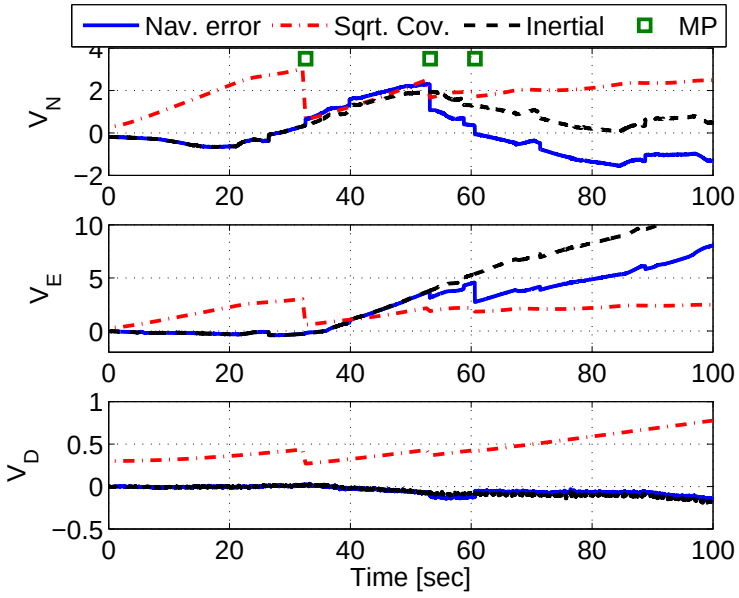


(b) Position errors - vehicle II

Figure 5.15: Position errors of vehicles I and II in the experiment.



(a) Velocity errors - vehicle I



(b) Velocity errors - vehicle II

Figure 5.16: Velocity errors of vehicles I and II in the experiment.

Chapter 6

Conclusions

The research presented in this dissertation focused on utilizing computer vision techniques for improving the performance of standard inertial navigation systems. Motivated mainly by the fact that in the absence of a GPS signal, or when such a signal is unreliable, the computed navigation solution is accompanied by unbounded errors that drastically increase over time, the main goal was to develop methods for eliminating, or at least reducing, these errors based on the imagery information acquired by a camera. The developed algorithms assumed that the platform was equipped with an inertial navigation system and a single onboard camera, possibly mounted on gimbals. In addition, the images were used for constructing a representation of the observed environment, i. e., mapping.

The fundamental approach of this research was to decouple the navigation aiding and mapping processes. This enables updating the navigation system in real time, while the mapping phase, such as mosaic construction, was executed in a background process. A constant-size state vector, comprised of the current navigation information only, was used by the navigation filter. The camera images, associated with the current navigation solution, were stored in a repository and retrieved upon demand later on for carrying out navigation aiding. The images stored in the repository were also used for online mosaicking.

The first algorithm dealt with improving navigation performance while operating in challenging scenarios, characterized by a narrow field of view camera observing low-texture scenes. It was proposed to couple between the mosaic construction and camera scanning processes, resulting in improved vision-based motion estimation. The estimated motion was fused with an inertial navigation system using an implicit extended Kalman filter, allowing to reduce some of the navigation errors, including position and velocity errors perpendicular to the motion heading. However, since the camera translation motion was estimated only up to scale, it was not possible to reduce position and velocity errors along the motion heading.

A special attention was given to scenarios, in which the platform visits the same regions more than once. Since the trajectory is usually unknown a priori, the actual regions to

be revisited and the time instances are also unknown. The loop scenarios hold great potential both for navigation aiding and for refining the constructed map. The common approaches for handling loops, such as bundle adjustment and simultaneous localization and mapping, require processing all the images captured during-loop, which is a costly operation.

In the second algorithm, it was proposed to utilize constraints, stemming from observing a general static scene by three distinct views, for navigation aiding. A new formulation of such constraints was developed and fused with the navigation system. The algorithm performed navigation aiding each time three images with a common overlapping area were obtained, without reconstructing the observed scene. Given three images with a common overlapping area, position errors in all axes were reduced to the levels of errors present while the first two images were captured. Other navigation errors were reduced as well. As opposed to navigation aiding, the map refinement can be executed in a background process, as discussed above. Such an approach is in particular notable when considering loops.

The second part of the thesis was devoted to cooperative navigation. A group of collaborating platforms, capable of intercommunication, was assumed, each platform equipped with its own inertial navigation system, a single camera and perhaps other external sensors.

Regardless of the method applied for cooperative navigation, the involved information, obtained from different platforms, can be in the general case correlated. This correlation should be taken into account for obtaining consistent information fusion. A new graph-based approach was developed for explicitly calculating the required correlation terms, assuming a general multi-platform measurement model, involving navigation information and readings of onboard sensors of any number of platforms, possibly obtained at different time instances.

Finally, it was proposed to extend the three-view geometry navigation aiding algorithm, originally developed for a single platform, to cooperative navigation. Thus, a measurement based on the three-view constraints was formulated whenever the same static scene was observed in three views, possibly captured by different platforms, not necessarily at the same time. Since information from different platforms was involved, the required correlation terms were calculated using the above-mentioned method, that was adapted to the specific three-view multi-platform measurement model. This approach for cooperative navigation reduced the navigation errors, in particular position and velocity errors in all axes, without requiring a range sensor.

Chapter 7

Recommendations for Future Research

The research presented herein can be extended in several directions. Some suggestions appear in the following list.

- Utilization of the three-view constraints for identifying and tracking dynamic objects, while performing navigation aiding using static landmarks. In particular, one can refer to tracking the dynamic objects as a sort of mapping and investigate if there are scenarios in which representing all the dynamic objects in an augmented state vector, such as in SLAM, is advantageous over tracking each dynamic object on its own.

- In this research, it was assumed that the platform is equipped with an INS and a single camera only. However, as discussed in Section 1.1, different methods have been developed for navigation aiding assuming existence of additional information (e. g. DTM) and additional external sensors (e. g. range sensor). In this context, it would be interesting to investigate whether applying the three-view constraints in conjunction with these methods can improve the performance. For example, would the constraints enhance the performance if they were to be applied in addition to the DTM-based pose and motion recovery method developed in [8]? Another interesting issue is utilization of the three-view constraints as part of the SLAM approach.

- Using the techniques developed in this research thesis, several interesting problems can be addressed:

  - Several platforms, each equipped with an INS and a single camera, observe some other platforms which are equipped only with an INS. Refer to the former as group A and to the latter as group B. As seen in Chapter 5, it is possible to perform CN for the platforms in group A. Now, the question is whether

platforms from group B can be also updated within the CN framework of group A

  - Autonomous return home upon identifying GPS dropout, using the so-far constructed mapping/mosaic and utilizing the VAN methods developed in this research.
  - Satellite orbit determination/navigation and cooperative navigation.

- A more direct continuation of this research thesis is: 1) To investigate how the three-view constraints are related to the trifocal tensor, and to perform a comparative performance analysis for both of them; 2) Further investigate the approach of performing the mapping in a background process, adopted in this research thesis (cf. Section 1.2), as opposed to the SLAM approach.

- Application of the approach for explicit calculation of cross-covariance terms, developed in Chapter 4, to other problems that involve information fusion of possibly correlated data.

- Application of advanced filtering methods such as particle filtering for navigation data fusion of multiple cooperative platforms relying on the three-view aiding method developed in this research.

Appendix A

Chapter 2 Extras

This appendix presents a development of a measurement model used in the method for mosaic-aided navigation, described in Chapter 2. The model relates between the image-based estimated camera relative motion $\mathbf{t}_{1\to2}^{C_2}, R_{C_1}^{C_2}$ and the developing navigation errors of a standard INS and the parameters that model the IMU errors. The state vector is defined in Eq. (1.10):

$$\mathbf{X} = \begin{bmatrix} \Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\boldsymbol{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T \end{bmatrix}^T \tag{A.1}$$

Under *ideal conditions*, i. e. when there are no navigation errors and $\mathbf{t}_{1\to2}^{C_2}$, $R_{C_1}^{C_2}$ are perfectly estimated, the following can be written:

$$\mathbf{Pos}_{True}^{L_2}(t_2) - \mathbf{Pos}_{True}^{L_2}(t_1) = \gamma C_{L_2,True}^{C_2} \mathbf{t}_{1\to2,True}^{C_2} \tag{A.2a}$$

$$C_{C_1,True}^{C_2} = R_{C_1,True}^{C_2} \tag{A.2b}$$

where $C_{L_2}^{C_2}$ is the DCM transforming from C to LLLN at the time instance $t = t_2$; $T_{C_1}^{C_2}$ is the DCM transforming from C at $t = t_2$ to C at $t = t_1$; and $\mathbf{Pos}^{L(t_2)}(t_1)$ is the platform's position at $t = t_1$ expressed in the LLLN system at $t = t_2$, so that $\mathbf{Pos}^{L(t_2)}(t_1) = C_{L(t_2)}^{L(t_1)} \mathbf{Pos}^{L(t_1)}(t_1)$. The subscript $(\cdot)_{True}$ in Eq. (A.2) indicates ideal conditions as defined above.

The DCM $C_{L_2}^{C_2}$ is required since the extracted translation $\mathbf{t}_{1\to2}^{C_2}$ is given in the camera reference frame, while the left side of Eq. (A.2a) is expressed in the LLLN system.

A.1 Translation Measurement Equation

In an ideal situation, with no navigation and image processing errors, the two sides of Eq. (A.2a) constitute parallel vectors. Thus, this equation yields the following constraint:

$$\left[\mathbf{Pos}_{True}^{L_2}(t_2) - \mathbf{Pos}_{True}^{L_2}(t_1)\right] \times C_{L_2,True}^{C_2} \mathbf{t}_{1\to2,True}^{C_2} = \mathbf{0} \tag{A.3}$$

In reality, there are navigation errors that increase with time, moreover, the estimated camera matrix contains errors due to image noise. Thus, Eq. (A.3) no longer holds. Denoting by Nav parameters that are taken from the navigation data and by $\hat{\mathbf{t}}_{1\to2}^{C_2}$ the actual estimated translation vector obtained from the image processing module, Eq. (A.3) becomes

$$\left[\mathbf{Pos}_{Nav}^{L_2}(t_2) - \mathbf{Pos}_{Nav}^{L_2}(t_1)\right] \times C_{L_2,Nav}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} = \mathbf{z}_{translation} \tag{A.4}$$

where $\mathbf{z}_{translation}$ denotes the residual measurement vector.

Taking into account the fact that $\mathbf{Pos}_{Nav}^{L_2}(.) = \mathbf{Pos}_{True}^{L_2}(.) + \Delta\mathbf{P}^{L_2}(.)$ and subtracting (A.3) from (A.4) results in

$$\left[\Delta\mathbf{P}(t_2) - \Delta\mathbf{P}(t_1)\right]^{L_2} \times C_{L_2,Nav}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} + \mathbf{v}_{tr} = \mathbf{z}_{translation} \tag{A.5}$$

where $\mathbf{v}_{tr} = \left[\mathbf{Pos}_{True}(t_2) - \mathbf{Pos}_{True}(t_1)\right]^{L_2} \times \left[C_{L_2,Nav}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2} - C_{L_2,True}^{C_2}\mathbf{t}_{1\to2,True}^{C_2}\right]$. The vector $\mathbf{v}_{tr}$ is due to imperfect translation measurements (which are image-based estimations) and navigation errors. One may verify that in ideal conditions this term is nullified.

The inertial position error for a sufficiently small $\Delta t = t_2 - t_1$ or for a straight and level flight is given by Eq. (1.17) (the Nav subscript is omitted for simplicity from here on; thus, all parameters are computed based on the navigation system data, unless otherwise specified):

$$\begin{aligned}\Delta\mathbf{P}(t_2) &= C_{L_2}^{L_1}\left[-\frac{1}{6}A_s(t_1)C_{L_1}^{B_1}\mathbf{d}(\Delta t)^3\right. \\ &+ \left.\frac{1}{2}\left[A_s(t_1)\Delta\mathbf{\Psi}(t_1) + C_{L_1}^{B_1}\mathbf{b}\right](\Delta t)^2 + \Delta\mathbf{V}(t_1)\Delta t + \Delta\mathbf{P}(t_1)\right]\end{aligned} \tag{A.6}$$

Note that a transformation matrix, $C_{L_2}^{L_1}$, was added to express the position error at $t = t_2$ in LLLN coordinates.

Substituting Eq. (A.6) into Eq. (A.5), canceling position errors at $t = t_1$ and denoting $\hat{\mathbf{t}}_{1\to2}^{L_2} \equiv C_{L_2,Nav}^{C_2}\hat{\mathbf{t}}_{1\to2}^{C_2}$ yields

$$\left[C_{L_2}^{L_1}\left[-\frac{1}{6}A_s(t_1)C_{L_1}^{B_1}\mathbf{d}\cdot(\Delta t)^3 + \frac{1}{2}\left[A_s(t_1)\Delta\mathbf{\Psi}(t_1) + C_{L_1}^{B_1}\mathbf{b}\right](\Delta t)^2 + \Delta\mathbf{V}(t_1)\Delta t\right]\right]^{\wedge}\hat{\mathbf{t}}_{1\to2}^{L_2} + \mathbf{v}_{tr} = \mathbf{z}_{translation} \tag{A.7}$$

where $(.)^{\wedge}$ denotes the matrix cross-product equivalent. One can see from Eq. (A.7) that the translation measurement equation is of the form $\mathbf{z}_{translation} = H^{tr}\mathbf{X} + \mathbf{v}_{Tr}$, where

$$H^{tr} = \begin{bmatrix}0_{3\times3} & H_{\Delta V}^{tr} & H_{\Delta\Psi}^{tr} & H_d^{tr} & H_b^{tr}\end{bmatrix} \tag{A.8}$$

After some algebraic manipulations on Eq. (A.7), the submatrices of H^{tr} can be rendered into

$$H_{\Delta V}^{tr} = -\left[\hat{\mathbf{t}}_{1\to2}^{L_2}\right]^{\wedge}C_{L_2}^{L_1}\Delta t \qquad H_{\Delta\Psi}^{tr} = -\frac{1}{2}\left[\hat{\mathbf{t}}_{1\to2}^{L_2}\right]^{\wedge}C_{L_2}^{L_1}A_s(t_1)(\Delta t)^2$$

$$H_d^{tr} = \frac{1}{6}\left[\hat{\mathbf{t}}_{1\to2}^{L_2}\right]^{\wedge}C_{L_2}^{L_1}A_s(t_1)C_{L_1}^{B_1}(\Delta t)^3 \qquad H_b^{tr} = -\frac{1}{2}\left[\hat{\mathbf{t}}_{1\to2}^{L_2}\right]^{\wedge}C_{L_2}^{L_1}C_{L_1}^{B_1}(\Delta t)^2$$

A.2 Rotation Measurement Equation

Recall Eq. (A.2b), written under the assumption of ideal conditions: $C^{C_2}_{C_1,True} = R^{C_2}_{C_1,True}$. When taking into account navigation errors and errors in the estimated rotation matrix, this equation no longer holds. Instead, we define a residual error angle vector, $\mathbf{z}_{rotation}$. Under the assumption of small angles, this vector can be written as

$$I - \mathbf{z}^{\wedge}_{rotation,A} = C^{C_2}_{C_1,Nav}\left[\hat{R}^{C_2}_{C_1}\right]^T \tag{A.9}$$

Here $C^{C_2}_{C_1,Nav}$ denotes the DCM transforming from C at $t = t_1$ to C at $t = t_2$, computed by the navigation system of the platform. This matrix differs from the true DCM due to platform navigation errors. The matrix $\hat{R}^{C_2}_{C_1}$ is the estimated rotation matrix. One can verify that under ideal conditions, $C^{C_2}_{C_1,True} = R^{C_2}_{C_1,True}$, and thus the rotation error angle $\mathbf{z}_{Rotation}$ is equal to zero. We omit the subscript (Nav) for simplicity, and write simply $C^{C_2}_{C_1}$.

In general, $C^{C_2}_{C_1}$ can be written as follows:

$$T^{C_2}_{C_1} = T^{B_1}_{C_1}T^{E}_{B_1}T^{B_2}_{E}T^{C_2}_{B_2} \tag{A.10}$$

where the matrices $T^{B_1}_{C_1}$ and $T^{C_2}_{B_2}$ are assumed to be known precisely – or at least with much more precision compared to the developing attitude errors. Thus, $C^{C}_{B} = C^{C}_{B,True}$.

The errors in the ECEF-to-body rotation matrix stem from two sources: position errors and attitude errors. Denote by L_C the *correct* LLLN system at the platform estimated position, and by L the LLLN system estimated by the navigation system. Thus, $C^{E}_{B} = C^{L}_{B}C^{E}_{L_C}$, where C^{L}_{B} is erroneous due to attitude errors and $C^{E}_{L_C}$ is erroneous due to position errors. When these errors are not present, $L = L_C = L_{True}$, where L_{True} is the true LLLN system.

The errors in $C^{E}_{L_C}$ are assumed to be negligible, since they depend on the position errors, which are small relative to Earth's radius. Thus, $L_C = L_{True}$ and therefore $C^{E}_{L_C} = C^{E}_{L_{True}}$. However, the attitude errors do not allow a perfect estimation of the DCM transforming from LLLN to B, since the estimated LLLN system is erroneous. Hence, $C^{L}_{B} = C^{L_C}_{B}C^{L}_{L_C}$.

Assuming small attitude errors, we write $\mathbf{\Psi}_{Nav} = \mathbf{\Psi}_{True} + \Delta\mathbf{\Psi}$ to obtain $C^{L}_{L_C} = I - \Delta\mathbf{\Psi}^{\wedge}$. Taking all the above into consideration, Eq. (A.10) turns into

$$T^{C_2}_{C_1} = T^{B_1}_{C_1}T^{L_{C_1}}_{B_1}\left[I - \Delta\mathbf{\Psi}^{\wedge}(t_1)\right]T^{E}_{L_{C_1}}T^{L_{C_2}}_{E}\left[I + \Delta\mathbf{\Psi}^{\wedge}(t_2)\right]T^{B_2}_{L_{C_2}}T^{C_2}_{B_2} \tag{A.11}$$

For a sufficiently small $t - t_0$ or for a straight and level flight, one can use the approximation of Eq. (1.14): $\Delta\mathbf{\Psi}(t_2) = -C^{B_1}_{L_1}\mathbf{d}\Delta t + \Delta\mathbf{\Psi}(t_1)$. Substituting this into Eq. (A.11), ignoring second-order terms and carrying out some additional algebraic manipulations we get

$$\begin{aligned} T^{C_2}_{C_1} &= T^{B_1}_{C_1}T^{L_{C_1}}_{B_1}T^{E}_{L_{C_1}}T^{L_{C_2}}_{E}T^{B_2}_{L_{C_2}}T^{C_2}_{B_2} + \\ &+ T^{B_1}_{C_1}T^{L_{C_1}}_{B_1}\left[T^{E}_{L_{C_1}}T^{L_{C_2}}_{E}\left(\Delta\mathbf{\Psi}(t_1) - C^{B_1}_{L_1}\mathbf{d}\Delta t\right)^{\wedge} - \Delta\mathbf{\Psi}^{\wedge}(t_1)T^{E}_{L_{C_1}}T^{L_{C_2}}_{E}\right]T^{B_2}_{L_{C_2}}T^{C_2}_{B_2} \end{aligned} \tag{A.12}$$

As was mentioned before, the rotation matrix that was estimated by the image processing module differs from the true matrix. Let $T_{R_{Err}}$ be the DCM transforming between the true rotation matrix and the estimated one: $\hat{R}_{C_1}^{C_2} = T_{R_{Err}} R_{C_1,True}^{C_2}$.

Multiplying Eq. (A.12) by $\left[\hat{R}_{C_1}^{C_2}\right]^T$ from the right and noting that $T_{C_1}^{B_1} T_{B_1}^{L_{C_1}} T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} T_{L_{C_2}}^{B_2} T_{B_2}^{C_2}$ is the nominal value of $C_{C_1,True}^{C_2}$ and hence also of $R_{C_1,True}^{C_2}$ yields

$$\begin{aligned} T_{C_1}^{C_2}\left[\hat{R}_{C_1}^{C_2}\right]^T &= \Big\{ I + T_{C_1}^{B_1} T_{B_1}^{L_{C_1}} \Big[T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} \left(\Delta\boldsymbol{\Psi}(t_1) - C_{L_1}^{B_1}\mathbf{d}\Delta t\right)^{\wedge} \\ &- \Delta\boldsymbol{\Psi}^{\wedge}(t_1) T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} \Big] T_{L_{C_2}}^{B_2} T_{B_2}^{C_2} R_{C_2,True}^{C_1} \Big\} T_{R_{err}}^{T} \end{aligned} \tag{A.13}$$

Assuming small estimation rotation errors $\mathbf{v}_{rot}$, one can write $T_{R_{err}}^{T} = I - \mathbf{v}_{rot}^{\wedge}$. Thus, substituting Eq. (A.13) into Eq. (A.9) yields

$$\begin{aligned} z_{rotation}^{\wedge} = \mathbf{v}_{rot}^{\wedge} + T_{C_1}^{B_1} T_{B_1}^{L_{C_1}} \Big[&-T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} \left(\Delta\boldsymbol{\Psi}(t_1) - C_{L_1}^{B_1}\mathbf{d}\Delta t\right)^{\wedge} \\ &+\Delta\boldsymbol{\Psi}^{\wedge}(t_1) T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} \Big] T_{L_{C_2}}^{B_2} T_{B_2}^{C_2} \left[\hat{R}_{C_1}^{C_2}\right]^T \end{aligned} \tag{A.14}$$

Using Eq. (A.10), one can write the following two relations:

$$T_{C_1}^{B_1} T_{B_1}^{L_{C_1}} T_{L_{C_1}}^{E} T_{E}^{L_{C_2}} = \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} \quad , \quad T_{C_1}^{B_1} T_{B_1}^{L_{C_1}} = \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} C_{L_{C_2}}^{E} C_{E}^{L_{C_1}} \tag{A.15}$$

Substituting Eqs. (A.15) into Eq. (A.14) and using the fact that for any matrix Λ and any vector $\boldsymbol{\xi}$, $\Lambda\boldsymbol{\xi}^{\wedge}\Lambda^T = (\Lambda\boldsymbol{\xi})^{\wedge}$, Eq. (A.14) transforms into

$$\begin{aligned} z_{rotation} &= \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} \left(C_{L_{C_2}}^{E} C_{E}^{L_{C_1}} - I \right) \Delta\boldsymbol{\Psi}(t_1) + \\ &+ \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} C_{L_1}^{B_1} \mathbf{d}\Delta t + \mathbf{v}_{rot} \end{aligned} \tag{A.16}$$

One can see that Eq. (A.16) is of the form $\mathbf{z}_{rotation} = H^{rot}\mathbf{X} + \mathbf{v}_{rot}$, where

$$H^{rot} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & H_{\Delta\Psi}^{rot} & H_{d}^{rot} & 0_{3\times3} \end{bmatrix}$$

$$H_{\Delta\Psi}^{rot} = \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} \left(C_{L_{C_2}}^{E} C_{E}^{L_{C_1}} - I \right) \quad , \quad H_{d}^{rot} = \hat{R}_{C_1}^{C_2} C_{C_2}^{B_2} C_{B_2}^{L_{C_2}} C_{L_1}^{B_1} \Delta t$$

A.3 Google Earth Interface

Given a platform trajectory and measurement settings (such as measurement frequency), a command is sent to Google Earth through the interface to display a region at a specified position (latitude, longitude and altitude) and inertial orientation. These are computed based on the current platform position, attitude and camera angles.

Since the current version of Google Earth allows changing only the camera heading and tilt angles, special care was taken to allow roll motion in Google Earth, that is

required for implementing the camera scanning procedure. This was achieved by shifting the yaw angle by 90^o relative to the flight heading angle and adjusting the camera system accordingly. As a result, camera/platform roll motion is obtained through tilt motion (handled automatically by the interface).

In the current implementation, the image acquisition through Google Earth is performed offline, i. e., this command is sent according to the measurement's frequency and the acquired images are saved into some repository. The images are injected into the image processing module in the simulation at appropriate time instances.

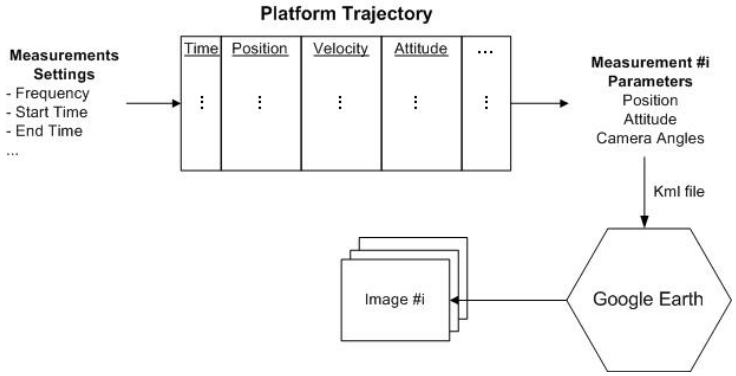


Figure A.1: A schematic illustration of an interface between the platform trajectory and Google Earth.

Appendix B

Chapter 3 Extras

B.1 Proof of Theorem 3.2.1

Recall the matrix A,

$$A = \begin{bmatrix} \mathbf{q}_1 & -\mathbf{q}_2 & \mathbf{0}_{3\times 1} & -\mathbf{T}_{12} \\ \mathbf{0}_{3\times 1} & \mathbf{q}_2 & -\mathbf{q}_3 & -\mathbf{T}_{23} \end{bmatrix} \in \mathbb{R}^{6\times 4} \tag{B.1}$$

and the constraints

$$\mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) = 0 \tag{B.2a}$$
$$\mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) = 0 \tag{B.2b}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \tag{B.2c}$$

Next we prove that the constraints (B.2) hold if and only if $\text{rank}(A) < 4$.

B.1.1 $\text{rank}(A) < 4 \Rightarrow$ Eqs. (B.2)

Since $\text{rank}(A) < 4$, there exists a nonzero vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ such that $A\boldsymbol{\beta} = \mathbf{0}$. The explicit equations stemming from $A\boldsymbol{\beta} = \mathbf{0}$ are

$$\mathbf{q}_1\beta_1 - \mathbf{q}_2\beta_2 - \mathbf{T}_{12}\beta_4 = \mathbf{0} \tag{B.3}$$
$$\mathbf{q}_2\beta_2 - \mathbf{q}_3\beta_3 - \mathbf{T}_{23}\beta_4 = \mathbf{0} \tag{B.4}$$

Cross-multiplying Eq. (B.3) by $\mathbf{q}_1$ and Eq. (B.4) by $\mathbf{q}_3$ yields

$$(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4 = \mathbf{0} \tag{B.5}$$
$$(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 = \mathbf{0} \tag{B.6}$$

If $\mathbf{q}_1 \times \mathbf{q}_2 \neq \mathbf{0}$ and $\mathbf{q}_3 \times \mathbf{q}_2 \neq \mathbf{0}$, then performing an inner product of Eq. (B.5) with $(\mathbf{q}_3 \times \mathbf{q}_2)$ and of Eq. (B.6) with $(\mathbf{q}_1 \times \mathbf{q}_2)$ yields

$$(\mathbf{q}_3 \times \mathbf{q}_2)^T(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_3 \times \mathbf{q}_2)^T(\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4 = 0 \tag{B.7}$$
$$(\mathbf{q}_1 \times \mathbf{q}_2)^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_1 \times \mathbf{q}_2)^T(\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 = 0 \tag{B.8}$$

Noting that $(\mathbf{q}_3 \times \mathbf{q}_2)^T(\mathbf{q}_1 \times \mathbf{q}_2) = -(\mathbf{q}_1 \times \mathbf{q}_2)^T(\mathbf{q}_3 \times \mathbf{q}_2)$ and adding Eqs. (B.7) and (B.8) gives the constraint (B.2c).

The first two constraints may be obtained similarly: Cross-multiplying Eq. (B.3) by $\mathbf{q}_2$ and then taking an inner product with $\mathbf{q}_1$ gives the constraint (B.2a). Cross-multiplying from the right Eq. (B.4) by $\mathbf{q}_3$ and then taking an inner product with $\mathbf{q}_2$ gives the constraint (B.2b). ■

Degenerate Cases

$\mathbf{q}_1 \times \mathbf{q}_2 = \mathbf{0}$ or $\mathbf{q}_3 \times \mathbf{q}_2 = \mathbf{0}$, or both, i. e. $\mathbf{q}_1 \parallel \mathbf{q}_2$ or $\mathbf{q}_2 \parallel \mathbf{q}_3$, or $\mathbf{q}_1 \parallel \mathbf{q}_2 \parallel \mathbf{q}_3$. Consider the case $\mathbf{q}_1 \parallel \mathbf{q}_2$. Since both $\mathbf{q}_1$ and $\mathbf{q}_2$ point to the same ground point, it may be concluded that $\mathbf{T}_{12}$ is parallel to $\mathbf{q}_1$ and $\mathbf{q}_2$. More formally, if r_1 and r_2 are the scale parameters such that $||r_i\mathbf{q}_i||$ is the range to the ground point, then $\mathbf{T}_{12} = r_2\mathbf{q}_2 - r_1\mathbf{q}_1 = r_2 a\mathbf{q}_1 - r_1\mathbf{q}_1 = (r_2 a - r_1)\mathbf{q}_1$, where a is a constant. Hence $\mathbf{T}_{12} \parallel \mathbf{q}_1 \parallel \mathbf{q}_2$. Consequently, Eq. (B.2b) is the only constraint from the three constraints in Eq. (B.2) that is not degenerate. This constraint may be obtained as explained above. The case $\mathbf{q}_2 \parallel \mathbf{q}_3$ is handled in a similar manner.

The last degenerated case is $\mathbf{q}_1 \parallel \mathbf{q}_2 \parallel \mathbf{q}_3$, which occurs when the vehicle moves along the line of sight vectors. In this case all the constraints in Eq. (B.2) are degenerate.

Note that in the first two degenerate cases ($\mathbf{q}_1 \parallel \mathbf{q}_2$ or $\mathbf{q}_2 \parallel \mathbf{q}_3$), it is possible to write another set of three constraints. For example, if $\mathbf{q}_1 \parallel \mathbf{q}_2$ (but not to $\mathbf{q}_3$), we can formulate two epipolar constraints between views 1 and 3, and between views 2 and 3, and provide the equivalent constraint to Eq. (B.2c) relating between $\mathbf{T}_{13}$ and $\mathbf{T}_{23}$.

B.1.2 Eqs. (B.2) $\Rightarrow \text{rank}(A) < 4$

The proof is based on steps similar to the previous section, in a reverse order. Recall the constraint (B.2c), multiplied by some constant $\beta_4 \neq 0$:

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_4 \tag{B.9}$$

Since $(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{q}_2)$ is a scalar and Eq. (B.9) is a scalar equation, there exists some $\beta_2 \neq 0$ such that

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 = (\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 \tag{B.10}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_4 \tag{B.11}$$

The above equation may be rewritten into

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 = 0 \tag{B.12}$$
$$(\mathbf{q}_3 \times \mathbf{q}_2)^T(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_3 \times \mathbf{q}_2)^T(\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4 = 0 \tag{B.13}$$

or equivalently

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T [(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4] = 0 \quad (B.14)$$
$$(\mathbf{q}_3 \times \mathbf{q}_2)^T [(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4] = 0 \quad (B.15)$$

At this point it is assumed that $\mathbf{q}_1 \times \mathbf{q}_2 \neq \mathbf{0}$ and $\mathbf{q}_3 \times \mathbf{q}_2 \neq \mathbf{0}$. The proof for cases in which this assumption does not hold is given in the sequel.

Noting that $\mathbf{q}_2^T(\mathbf{q}_3 \times \mathbf{q}_2) \equiv 0$, and since the constraint (B.2b) is satisfied, the vectors $(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4$ and $(\mathbf{q}_2 \times \mathbf{q}_1)$ are not perpendicular. In the same manner, since $\mathbf{q}_2^T(\mathbf{q}_1 \times \mathbf{q}_2) = 0$ and the constraint (B.2a) is met, the vectors $(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4$ and $(\mathbf{q}_3 \times \mathbf{q}_2)$ are not perpendicular as well. Therefore the last two equations lead to

$$(\mathbf{q}_3 \times \mathbf{q}_2)\beta_2 - (\mathbf{q}_3 \times \mathbf{T}_{23})\beta_4 = \mathbf{0} \quad (B.16)$$
$$(\mathbf{q}_1 \times \mathbf{q}_2)\beta_2 + (\mathbf{q}_1 \times \mathbf{T}_{12})\beta_4 = \mathbf{0} \quad (B.17)$$

that me by rewritten as

$$\mathbf{q}_3 \times (\mathbf{q}_2\beta_2 - \mathbf{T}_{23}\beta_4 + \mathbf{q}_3\beta_3) = \mathbf{0} \quad (B.18)$$
$$\mathbf{q}_1 \times (\mathbf{q}_2\beta_2 + \mathbf{T}_{12}\beta_4 + \mathbf{q}_1\beta_1) = \mathbf{0} \quad (B.19)$$

for any β_1, β_3. Consequently,

$$\mathbf{q}_2\beta_2 + \mathbf{q}_3\beta_3 - \mathbf{T}_{23}\beta_4 = \mathbf{0} \quad (B.20)$$
$$\mathbf{q}_1\beta_1 + \mathbf{q}_2\beta_2 + \mathbf{T}_{12}\beta_4 = \mathbf{0} \quad (B.21)$$

In order to obtain the same expression for the matrix A, the vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is defined as

$$\alpha_1 \doteq \beta_1 \quad , \quad \alpha_2 \doteq -\beta_2 \quad , \quad \alpha_3 \doteq \beta_3 \quad , \quad \alpha_4 \doteq -\beta_4 \quad (B.22)$$

so that Eqs. (B.20) and (B.21) turn into

$$-\mathbf{q}_2\alpha_2 + \mathbf{q}_3\alpha_3 + \mathbf{T}_{23}\alpha_4 = \mathbf{0} \quad (B.23)$$
$$\mathbf{q}_1\alpha_1 - \mathbf{q}_2\alpha_2 - \mathbf{T}_{12}\alpha_4 = \mathbf{0} \quad (B.24)$$

The above may be rewritten as

$$A\boldsymbol{\alpha} = \mathbf{0} \quad (B.25)$$

and since $\boldsymbol{\alpha}$ is a nonzero vector, one may conclude that $\text{rank}(A) < 4$. ■

Note that the epipolar constraints (B.2a) and (B.2b) only guarantee that the matrices $\begin{bmatrix}\mathbf{q}_1 & -\mathbf{q}_2 & -\mathbf{T}_{12}\end{bmatrix}$ and $\begin{bmatrix}\mathbf{q}_2 & -\mathbf{q}_3 & -\mathbf{T}_{23}\end{bmatrix}$ are singular, which not necessarily leads to $\text{rank}(A) < 4$.

Degenerate Cases

Next we prove that $\text{rank}(A) < 4$ also when $\mathbf{q}_1 \parallel \mathbf{q}_2$ or $\mathbf{q}_2 \parallel \mathbf{q}_3$, or $\mathbf{q}_1 \parallel \mathbf{q}_2 \parallel \mathbf{q}_3$.

Let $\mathbf{q}_1 \parallel \mathbf{q}_2$ while $\mathbf{q}_3$ is not parallel to $\mathbf{q}_1$. As proven above, $\mathbf{q}_1 \parallel \mathbf{q}_2 \parallel \mathbf{T}_{12}$, and thus, the matrix A is of the form

$$A = \begin{bmatrix} \mathbf{q}_1 & a\mathbf{q}_1 & \mathbf{0}_{3\times 1} & b\mathbf{q}_1 \\ \mathbf{0}_{3\times 1} & a\mathbf{q}_1 & \mathbf{q}_3 & \mathbf{T}_{23} \end{bmatrix} \tag{B.26}$$

for some scalars a, b. In order to prove that $\text{rank}(A) < 4$, we need to show that $A\boldsymbol{\beta} = \mathbf{0}$ for some nonzero vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4)^T$. Assume a general vector $\boldsymbol{\beta}$ and explicitly write $A\boldsymbol{\beta} = 0$:

$$\mathbf{q}_1\beta_1 + a\mathbf{q}_1\beta_2 + b\mathbf{q}_1\beta_4 = 0 \tag{B.27}$$

$$a\mathbf{q}_1\beta_2 + \mathbf{q}_3\beta_3 + \mathbf{T}_{23}\beta_4 = 0 \tag{B.28}$$

Observe that the second equation leads to the epipolar constraint $\mathbf{q}_2^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = 0$. Since the constraints (B.2) hold, it follows that the matrix $\begin{bmatrix} \mathbf{q}_2 & -\mathbf{q}_3 & -\mathbf{T}_{23} \end{bmatrix}$ is singular, and since $\mathbf{q}_2 = a\mathbf{q}_1$, it is possible to find nonzero entries for β_2, β_3 and β_4 so that Eq. (B.28) is satisfied. From Eq. (B.27) it is easy to see that $\beta_1 = -a\beta_2 - b\beta_4$. Thus, a nonzero vector β was found such that $A\boldsymbol{\beta} = \mathbf{0}$, which leads to the conclusion that $\text{rank}(A) < 4$. A similar procedure may be applied when $\mathbf{q}_2 \parallel \mathbf{q}_3$ while $\mathbf{q}_1$ is not parallel to $\mathbf{q}_2$.

The last degenerate case is when all the three vectors are parallel. As already mentioned, both of the translation vectors in this case are parallel to the line of sight vectors, i. e. $\mathbf{q}_1 \parallel \mathbf{q}_2 \parallel \mathbf{q}_3 \parallel \mathbf{T}_{12} \parallel \mathbf{T}_{23}$. The matrix A is then of the following form:

$$A = \begin{bmatrix} \mathbf{q}_1 & a\mathbf{q}_1 & \mathbf{0}_{3\times 1} & b\mathbf{q}_1 \\ \mathbf{0}_{3\times 1} & -a\mathbf{q}_1 & c\mathbf{q}_1 & d\mathbf{q}_1 \end{bmatrix} \tag{B.29}$$

where a, b, c and d are some constants. Since one may find some nonzero vector $\boldsymbol{\beta}$ such that $A\boldsymbol{\beta} = \mathbf{0}$, (e. g. $\boldsymbol{\beta} = (b, 0, d/c, -1)^T$), the conclusion is that $\text{rank}(A) < 4$.

B.1.3 Alternative Proof of $\text{rank}(A) < 4 \Rightarrow$ Eqs. (B.2)

Since $\text{rank}(A) < 4$, the determinant of any 4×4 submatrix of A should be equal to zero. A careful examination of all such possible submatrices of A will give a complete set of constraints derived from a general three-view geometry.

All the 4×4 submatrices of A comprised of the first three rows with any of the other rows of A yield the epipolar constraint for the first two views, i. e., Eq. (B.2a):

$$\mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) = 0 \tag{B.30}$$

In the same manner, the last three rows, with any of the first three rows of A, provide the epipolar constraint for the second and the third views, i. e., Eq. (B.2b):

$$\mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) = 0 \tag{B.31}$$

The more interesting result, however, stems from analyzing the determinants of all the other possible submatrices of A. There are nine such submatrices, which consist of the following row permutations of A:

$$\begin{array}{lll} 1,\ 2,\ 4,\ 5\ ; & 1,\ 2,\ 4,\ 6\ ; & 1,\ 2,\ 5,\ 6\ ; \\ 2,\ 3,\ 4,\ 5\ ; & 2,\ 3,\ 4,\ 6\ ; & 2,\ 3,\ 5,\ 6\ ; \\ 1,\ 3,\ 4,\ 5\ ; & 1,\ 3,\ 4,\ 6\ ; & 1,\ 3,\ 5,\ 6\ . \end{array} \tag{B.32}$$

Let

$$E_1 \doteq \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad E_2 \doteq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad E_3 \doteq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{B.33}$$

so that the matrix $E_i\mathbf{a}$ removes the ith element of any $\mathbf{a} \in \mathbb{R}^{3\times 1}$. Consider, for example, the submatrix comprised of rows $1, 2, 4, 5$ of A. This 4×4 matrix, denoted by $\tilde{A}$, is

$$\tilde{A} = \begin{bmatrix} E_3\mathbf{q}_1 & -E_3\mathbf{q}_2 & \mathbf{0}_{2\times 1} & -E_3\mathbf{T}_{12} \\ \mathbf{0}_{2\times 1} & E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 & -E_3\mathbf{T}_{23} \end{bmatrix} \tag{B.34}$$

Therefore

$$\det(\tilde{A}) = (\mathbf{q}_1)_1 \begin{vmatrix} -(\mathbf{q}_2)_2 & 0 & -(\mathbf{T}_{12})_2 \\ E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 & -E_3\mathbf{T}_{23} \end{vmatrix} - (\mathbf{q}_1)_2 \begin{vmatrix} -(\mathbf{q}_2)_1 & 0 & -(\mathbf{T}_{12})_1 \\ E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 & -E_3\mathbf{T}_{23} \end{vmatrix} = 0 \tag{B.35}$$

where $(\mathbf{a})_i$ denotes the ith element of some vector $\mathbf{a}$. After some algebra, we obtain

$$\left|\begin{bmatrix} E_3\mathbf{q}_1 & -E_3\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_3 & E_3\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_3\mathbf{q}_1 & E_3\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 \end{matrix}\right| \tag{B.36}$$

Similarly, the remaining submatrices (cf. Eq. (B.32)) yield

$$\left|\begin{bmatrix} E_1\mathbf{q}_1 & -E_1\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_3 & E_3\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_1\mathbf{q}_1 & E_1\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 \end{matrix}\right| \tag{B.37}$$

$$\left|\begin{bmatrix} E_2\mathbf{q}_1 & -E_2\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_3 & E_3\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_2\mathbf{q}_1 & E_2\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_3\mathbf{q}_2 & -E_3\mathbf{q}_3 \end{matrix}\right| \tag{B.38}$$

$$\left|\begin{bmatrix} E_3\mathbf{q}_1 & -E_3\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_3 & E_1\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_3\mathbf{q}_1 & E_3\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_2 & -E_1\mathbf{q}_3 \end{matrix}\right| \tag{B.39}$$

$$\left|\begin{bmatrix} E_1\mathbf{q}_1 & -E_1\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_3 & E_1\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_1\mathbf{q}_1 & E_1\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_2 & -E_1\mathbf{q}_3 \end{matrix}\right| \tag{B.40}$$

$$\left|\begin{bmatrix} E_2\mathbf{q}_1 & -E_2\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_3 & E_1\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_2\mathbf{q}_1 & E_2\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_1\mathbf{q}_2 & -E_1\mathbf{q}_3 \end{matrix}\right| \tag{B.41}$$

$$\left|\begin{bmatrix} E_3\mathbf{q}_1 & -E_3\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_3 & E_2\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_3\mathbf{q}_1 & E_3\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_2 & -E_2\mathbf{q}_3 \end{matrix}\right| \tag{B.42}$$

$$\left|\begin{bmatrix} E_1\mathbf{q}_1 & -E_1\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_3 & E_2\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_1\mathbf{q}_1 & E_1\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_2 & -E_2\mathbf{q}_3 \end{matrix}\right| \tag{B.43}$$

$$\left|\begin{bmatrix} E_2\mathbf{q}_1 & -E_2\mathbf{q}_2 \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_3 & E_2\mathbf{T}_{23} \end{matrix}\right| = \left|\begin{bmatrix} E_2\mathbf{q}_1 & E_2\mathbf{T}_{12} \end{bmatrix}^T\right| \left|\begin{matrix} E_2\mathbf{q}_2 & -E_2\mathbf{q}_3 \end{matrix}\right| \tag{B.44}$$

However, although there are nine scalar constraints, given by Eqs. (B.36-B.44), only *one* scalar constraint is actually contributed. To see this, write a compact version of the above equations:

$$\begin{array}{lll} a_1b_1 = c_1d_1 & a_1b_2 = c_1d_2 & a_1b_3 = c_1d_3 \\ a_2b_1 = c_2d_1 & a_2b_2 = c_2d_2 & a_2b_3 = c_2d_3 \\ a_3b_1 = c_3d_1 & a_3b_2 = c_3d_2 & a_3b_3 = c_3d_3 \end{array}$$

Since all the parameters are scalar, it may be concluded that

$$\begin{array}{rcl} \exists k & : & ka_2 = a_1 \ , \ kc_2 = c_1 \\ \exists \tilde{k} & : & \tilde{k}a_3 = a_1 \ , \ \tilde{k}c_3 = c_1 \\ \exists n & : & nb_2 = b_1 \ , \ nd_2 = d_1 \\ \exists \tilde{n} & : & \tilde{n}b_3 = b_1 \ , \ \tilde{n}d_3 = d_1 \end{array}$$

From here it follows that all the nine equations are equivalent. For example, multiplying the equation $a_3b_3 = c_3d_3$ by $\tilde{k}\tilde{n}$ yields the equation $a_1b_1 = c_1d_1$:

$$\tilde{k}a_3\tilde{n}b_3 = \tilde{k}c_3\tilde{n}d_3 \quad \Leftrightarrow \quad a_1b_1 = c_1d_1$$

Thus, all the row permutations that do not yield the epipolar constraint give a single scalar constraint. This scalar constraint may be expressed in a more compact form, which is developed next.

Reformulating the Scalar Constraint

The scalar constraint induced by any of the nine equations (or a combination thereof) may be written in a compact form. For example, consider Eq. (B.36). Since

$$\begin{array}{rcl} \left|\begin{bmatrix} E_3\mathbf{q}_1 & -E_3\mathbf{q}_2 \end{bmatrix}^T\right| & = & (\mathbf{q}_2 \times \mathbf{q}_1)_3 \\ \left|E_3\mathbf{q}_3 \quad E_3\mathbf{T}_{23}\right| & = & (\mathbf{q}_3 \times \mathbf{T}_{23})_3 \\ \left|\begin{bmatrix} E_3\mathbf{q}_1 & E_3\mathbf{T}_{12} \end{bmatrix}^T\right| & = & (\mathbf{q}_1 \times \mathbf{T}_{12})_3 \\ \left|E_3\mathbf{q}_2 \quad -E_3\mathbf{q}_3\right| & = & (\mathbf{q}_3 \times \mathbf{q}_2)_3 \end{array}$$

It follows that Eq. (B.36) can be rewritten as

$$(\mathbf{q}_2 \times \mathbf{q}_1)_3(\mathbf{q}_3 \times \mathbf{T}_{23})_3 = (\mathbf{q}_1 \times \mathbf{T}_{12})_3(\mathbf{q}_3 \times \mathbf{q}_2)_3 \tag{B.45}$$

Following the same procedure, it is possible to express Eqs. (B.37-B.44) in terms of vector cross products:

$$(\mathbf{q}_2 \times \mathbf{q}_1)_1(\mathbf{q}_3 \times \mathbf{T}_{23})_3 = (\mathbf{q}_1 \times \mathbf{T}_{12})_1(\mathbf{q}_3 \times \mathbf{q}_2)_3 \tag{B.46}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_2(\mathbf{q}_3 \times \mathbf{T}_{23})_3 = (\mathbf{q}_1 \times \mathbf{T}_{12})_2(\mathbf{q}_3 \times \mathbf{q}_2)_3 \tag{B.47}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_3(\mathbf{q}_3 \times \mathbf{T}_{23})_1 = (\mathbf{q}_1 \times \mathbf{T}_{12})_3(\mathbf{q}_3 \times \mathbf{q}_2)_1 \tag{B.48}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_1(\mathbf{q}_3 \times \mathbf{T}_{23})_1 = (\mathbf{q}_1 \times \mathbf{T}_{12})_1(\mathbf{q}_3 \times \mathbf{q}_2)_1 \tag{B.49}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_2(\mathbf{q}_3 \times \mathbf{T}_{23})_1 = (\mathbf{q}_1 \times \mathbf{T}_{12})_2(\mathbf{q}_3 \times \mathbf{q}_2)_1 \tag{B.50}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_3(\mathbf{q}_3 \times \mathbf{T}_{23})_2 = (\mathbf{q}_1 \times \mathbf{T}_{12})_3(\mathbf{q}_3 \times \mathbf{q}_2)_2 \tag{B.51}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_1(\mathbf{q}_3 \times \mathbf{T}_{23})_2 = (\mathbf{q}_1 \times \mathbf{T}_{12})_1(\mathbf{q}_3 \times \mathbf{q}_2)_2 \tag{B.52}$$
$$(\mathbf{q}_2 \times \mathbf{q}_1)_2(\mathbf{q}_3 \times \mathbf{T}_{23})_2 = (\mathbf{q}_1 \times \mathbf{T}_{12})_2(\mathbf{q}_3 \times \mathbf{q}_2)_2 \tag{B.53}$$

Eqs. (B.45-B.53) contribute only one constraint, which may be also written as any linear combination of these equations. In particular, adding Eqs. (B.49), (B.53), and (B.45) yields

$$\begin{aligned}(\mathbf{q}_2 \times \mathbf{q}_1)_1(\mathbf{q}_3 \times \mathbf{T}_{23})_1 + (\mathbf{q}_2 \times \mathbf{q}_1)_2(\mathbf{q}_3 \times \mathbf{T}_{23})_2 + (\mathbf{q}_2 \times \mathbf{q}_1)_3(\mathbf{q}_3 \times \mathbf{T}_{23})_3 = \\ (\mathbf{q}_1 \times \mathbf{T}_{12})_1(\mathbf{q}_3 \times \mathbf{q}_2)_1 + (\mathbf{q}_1 \times \mathbf{T}_{12})_2(\mathbf{q}_3 \times \mathbf{q}_2)_2 + (\mathbf{q}_1 \times \mathbf{T}_{12})_3(\mathbf{q}_3 \times \mathbf{q}_2)_3\end{aligned}$$

which can be written as Eq. (B.2c):

$$(\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) = (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \tag{B.54}$$

B.2 IEKF Matrices

In this appendix we present the development of the IEKF matrices H_3, H_2, H_1, D and R. Recall the residual measurement definition (cf. Eqs. (3.20), (3.21) and (3.23))

$$\begin{aligned}\mathbf{z}_{N\times 1} &= \mathcal{A}\mathbf{T}_{23} - \mathcal{B}\mathbf{T}_{12} = \\ &= \mathbf{h}\left(\mathbf{Pos}(t_3), \boldsymbol{\Psi}(t_3), \mathbf{Pos}(t_2), \boldsymbol{\Psi}(t_2), \mathbf{Pos}(t_1), \boldsymbol{\Psi}(t_1), \left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}\right) \\ &\approx H_3\mathbf{X}(t_3) + H_2\mathbf{X}(t_2) + H_1\mathbf{X}(t_1) + D\mathbf{v}\end{aligned} \tag{B.55}$$

where

$$\mathcal{A} \doteq \begin{bmatrix} U \\ F \\ 0 \end{bmatrix}_{N\times 3} \quad , \quad \mathcal{B} \doteq \begin{bmatrix} W \\ 0 \\ G \end{bmatrix}_{N\times 3} \tag{B.56}$$

and $\mathbf{X}$ is the state vector defined in Eq. (1.10):

$$\mathbf{X}_{15\times 1} = \begin{bmatrix}\Delta\mathbf{P}^T & \Delta\mathbf{V}^T & \Delta\boldsymbol{\Psi}^T & \mathbf{d}^T & \mathbf{b}^T\end{bmatrix}^T \tag{B.57}$$

Recall also that the time instant of the third image, t_3, of the three overlapping images is the current time. Therefore, in Eq. (B.55) $\mathbf{X}(t_3)$ is the state vector to be estimated, while $\tilde{\mathbf{X}}(t_2) = \mathbf{X}(t_2)$ and $\tilde{\mathbf{X}}(t_1) = \mathbf{X}(t_1)$ are the estimation errors at the first two time instances represented by the filter covariance attached to each image. These last two terms, accompanied by the Jacobian matrices H_2 and H_1 and the image noise $\mathbf{v}$ along with the Jacobian matrix D, constitute the measurement noise. Since navigation and imagery information is independent of each other, these two sources of information will be analyzed separately.

B.2.1 Calculation of the Matrices H_3, H_2 and H_1

The matrices H_3, H_2 and H_1, are $N \times 15$ and are defined as

$$H_3 \doteq \nabla_{\boldsymbol{\zeta}(t_3)}\mathbf{h} \quad , \quad H_2 \doteq \nabla_{\boldsymbol{\zeta}(t_2)}\mathbf{h} \quad , \quad H_1 \doteq \nabla_{\boldsymbol{\zeta}(t_1)}\mathbf{h} \tag{B.58}$$

where $\boldsymbol{\zeta}$ is defined in Eq. (1.3).
From Eq. (B.55) it is clear that these matrices are of the following form:

$$H_i = \begin{bmatrix} H^{\mathbf{Pos}(t_i)} & 0 & H^{\boldsymbol{\Psi}(t_i)} & 0 & 0 \end{bmatrix} \tag{B.59}$$

with $i = 1, 2, 3$. Since $\mathbf{T}_{23} = \mathbf{Pos}(t_3) - \mathbf{Pos}(t_2)$ and $\mathbf{T}_{12} = \mathbf{Pos}(t_2) - \mathbf{Pos}(t_1)$,

$$H^{\mathbf{Pos}(t_3)} = \mathcal{A} \tag{B.60}$$
$$H^{\mathbf{Pos}(t_2)} = -(\mathcal{A} + \mathcal{B}) \tag{B.61}$$
$$H^{\mathbf{Pos}(t_1)} = \mathcal{B} \tag{B.62}$$

Note that the influence of position errors on the LOS vectors that appear in the matrices $\mathcal{A}$ and $\mathcal{B}$ is neglected: the position errors affect only the rotation matrices transforming the LOS vectors to the LLLN system at t_2. These errors are divided by the Earth radius, and therefore their contribution is insignificant.

Calculation of $H^{\boldsymbol{\Psi}(t_3)}, H^{\boldsymbol{\Psi}(t_2)}$ and $H^{\boldsymbol{\Psi}(t_1)}$

Recall the definition of the matrices F, G, U and W

$$U = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_{N_{123}} \end{bmatrix}^T \quad F = \begin{bmatrix} \mathbf{f}_1 & \dots & \mathbf{f}_{N_{23}} \end{bmatrix}^T \tag{B.63}$$
$$W = \begin{bmatrix} \mathbf{w}_1 & \dots & \mathbf{w}_{N_{123}} \end{bmatrix}^T \quad G = \begin{bmatrix} \mathbf{g}_1 & \dots & \mathbf{g}_{N_{12}} \end{bmatrix}^T \tag{B.64}$$

with

$$\mathbf{u}_i = \mathbf{u}_i(\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}) = -\left[\mathbf{q}_{3_i}\right]_\times \left[\mathbf{q}_{1_i}\right]_\times \mathbf{q}_{2_i} \tag{B.65a}$$
$$\mathbf{w}_i = \mathbf{w}_i(\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}) = -\left[\mathbf{q}_{1_i}\right]_\times \left[\mathbf{q}_{2_i}\right]_\times \mathbf{q}_{3_i} \tag{B.65b}$$
$$\mathbf{f}_i = \mathbf{f}_i(\mathbf{q}_{2_i}, \mathbf{q}_{3_i}) = \left[\mathbf{q}_{2_i}\right]_\times \mathbf{q}_{3_i} \tag{B.65c}$$
$$\mathbf{g}_i = \mathbf{g}_i(\mathbf{q}_{1_i}, \mathbf{q}_{2_i}) = \left[\mathbf{q}_{1_i}\right]_\times \mathbf{q}_{2_i} \tag{B.65d}$$

Since the development of expressions for the matrices $H^{\mathbf{\Psi}(t_3)}$, $H^{\mathbf{\Psi}(t_2)}$ and $H^{\mathbf{\Psi}(t_1)}$ is similar, we elaborate only on the development process of $H^{\mathbf{\Psi}(t_3)}$. This matrix is given by

$$H^{\mathbf{\Psi}(t_3)} = \nabla_{\mathbf{\Psi}(t_3)}\mathbf{h} = \nabla_{\mathbf{\Psi}(t_3)}\left[\mathcal{A}\mathbf{T}_{23}\right] - \nabla_{\mathbf{\Psi}(t_3)}\left[\mathcal{B}\mathbf{T}_{12}\right] \tag{B.66}$$

We start by developing the first term in Eq. (B.66). According to the structure of the matrices U and F, the following may be written:

$$\nabla_{\mathbf{\Psi}(t_3)}\left[\mathcal{A}\mathbf{T}_{23}\right] = \sum_{i=1}^{N_{123}} \frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{u}_i}\nabla_{\mathbf{\Psi}(t_3)}\mathbf{u}_i + \sum_{i=1}^{N_{23}} \frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{f}_i}\nabla_{\mathbf{\Psi}(t_3)}\mathbf{f}_i \tag{B.67}$$

Since $\mathbf{u}_i$ and $\mathbf{f}_i$ are independent of each other, and $\frac{\partial \mathbf{x}^T\mathbf{T}_{23}}{\partial \mathbf{x}_i} = \mathbf{T}_{23}^T$ for any vector $\mathbf{x}$, we have

$$\frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{u}_i} = \mathbf{e}_i\mathbf{T}_{23}^T \tag{B.68}$$

$$\frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{f}_i} = \mathbf{e}_{N_{123}+i}\mathbf{T}_{23}^T \tag{B.69}$$

where $\mathbf{e}_j$ is a $N \times 1$ vector that is comprised of zero entries except for the jth element which is equal to one. Note also that the size of the matrices $\frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{u}_i}$, $\frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{f}_i}$ is $N \times 3$. The remaining quantities in Eq. (B.67), $\nabla_{\mathbf{\Psi}(t_3)}\mathbf{u}_i$ and $\nabla_{\mathbf{\Psi}(t_3)}\mathbf{f}_i$, can be calculated as

$$\nabla_{\mathbf{\Psi}(t_3)}\mathbf{u}_i = \frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{3_i}}\nabla_{\mathbf{\Psi}(t_3)}\mathbf{q}_{3_i} \tag{B.70}$$

$$\nabla_{\mathbf{\Psi}(t_3)}\mathbf{f}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{3_i}}\nabla_{\mathbf{\Psi}(t_3)}\mathbf{q}_{3_i} \tag{B.71}$$

here $\mathbf{q}_{3_i}$ refers to the LOS vector of the ith feature in the third view.

Analytical expressions for $\frac{\partial \mathbf{f}}{\partial \mathbf{q}_j}$, $\frac{\partial \mathbf{g}}{\partial \mathbf{q}_j}$, $\frac{\partial \mathbf{u}}{\partial \mathbf{q}_j}$, $\frac{\partial \mathbf{w}}{\partial \mathbf{q}_j}$, for $j = 1, 2, 3$, are easily obtained based on Eq. (B.65) as

$$\begin{array}{llllllllll}
\frac{\partial \mathbf{u}}{\partial \mathbf{q}_1} = & [\mathbf{q}_3]_\times [\mathbf{q}_2]_\times & \frac{\partial \mathbf{w}}{\partial \mathbf{q}_1} = & \left[[\mathbf{q}_2]_\times \mathbf{q}_3\right]_\times & \frac{\partial \mathbf{f}}{\partial \mathbf{q}_1} = 0_{3\times 3} & \frac{\partial \mathbf{g}}{\partial \mathbf{q}_1} = & -[\mathbf{q}_2]_\times \\
\frac{\partial \mathbf{u}}{\partial \mathbf{q}_2} = & -[\mathbf{q}_3]_\times [\mathbf{q}_1]_\times & \frac{\partial \mathbf{w}}{\partial \mathbf{q}_2} = & [\mathbf{q}_1]_\times [\mathbf{q}_3]_\times & \frac{\partial \mathbf{f}}{\partial \mathbf{q}_2} = -[\mathbf{q}_3]_\times & \frac{\partial \mathbf{g}}{\partial \mathbf{q}_2} = & [\mathbf{q}_1]_\times \\
\frac{\partial \mathbf{u}}{\partial \mathbf{q}_3} = & \left[[\mathbf{q}_1]_\times \mathbf{q}_2\right]_\times & \frac{\partial \mathbf{w}}{\partial \mathbf{q}_3} = & -[\mathbf{q}_1]_\times [\mathbf{q}_2]_\times & \frac{\partial \mathbf{f}}{\partial \mathbf{q}_3} = [\mathbf{q}_2]_\times & \frac{\partial \mathbf{g}}{\partial \mathbf{q}_3} = & 0_{3\times 3}
\end{array} \tag{B.72}$$

As for $\nabla_{\mathbf{\Psi}(t_3)}\mathbf{q}_3$, recall that the LOS vectors in $\mathbf{f}, \mathbf{g}, \mathbf{u}, \mathbf{w}$ are expressed in the LLLN system at t_2. Thus, for example, for some LOS vector from the first view

$$\mathbf{q}_1 = C_{L_2}^{C_1}\mathbf{q}_1^{C_1} = C_{L_2}^{L_1}C_{L_1}^{B_1}C_{B_1}^{C_1}\mathbf{q}_1^{C_1} = C_{L_2}^{L_1}\left[I + [\Delta\mathbf{\Psi}_1]_\times\right]C_{L_{1,True}}^{B_1}C_{B_1}^{C_1}\mathbf{q}_1^{C_1} \tag{B.73}$$

$$\approx \bar{\mathbf{q}}_1 - C_{L_2}^{L_1}[\mathbf{q}_1^{L_1}]_\times\Delta\mathbf{\Psi}_1 \tag{B.74}$$

here $\bar{\mathbf{q}}$ is the true value of $\mathbf{q}$. In a similar manner we get:

$$\mathbf{q}_2 \approx \bar{\mathbf{q}}_2 - [\mathbf{q}_2^{L_2}]_\times\Delta\mathbf{\Psi}_2 \tag{B.75}$$

$$\mathbf{q}_3 \approx \bar{\mathbf{q}}_3 - C_{L_2}^{L_3}[\mathbf{q}_3^{L_3}]_\times\Delta\mathbf{\Psi}_3 \tag{B.76}$$

Consequently,

$$\nabla_{\mathbf{\Psi}(t_1)}\mathbf{q}_1 = -C_{L_2}^{L_1}[\mathbf{q}_1^{L_1}]_\times \tag{B.77}$$
$$\nabla_{\mathbf{\Psi}(t_2)}\mathbf{q}_2 = -[\mathbf{q}_2^{L_2}]_\times \tag{B.78}$$
$$\nabla_{\mathbf{\Psi}(t_3)}\mathbf{q}_3 = -C_{L_2}^{L_3}[\mathbf{q}_3^{L_3}]_\times \tag{B.79}$$

Incorporating all the above expressions, Eq. (B.67) turns into

$$\nabla_{\mathbf{\Psi}(t_3)}\left[\mathcal{A}\mathbf{T}_{23}\right] = -\sum_{i=1}^{N_{123}} \mathbf{e}_i \mathbf{T}_{23}^T \frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{3_i}} C_{L_2}^{L_3}[\mathbf{q}_{3_i}^{L_3}]_\times - \sum_{i=1}^{N_{23}} \mathbf{e}_{N_{123}+i} \mathbf{T}_{23}^T \frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{3_i}} C_{L_2}^{L_3}[\mathbf{q}_{3_i}^{L_3}]_\times \tag{B.80}$$

Noting that $\mathbf{g}$ is not a function of $\mathbf{q}_3$ and following a similar procedure we get

$$\nabla_{\mathbf{\Psi}(t_3)}\left[\mathcal{B}\mathbf{T}_{12}\right] = -\sum_{i=1}^{N_{123}} \mathbf{e}_i \mathbf{T}_{12}^T \frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{3_i}} C_{L_2}^{L_3}[\mathbf{q}_{3_i}^{L_3}]_\times \tag{B.81}$$

In conclusion, $H^{\mathbf{\Psi}(t_3)}$ may be calculated according to

$$\begin{aligned} H^{\mathbf{\Psi}(t_3)}|_{N\times 3} &= \sum_{i=1}^{N_{123}} \mathbf{e}_i \left[\mathbf{T}_{12}^T \frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{3_i}} - \mathbf{T}_{23}^T \frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{3_i}}\right] C_{L_2}^{L_3}[\mathbf{q}_{3_i}^{L_3}]_\times - \\ &- \sum_{i=1}^{N_{23}} \mathbf{e}_{N_{123}+i} \mathbf{T}_{23}^T \frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{3_i}} C_{L_2}^{L_3}[\mathbf{q}_{3_i}^{L_3}]_\times \end{aligned} \tag{B.82}$$

Applying the same technique, the matrices $H^{\mathbf{\Psi}(t_2)}$ and $H^{\mathbf{\Psi}(t_1)}$ were obtained as

$$\begin{aligned} H^{\mathbf{\Psi}(t_2)} &= \sum_{i=1}^{N_{123}} \mathbf{e}_i \left[\mathbf{T}_{12}^T \frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{2_i}} - \mathbf{T}_{23}^T \frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{2_i}}\right] [\mathbf{q}_{2_i}^{L_2}]_\times - \\ &- \sum_{i=1}^{N_{23}} \mathbf{e}_{N_{123}+i} \mathbf{T}_{23}^T \frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{2_i}} [\mathbf{q}_{2_i}^{L_2}]_\times + \sum_{i=1}^{N_{12}} \mathbf{e}_{N_{123}+N_{23}+i} \mathbf{T}_{12}^T \frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{2_i}} [\mathbf{q}_{2_i}^{L_2}]_\times \end{aligned} \tag{B.83}$$

$$\begin{aligned} H^{\mathbf{\Psi}(t_1)} &= \sum_{i=1}^{N_{123}} \mathbf{e}_i \left[\mathbf{T}_{12}^T \frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{1_i}} - \mathbf{T}_{23}^T \frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{1_i}}\right] C_{L_2}^{L_1}[\mathbf{q}_{1_i}^{L_1}]_\times + \\ &+ \sum_{i=1}^{N_{12}} \mathbf{e}_{N_{123}+N_{23}+i} \mathbf{T}_{12}^T \frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{1_i}} C_{L_2}^{L_1}[\mathbf{q}_{1_i}^{L_1}]_\times \end{aligned} \tag{B.84}$$

B.2.2 Calculation of the Matrices D and R

The matrices D and R are given by:

$$\mathcal{D} \doteq \nabla_{\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}} \mathbf{h} \tag{B.85}$$
$$R \doteq \operatorname{cov}\left(\left\{\mathbf{q}_{1_i}^{C_1}, \mathbf{q}_{2_i}^{C_2}, \mathbf{q}_{3_i}^{C_3}\right\}\right) \tag{B.86}$$

D reflects the influence of image noise on the measurement $\mathbf{z}$, while R is the image noise covariance for each matching LOS vector in the given images. Assuming that the camera optical axis lies along the z direction, a general LOS vector is contaminated by image noise $\mathbf{v} = (v_x, v_y)^T$, according to

$$\mathbf{q}^C = \bar{\mathbf{q}}^C + \begin{pmatrix} v_x & v_y & 0 \end{pmatrix}^T \tag{B.87}$$

where $\bar{\mathbf{q}}^C$ is the true value of the LOS vector, without noise contamination. Note that thus far we have omitted the explicit notation in the LOS vectors, thereby assuming that all the vectors are given in the LLLN system of t_2.

Recall that the sets of matching triplets and matching pairs

$$\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}} \quad , \quad \{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}} \quad , \quad \{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}} \tag{B.88}$$

were assumed to be consistent (cf. Section 3.2.1). Thus, for example, the matrices U and F, which are part of the matrix $\mathcal{A}$, are comprised of

$$U = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_{N_{123}} \end{bmatrix}^T \quad F = \begin{bmatrix} \mathbf{f}_1 & \dots & \mathbf{f}_{N_{23}} \end{bmatrix}^T \tag{B.89}$$

with $\mathbf{u}_i$ and $\mathbf{f}_i$, constructed using the same LOS vectors, for $i \leq N_{123}$. We define ΔN_{12} and ΔN_{23} as the number of additional pairs in $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}\}_{i=1}^{N_{12}}$ and $\{\mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{23}}$ that are not present in $\{\mathbf{q}_{1_i}, \mathbf{q}_{2_i}, \mathbf{q}_{3_i}\}_{i=1}^{N_{123}}$: $N_{12} = N_{123} + \Delta N_{12}$ and $N_{23} = N_{123} + \Delta N_{23}$. Although the overall number of matches in the above sets (Eq. (B.88)) is $N = N_{123} + N_{12} + N_{23}$, the actual number of *different* matches is $N_{123} + \Delta N_{12} + \Delta N_{23}$.

Assuming that the covariance of the image noise is the same for all the LOS vectors in the three images, and recalling the structure of the matrices $\mathcal{A}, \mathcal{B}$ that are used for calculating $\mathbf{h}$, we can write

$$\begin{aligned} DRD^T \;=\; & \sum_{i=1}^{N_{123}+\Delta N_{12}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{1_i}^{C_1}} + \sum_{i=1}^{N_{123}+\Delta N_{12}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{2_i}^{C_2}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{2_i}^{C_2}} + \\ & + \sum_{i=1}^{N_{123}+\Delta N_{23}} \frac{\partial \mathbf{h}}{\partial \mathbf{q}_{3_i}^{C_3}} R_{\mathbf{v}} \frac{\partial \mathbf{h}^T}{\partial \mathbf{q}_{3_i}^{C_3}} \end{aligned} \tag{B.90}$$

In the above equation, each summation refers to all the LOS vectors from the relevant image that participate in the calculation of $\mathbf{h}$. For example, the first summation refers to the first image. $R_{\mathbf{v}}$ is a 3×3 covariance matrix of the image noise

$$R_{\mathbf{v}} = \begin{bmatrix} R_x & 0 & 0 \\ 0 & R_y & 0 \\ 0 & 0 & R_f \end{bmatrix} \tag{B.91}$$

with $R_x = E(v_x v_x^T)$ and $R_y = E(v_y v_y^T)$. R_f represents the uncertainty in the camera focal length. Assuming the focal length is known precisely, it can be chosen as zero.

Next we develop expressions for $\frac{\partial \mathbf{h}}{\partial \mathbf{q}_k^{C_k}}$ for each image (i. e. $k = 1, 2, 3$). We begin with $\frac{\partial \mathbf{h}}{\partial \mathbf{q}_1^{C_1}}$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}}|_{N\times 3} = \frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{q}_{1_i}^{C_1}} - \frac{\partial \mathcal{B}\mathbf{T}_{12}}{\partial \mathbf{q}_{1_i}^{C_1}} \tag{B.92}$$

Since the matrices U, W and G contain LOS vectors from the first view while the matrix F does not, the above equals to

$$\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}} = \sum_{k=1}^{N_{123}} \frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{u}_k}\frac{\partial \mathbf{u}_k}{\partial \mathbf{q}_{1_i}^{L_2}}\frac{\partial \mathbf{q}_{1_i}^{L_2}}{\partial \mathbf{q}_{1_i}^{C_1}} - \sum_{k=1}^{N_{123}} \frac{\partial \mathcal{B}\mathbf{T}_{12}}{\partial \mathbf{w}_k}\frac{\partial \mathbf{w}_k}{\partial \mathbf{q}_{1_i}^{L_2}}\frac{\partial \mathbf{q}_{1_i}^{L_2}}{\partial \mathbf{q}_{1_i}^{C_1}} - \sum_{k=1}^{N_{12}} \frac{\partial \mathcal{B}\mathbf{T}_{12}}{\partial \mathbf{g}_k}\frac{\partial \mathbf{g}_k}{\partial \mathbf{q}_{1_i}^{L_2}}\frac{\partial \mathbf{q}_{1_i}^{L_2}}{\partial \mathbf{q}_{1_i}^{C_1}} \tag{B.93}$$

Noting that $\forall i \neq k$, $\frac{\partial \mathbf{u}_k}{\partial \mathbf{q}_{1_i}^{L_2}} = \frac{\partial \mathbf{w}_k}{\partial \mathbf{q}_{1_i}^{L_2}} = \frac{\partial \mathbf{g}_k}{\partial \mathbf{q}_{1_i}^{L_2}} = 0$, and taking into account that $\frac{\partial \mathbf{q}_{1_i}^{L_2}}{\partial \mathbf{q}_{1_i}^{C_1}} = C_{L_2}^{C_1}$, the above turns into

$$\begin{aligned}
\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{1_i}^{C_1}}|_{N\times 3} &= \left[\frac{\partial \mathcal{A}\mathbf{T}_{23}}{\partial \mathbf{u}_i}\frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{1_i}^{L_2}} - \frac{\partial \mathcal{B}\mathbf{T}_{12}}{\partial \mathbf{w}_i}\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{1_i}^{L_2}} - \frac{\partial \mathcal{B}\mathbf{T}_{12}}{\partial \mathbf{g}_i}\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{1_i}^{L_2}}\right]\frac{\partial \mathbf{q}_{1_i}^{L_2}}{\partial \mathbf{q}_{1_i}^{C_1}} = \\
&= \begin{cases} \left\{\mathbf{e}_i\left[\mathbf{T}_{23}^T\frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{1_i}^{L_2}} - \mathbf{T}_{12}^T\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{1_i}^{L_2}}\right] - \mathbf{e}_{N_{123}+N_{23}+i}\mathbf{T}_{12}^T\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{1_i}^{L_2}}\right\}C_{L_2}^{C_1} & i \leq N_{123} \\ -\mathbf{e}_{N_{123}+N_{23}+i}\mathbf{T}_{12}^T\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{1_i}^{L_2}}C_{L_2}^{C_1} & N_{123} < i \leq N_{123}+\Delta N_{12} \end{cases}
\end{aligned}$$

where the derivatives $\frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{1_i}^{L_2}}$, $\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{1_i}^{L_2}}$ and $\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{1_i}^{L_2}}$ were already computed (cf. Eq. (B.72)).

Using the same procedure we get the following expressions for the $N \times 3$ matrices $\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{2_i}^{C_2}}$ and $\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{3_i}^{C_3}}$:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{2_i}^{C_2}} = \begin{cases} \mathbf{e}_i\left[\mathbf{T}_{23}^T\frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{2_i}^{L_2}} - \mathbf{T}_{12}^T\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{2_i}^{L_2}}\right]C_{L_2}^{C_2} + \\ + \left[\mathbf{e}_{N_{123}+i}\mathbf{T}_{23}^T\frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{2_i}^{L_2}} - \mathbf{e}_{N_{123}+N_{23}+i}\mathbf{T}_{12}^T\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{2_i}^{L_2}}\right]C_{L_2}^{C_2} & i \leq N_{123} \\ \mathbf{e}_{N_{123}+i}\mathbf{T}_{23}^T\frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{2_i}^{L_2}}C_{L_2}^{C_2} & N_{123} < i \leq N_{123}+\Delta N_{23} \\ -\mathbf{e}_{2N_{123}+i}\mathbf{T}_{12}^T\frac{\partial \mathbf{g}_i}{\partial \mathbf{q}_{2_i}^{L_2}}C_{L_2}^{C_2} & N_{123}+\Delta N_{23} < i \leq N_{123}+\Delta N_{23}+\Delta N_{12} \end{cases}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{q}_{3_i}^{C_3}} = \begin{cases} \left\{\mathbf{e}_i\left[\mathbf{T}_{23}^T\frac{\partial \mathbf{u}_i}{\partial \mathbf{q}_{3_i}^{L_2}} - \mathbf{T}_{12}^T\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_{3_i}^{L_2}}\right] + \mathbf{e}_{N_{123}+i}\mathbf{T}_{23}^T\frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{3_i}^{L_2}}\right\}C_{L_2}^{C_3} & i \leq N_{123} \\ \mathbf{e}_{N_{123}+i}\mathbf{T}_{23}^T\frac{\partial \mathbf{f}_i}{\partial \mathbf{q}_{3_i}^{L_2}}C_{L_2}^{C_3} & N_{123} < i \leq N_{123}+\Delta N_{23} \end{cases}$$

Appendix C

Chapter 4 Extras

C.1 Computational Complexity Analysis

As seen in Section 4.2, the computational complexity changes from one scenario to another. Therefore, the analysis of the computational complexity is given here in terms of worst case.

Assume that $n-1$ MP update events have been carried out and currently the nth update event should be performed. Since each MP measurement is represented in the graph G by $r+1$ nodes, prior to the nth update event the graph G will contain $(r+1)(n-1) = (r+1)n-r-1$ nodes. These nodes are scattered among the platform threads in G. Since each node in the two trees may have one or r parents, the number of nodes in the ith level is bounded by r^{i-1}.

A tighter bound can be obtained by noting that at least one level should separate between two update-event nodes. Therefore, if a node has r parents, each of these parents nodes will have only one parent. Consequently, the number of nodes in the ith level is bounded by $r^{\lfloor 0.5(i-1) \rfloor}$.

The analyzed worst case is comprised of the following assumptions: (i) The number of nodes in each level i in the two trees is r^{i-1}; (ii) known pairs of nodes, in the sense of Definition 4.3.1, are found only upon reaching the top level in both trees, thereby ensuring maximum-size permutation sets $\mathcal{M}_k$ and that all the levels are processed by Algorithm 2; (iii) the computational cost of checking whether $\odot(a,b)$, i. e. whether $E[\tilde{\mathbf{X}}_a \tilde{\mathbf{X}}_b^T]$ is known, is $O(1)$.

Following these assumptions, the height h of each of the two trees can be calculated from

$$(r+1)n - r - 1 = \sum_{i=1}^{h} r^{i-1} = r^h - 1 \tag{C.1}$$

which implies

$$h \approx log_r(rn+n) \tag{C.2}$$

In addition,

$$\mathrm{card}(\mathcal{M}_k) \leq r^{2(k-1)} \tag{C.3}$$

The complexity of processing a single member from $\mathcal{M}_k$ is bounded by $2i$. Thus, without taking into account the involved complexity of Algorithm 4 for calculating the contribution of noise terms, the overall computational complexity is bounded by

$$\sum_{i=1}^{h} r^{2(i-1)} \cdot 2i = r^{-2} \sum_{i=1}^{h} r^{2i} \cdot 2i \tag{C.4}$$

Letting $j \doteq 2i$,

$$r^{-2} \sum_{i=1}^{h} r^{2i} \cdot 2i = r^{-2} \sum_{j=1}^{2h} j r^{j} - r^{-1} \tag{C.5}$$

Now, using the relation

$$\sum_{i=1}^{m} i r^{i} = \frac{r}{(r-1)^2} \left(1 - r^m - m r^m + m r^{1+m}\right) \approx \frac{1}{r} \left[m r^m (r-1) - (r^m - 1)\right] \tag{C.6}$$

and recalling that $h = log_r(rn+n)$ gives

$$\begin{aligned} r^{-2} \sum_{j=1}^{2h} j r^{j} - r^{-1} &< r^{-2} \sum_{j=1}^{2h} j r^{j} \\ &= r^{-3} \left[log_r(rn+n)^2 \cdot (rn+n)^2 (r-1) - ((rn+n)^2 - 1)\right] \\ &\approx r^{-3}(r+1)^2(r-1) n^2 log_r(rn+n)^2 \sim O\left(n^2 log(rn)\right) \end{aligned} \tag{C.7}$$

The computational complexity cost of calculating the contribution of the noise terms to the cross-covariance (Algorithm 4) can be bounded as follows. It is assumed that Algorithm 4 is carried out each time a pair from $\mathcal{M}_k$ is processed. Note that in practice, Algorithm 4 should be executed only upon finding a known pair of nodes. A single execution of this algorithm for a pair of nodes (c_i, d_i) from the ith level requires checking for each $a \in \mathcal{D}_c(c_i)$ whether $a \in \mathcal{A}_d(d_i)$, and for each $b \in \mathcal{D}_d(d_i)$ whether $b \in \mathcal{D}_c(c_i)$. This procedure is therefore bounded by $2ir^{h-i}$. Thus, processing a single member from $\mathcal{M}_i$ is now bounded by $2i + 2ir^{h-i}$ instead of $2i$. The overall computational complexity, including the complexity of Algorithm 4, is therefore bounded by

$$\sum_{i=1}^{h} r^{2(i-1)} \cdot (2i + 2ir^{h-i}) \sim O\left(n^2 log(rn)\right) \tag{C.8}$$

In conclusion, the worst-case complexity of calculating a cross-covariance term in a general scenario is bounded by $O\left(n^2 log(rn)\right)$.

C.2 Efficient Implementation

The computational load can be significantly reduced by efficient implementation methods. One possible implementation is described next.

A meta-structure $\mathcal{H}$ is created and maintained when constructing the two trees T_c and T_d. This structure is comprised of a header containing the details of all the nodes participating in either of the two trees. Each cell in the header, representing some node b, has also a flag indicating whether b appears in both of the trees. In addition, each cell points towards a structure that contains the following fields: The name of the tree in which b appears; height of the node b; link to the location of b in the tree. The structure contains also pointers to nodes $u^1, \dots u^{r-1}$, if such nodes exist, such that b and the nodes $u^1, \dots u^{r-1}$ belong to the same MP measurement update (and therefore, $E[\tilde{\mathbf{X}}_b \tilde{\mathbf{X}}_{u^i}^T]$, $i = 1, \dots, r-1$, are known). If b appears in the trees several times, a linked list is used in which each cell is a structure representing a single appearance of b in the trees. Figure C.1 shows schematically such a structure for $r = 3$.

This implementation allows processing each member $(c_k, d_k) \in \mathcal{M}_k$ more efficiently, although the worst-case computational complexity does not change. Instead of looking for $\odot(c_k, d_j)$ or $\odot(c_j, d_k)$, by going over the nodes in $c_k \overset{T_c}{\Longrightarrow} c$ and $d_k \overset{T_d}{\Longrightarrow} d$, the following may be performed: Check in the meta-structure $\mathcal{H}$ whether c_k is linked to any other nodes, which were part of the same MP update. For each such node u (there are only $r-1$ such nodes), check if $u \in V_{T_d}$ by going over the linked list of u in $\mathcal{H}$. For each appearance $u \in V_{T_d}$, check if $h_d(u) < k$, and then check if $d_k \in \mathcal{A}_d(u)$. Choose the node u with the smallest height. Repeat the process for d_k with the proper adjustments.

Assume that $\odot(c_j, d_k)$. When computing the contribution of the noise terms (cf. Section 4.3.2.3), instead of processing all the nodes[1] in $(T_d)^{d_k}$ and $(T_c)^{c_j}$, checking whether they appear in $c_j \overset{T_c}{\Longrightarrow} c$ and $d_k \overset{T_d}{\Longrightarrow} d$, respectively, the following may be performed. For each node $c_r \in c_j \overset{T_c}{\Longrightarrow} c$, check in $\mathcal{H}$ whether it appears in T_d (indicated by flag $= 1$). If it does, go over the linked list of c_r in $\mathcal{H}$ and choose only the appearances of c_r in T_d which are higher than k. For each chosen appearance of c_r, verify that d_k is a descendant. Repeat the process for $d_k \overset{T_d}{\Longrightarrow} d$ (with respect to T_c).

C.3 Efficient Calculation of Transition and Process Noise Covariance Matrices

The problem this section refers to is of calculating transition and process noise covariance matrices between some two time instances which are unknown a priori. These matrices participate in calculation of the cross-covariance terms (cf. Section 4.2).

[1] The number of nodes in $(T_a)^b$ is bounded by r^{h-h_b}, where h is the height of the tree T_a, and h_b is the height of node b in T_a.

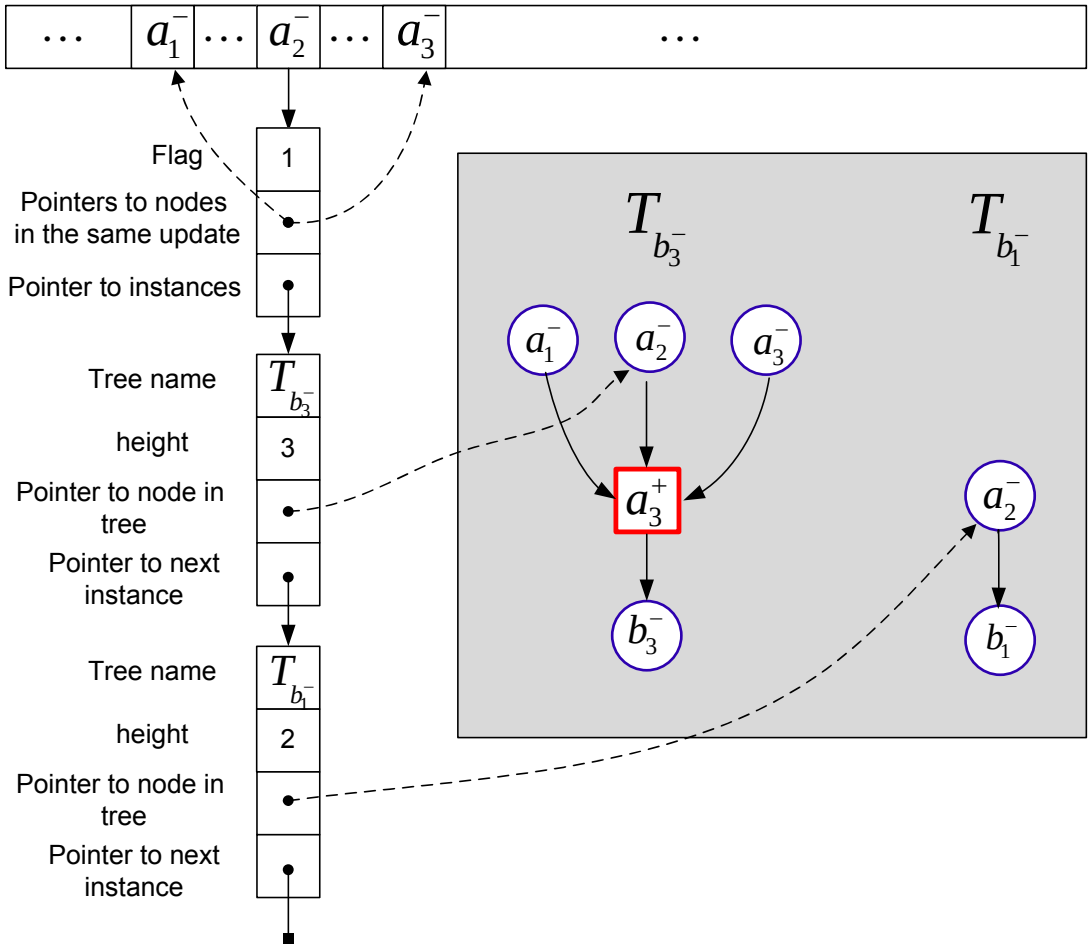


Figure C.1: Schematic illustration of a possible implementation of $\mathcal{H}$ for the scenario shown in Figure 4.2. Only the structure for the node a_2^- is shown. Note that flag's value is 1 since a_2^- appears in both trees.

In addition to the graph G, locally constructed by every platform in the group, each platform (or some of the platforms) is assumed to maintain a repository composed of external sensors readings and of navigation data at different time instances. Thus, the ith platform maintains a repository storing $\boldsymbol{\zeta}_i$ and $\mathbf{y}_i$ at different time instances. The platforms' repositories are used for calculating the MP measurement $\mathbf{z}$ (cf. Eq. (4.7)).

We first discuss calculation of transition matrices. Consider calculation of a transition matrix for the ith platform relating $\mathbf{X}_i(t_k)$ with $\mathbf{X}_i(t_{k+s})$, i. e., $\Phi^i_{t_k \to t_{k+s}}$. The parameter s is any positive number for which Eq. (4.5) correctly represents the relation between $\mathbf{X}_i(t_k)$ and $\mathbf{X}_i(t_{k+s})$. Assume that the information sets $(\boldsymbol{\zeta}_i(t_k), \mathbf{y}_i(t_k)), \ldots, (\boldsymbol{\zeta}_i(t_{k+s}), \mathbf{y}_i(t_{k+s}))$ are stored in the repository of the ith platform.

A naive approach for calculating $\Phi^i_{t_k \to t_{k+s}}$ would be based on

$$\Phi^i_{t_k \to t_{k+s}} = \Phi^i_{t_{k+s} \to t_{k+s-1}} \cdot \ldots \cdot \Phi^i_{t_k \to t_{k+1}} \tag{C.9}$$

However, a much more time-efficient alternative is to calculate $\Phi^i_{t_k \to t_{k+s}}$ using transition matrices bridging between several time instances. For example, if we had available the matrix $\Phi^i_{t_k \to t_{k+s-1}}$, the computation of $\Phi^i_{t_k \to t_{k+s}}$ would require multiplication of only two matrices:

$$\Phi^i_{t_k \to t_{k+s}} = \Phi^i_{t_{k+s-1} \to t_{k+s}} \cdot \Phi^i_{t_k \to t_{k+s-1}} \tag{C.10}$$

This concept can be obtained by maintaining a skip list [87] type database. The lowest level is comprised of the stored images and its associated navigation data, including the transition matrices between adjacent stored images. This level is a possible implementation of the repository maintained by all/some platforms. Each next level is constructed by skipping several nodes in the lower level, and assigning the appropriate transition matrix, transferring from previous node to next node in the same level. No other data is stored outside the first level nodes.

An example of this concept is given in Figure C.2, in which every two nodes in some level contribute a node in the next level. A simplified notation is used in this figure, where $\Phi^i_{t_k \to t_{k+s}}$ is represented by $\Phi_{k \to k+s}$ (the platform identity notation is omitted). Thus, for instance, calculation of $\Phi_{2 \to 5}$ can be performed by searching for the appropriate route in the skip list formation, which will yield $\Phi_{2 \to 5} = \Phi_{3 \to 5}\Phi_{2 \to 3}$, instead of carrying out the three matrix multiplications $\Phi_{2 \to 5} = \Phi_{4 \to 5}\Phi_{3 \to 4}\Phi_{2 \to 3}$.

The process noise covariance matrix between any two time instances can be efficiently calculated following a similar approach. For example, $Q^i_{t_k:t_{k+s}}$ is given by

$$\begin{aligned} Q^i_{t_k:t_{k+s}} &= Q^i_{t_{k+s-1}:t_{k+s}} + \\ &+ \Phi^i_{t_{k+s-1} \to t_{k+s}} Q^i_{t_{k+s-2}:t_{k+s-1}} (\Phi^i_{t_{k+s-1} \to t_{k+s}})^T + \cdots + \Phi^i_{t_{k+1} \to t_{k+s}} Q^i_{t_k:t_{k+1}} (\Phi^i_{t_{k+1} \to t_{k+s}})^T \end{aligned} \tag{C.11}$$

However, if each node in the skip list database contains the noise covariance matrix between the previous node in the same level, $Q^i_{t_k:t_{k+s}}$ can be also calculated, for instance, as

$$Q^i_{t_k:t_{k+s}} = Q^i_{t_{k+s-1}:t_{k+s}} + \Phi^i_{t_{k+s-1} \to t_{k+s}} Q^i_{t_k:t_{k+s-1}} (\Phi^i_{t_{k+s-1} \to t_{k+s}})^T \tag{C.12}$$

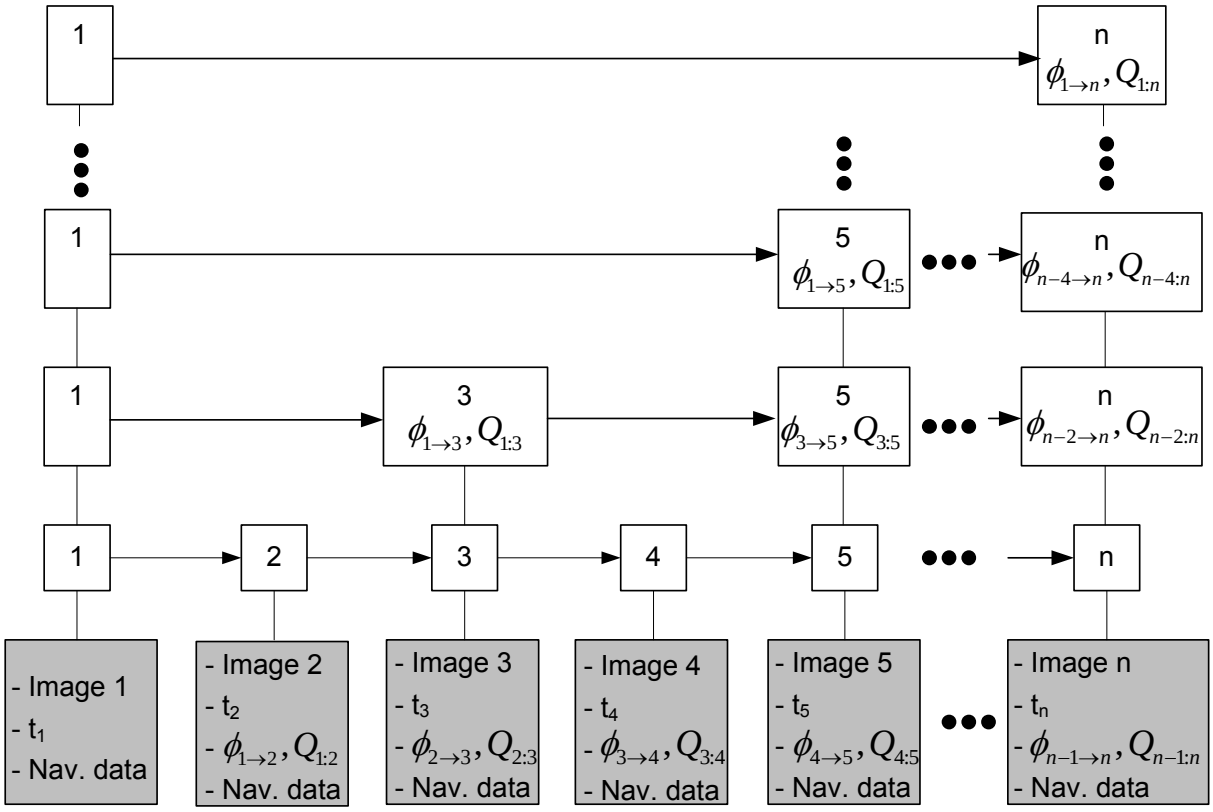


Figure C.2: Skip list repository database example.

Appendix D

List of Publications

This appendix lists the journal and conference papers published based on this PhD research.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Distributed Vision-Aided Cooperative Localization and Navigation Based on Three-View Geometry", submitted to *Journal of Field Robotics*.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Graph-Based Distributed Cooperative Navigation for a General Multi-Robot Measurement Model", submitted to *International Journal of Robotics Research*.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Real-Time Vision-Aided Localization and Navigation Based on Three-View Geometry", *IEEE Transactions on Aerospace and Electronic Systems*, accepted.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Graph-based Distributed Cooperative Navigation", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Distributed Vision-Aided Cooperative Localization and Navigation based on Three-View Geometry", *Proceedings of the IEEE Aerospace Conference*, Montana, USA, March 2011.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Navigation Aiding Based on Coupled Online Mosaicking and Camera Scanning", *Journal of Guidance, Control and Dynamics*, Vol. 33, No. 6, 2010, pp. 1866-1882.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Mosaic Aided Navigation: Tools, Methods and Results", *IEEE/ION Position Location and Navigation System (PLANS) Conference*, California, USA, May 2010.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Real-Time Mosaic-Aided Aerial Navigation: I. Motion Estimation", *AIAA Guidance, Navigation and Control Conference*, Chicago, USA, Aug. 2009.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Real-Time Mosaic-Aided Aerial Navigation: II. Sensor Fusion", *AIAA Guidance, Navigation and Control Conference*, Chicago, USA, Aug. 2009.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Navigation Aiding Using On-Line Mosaicking", *IEEE/ION Position Location and Navigation System (PLANS) Conference*, California, USA, May 2008.

- V. Indelman, P. Gurfil, E. Rivlin and H. Rotstein, "Navigation Performance Enhancement Using Rotation and Translation Measurements from Online Mosaicking", *AIAA Guidance, Navigation and Control Conference*, Hilton Head, SC, USA, Aug. 2007.

Bibliography

[1] Titterton, D. H. and Weston, J. L., *Strapdown Inertial Navigation Technology*, AIAA, 2004.

[2] Merhav, S. and Bresler, Y., "On-Line Vehicle Motion Estimation from Visual Terrain Information Part 1: Recursive Image Registration," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 22, No. 5, 1986, pp. 583–587.

[3] Soatto, S. and Perona, P., "Recursive 3-D Visual Motion Estimation Using Subspace Constraints," *International Journal of Computer Vision*, Vol. 22, No. 3, 1997, pp. 235–259.

[4] Gurfil, P. and Rotstein, H., "Partial Aircraft State Estimation from Visual Motion Using the Subspace Constraints Approach," *Journal of Guidance, Control and Dynamics*, Vol. 24, No. 5, July 2001, pp. 1016–1028.

[5] Diel, D., DeBitetto, P., and Teller, S., "Epipolar Constraints for Vision-Aided Inertial Navigation," *Proceedings of the IEEE Workshop on Motion and Video Computing*, Vol. 2, 2005, pp. 221–228.

[6] Roumeliotis, S., Johnson, A., and Montgomery, J., "Augmenting Inertial Navigation with Image-Based Motion Estimation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4, 2002, pp. 4326–4333.

[7] Merhav, S. and Bresler, Y., "On-Line Vehicle Motion Estimation from Visual Terrain Information Part 2: Ground Velocity and Position Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 22, No. 5, 1986, pp. 588–604.

[8] Lerner, R., Rivlin, E., and Rotstein, H., "Pose and Motion Recovery from Feature Correspondences and a Digital Terrain Map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 9, September 2006, pp. 1404–1417.

[9] Sim, D., Park, R., Kim, R., Lee, S., and Kim, I., "Integrated Position Estimation Using Aerial Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, January 2002, pp. 1–18.

[10] Gracias, N., Zwaan, S., Bernardino, A., and Santos-Victor, J., "Results on Underwater Mosaic-based Navigation," *IEEE OCEANS*, Vol. 3, 2002, pp. 1588–1594.

[11] Gracias, N. and Santos-Victor, J., "Underwater Video Mosaics as Visual Navigation Maps," *Computer Vision and Image Understanding*, Vol. 79, 2000, pp. 66–91.

[12] Eustice, R. M., Pizarro, O., and Singth, H., "Visually Augmented Navigation for Autonomous Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, Vol. 33, No. 2, 2008, pp. 103–122.

[13] Hartley, R. and Zisserman, A., *Multiple View Geometry*, Cambridge University Press, 2000.

[14] Tsai, R., Huang, T., and Zhu, W., "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, II: Singular Value Decomposition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 30, No. 4, 1982, pp. 525–534.

[15] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Improving Vision-based Planar Motion Estimation for Unmanned Aerial Vehicles through Online Mosaicing," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006, pp. 2860–2865.

[16] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Vision-Based Odometry and SLAM for Medium and High Altitude UAVs," *Journal of Intelligent and Robotic Systems*, Vol. 54, No. 1-3, March 2009, pp. 137–161.

[17] Mourikis, A. and Roumeliotis, I., "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 3565–3572.

[18] Mourikis, A. and Roumeliotis, I., "A Dual-Layer Estimator Architecture for Long-term Localization," *Proceedings of the IIEEE Computer Vision and Pattern Recognition Workshops*, June 2008, pp. 1–8.

[19] Shakernia, O., Vidal, R., Sharp, C., Ma, Y., and Sastry, S., "Multiple View Motion Estimation and Control for Landing an Unmanned Aerial Vehicle," *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, D.C., USA, May 2002, pp. 2793–2798.

[20] Ma, Y., Huang, K., Vidal, R., Kosecka, J., and Sastry, S., "Rank Conditions on the Multiple-View Matrix," *International Journal of Computer Vision*, Vol. 59, No. 2, May 2004, pp. 115–137.

[21] Yu, Y. K., Wong, K. H., Chang, M. M. Y., and Or, S. H., "Recursive Camera-Motion Estimation With the Trifocal Tensor," *IEEE Transactions on Systems, Man, And Cybernetics - Part B: Cybernetics*, Vol. 36, No. 5, October 2006, pp. 1081–1090.

[22] Davison, A. J., Reid, I. D., and Molton, N. D., "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, 2007.

[23] Liu, Y. and Thrun, S., "Results for outdoor-SLAM using sparse extended information filters," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003, pp. 1227–1233.

[24] Kim, J. and Sukkarieh, S., "6DoF SLAM aided GNSS/INS Navigation in GNSS Denied and Unknown Environments," *Journal of Global Positioning Systems*, Vol. 4, No. 1-2, 2005, pp. 120–128.

[25] Garcia, R., *A proposal to estimate the motion of an underwater vehicle through visual mosaicking*, Phd thesis. University of Girona, Spain, 2002.

[26] Durrant-Whyte, H. F. and Bailey, T., "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics and Automation Magazine*, Vol. 2, 2006.

[27] Garcia, R., Puig, J., Ridao, P., and Cufi, X., "Augmented State Kalman Filtering for AUV Navigation," *IEEE Proceedings on International Conference Robotics and Automation*, Vol. 4, 2002, pp. 4010– 4015.

[28] Leonard, J. J. and Durrant-Whyte, H. F., "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Osaka, Japan, November 1991.

[29] Bryson, M. and Sukkarieh, S., "Active Airborne Localization and Exploration in Unknown Environments using Inertial SLAM," *Proceedings of the IEEE Aerospace Conference*, Montana, USA, 2006.

[30] Bryson, M. and Sukkarieh, S., "Bearing-Only SLAM for an Airborne Vehicle," Sydney, Australia, 2005.

[31] Leonard, J. J., Rikoski, R. J., Newman, P. M., and Bosse, M., "Mapping partially observable features from multiple uncertain vantage points," *International Journal of Robotics Research*, Vol. 21, 2002, pp. 943–975.

[32] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H., "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research*, Vol. 23, No. 7-8, 2004, pp. 693–716.

[33] George, M. and Sukkarieh, S., "Inertial Navigation Aided by Monocular Camera Observations of Unknown Features," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007.

[34] Leonard, J. J., Jacob, H., and Feder, S., "A computationally efficient method for large-scale concurrent mapping and localization," *Proceedings of the Ninth International Symposium on Robotics Research*, Springer-Verlag, 1999, pp. 169–176.

[35] Fleischer, S., Marks, R., Rock, S., and Lee, M., "Improved Real-Time Video Mosaicking of the Ocean Floor," *Proceedings of the Oceans Conference*, Vol. 3, San Diego, California, USA, October 1995, pp. 1935–1944.

[36] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Homography Based Kalman Filter for Mosaic Building. Applications to UAV Position Estimation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 2004–2009.

[37] Szeliski, R., *Image alignment and stitching: A tutorial*, Tech. Rep. MSR-TR-2004-92, Microsoft Research, 2005.

[38] Negahdaripour, S. and Xu, X., "Mosaic-Based Positioning and Improved Motion-Estimation Methods for Automatic Navigation of Submersible Vehicles," *IEEE Journal of Oceanic Engineering*, Vol. 27, No. 1, 2002, pp. 79–99.

[39] Gracias, N., Zwaan, S., Bernardino, A., and Santos-Victor, J., "Moasic Based Navigation for Autonomous Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, Vol. 28, No. 4, 2003, pp. 609–624.

[40] Lovegrove, S. and Davison, A. J., *Real-Time Spherical Mosaicing Using Whole Image Alignment*, Vol. 6313, Lecture Notes in Computer Science, ECCV 2010, Part III, Springer Berlin Heidelberg, 2010.

[41] Richmond, K. and Rock, S., "An Operational Real-Time Large-Scale Visual Mosaicking and Navigation System," *OCEANS*, Portugal, September 2006, pp. 1–6.

[42] Madhavan, R., Fregene, K., and Parker, L. E., "Distributed Cooperative Outdoor Multirobot Localization and Mapping," *Autonomous Robots*, Vol. 17, 2004, pp. 23–29.

[43] Nettletona, E., Thrun, S., Durrant-Whyte, H., and Sukkarieh, S., "Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles," *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.

[44] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S., "Multiple Relative Pose Graphs for Robust Cooperative Mapping," *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[45] Lazaro, M. T. and Castellanos, J. A., "Localization of Probabilistic Robot Formations in SLAM," *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[46] Shaferman, V. and Shaferman, T., "Unmanned Aerial Vehicles Cooperative Tracking of Moving Ground Target in Urban Environments," *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 5, 2008, pp. 1360–1371.

[47] Smaili, C., Najjar, M. E. E., and Charpillet, F., "Multi-sensor Fusion Method Using Bayesian Network for Precise Multi-vehicle Localization," *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Beijing, China, 2008, pp. 906–911.

[48] Roumeliotis, S. I. and Bekey, G. A., "Distributed Multirobot Localization," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 781–795.

[49] Kurazume, R., Nagata, S., and Hirose, S., "Cooperative Positioning with Multiple Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 1250–1257.

[50] Fenwick, J. W., Newman, P. M., and Leonard, J. J., "Cooperative Concurrent Mapping and Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, USA, May 2002.

[51] Knuth, J. and Barooah, P., "Distributed collaborative localization of multiple vehicles from relative pose measurements," *Forty-Seventh Annual Allerton Conference*, Illinois, USA, 2009, pp. 314–321.

[52] Sharma, R. and Taylor, C. N., "Vision Based Distributed Cooperative Navigation for MAVs in GPS denied areas," *Proceedings of the AIAA Infotech@Aerospace Conference*, Washington, USA, April 2009.

[53] Huang, G. P., Trawny, N., Mourikis, A. I., and Roumeliotis, S. I., "Observability-based Consistent EKF Estimators for Multi-robot Cooperative Localization," *Auton. Robots*, Vol. 30, January 2011, pp. 99–122.

[54] Martinelli, A., Pont, F., and Siegwart, R., "Multi-Robot Localization Using Relative Observations," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Barcelona, Spain, 2005, pp. 2797–2802.

[55] Caglioti, V., Citterio, A., and Fossati, A., "Cooperative, Distributed Localization in Multi-robot Systems: a Minimum-entropy Approach," *Proceedings of the IEEE Workshop on Distributed Intelligent Systems*, 2006, pp. 25–30.

[56] Nerurkar, E. D., Roumeliotis, S. I., and Martinelli, A., "Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 1402–1409.

[57] Nerurkar, E. D. and Roumeliotis, S. I., "Multi-Centralized Cooperative Localization under Asynchronous Communication," *Department of Computer Science and Engineering, University of Minnesota, Technical Report*, March 2010.

[58] Howard, A., Mataric, M. J., and Sukhatme, G. S., "Putting the 'I' in 'Team' - an Ego-Centric Approach to Cooperative Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003, pp. 868–874.

[59] Karam, N., Chausse, F., Aufrere, R., and Chapuis, R., "Localization of a Group of Communicating Vehicles by State Exchange," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 519–524.

[60] Sharma, R. and Taylor, C., "Cooperative Navigation of MAVs In GPS Denied Areas," *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul, Korea, 2008, pp. 481–486.

[61] Merino, L., Wiklund, J., Caballero, F., Moe, A., Ramiro, J., Forssen, E., Nordberg, K., and Ollero, A., "Vision-Based Multi-UAV Position Estimation," *IEEE Robotics and Automation Magazine*, September 2006, pp. 53–62.

[62] Bahr, A., Walter, M. R., and Leonard, J. J., "Consistent Cooperative Localization," *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 3415–3422.

[63] Mourikis, A. I., Roumeliotis, S. I., and Burdick, J. W., "SC-KF Mobile Robot Localization: A Stochastic Cloning Kalman Filter for Processing Relative-State Measurements," *IEEE Transactions on Robotics*, Vol. 23, No. 4, 2007, pp. 717–730.

[64] Julier, S. J. and Uhlmann, J. K., "A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations," *Proceedings of the American Control Conference*, Albuquerque, New Mexico, June 1997, pp. 2369–2373.

[65] Xu, X. and Negahdaripour, S., "Application of extended covariance intersection principle for mosaic-based optical positioning and navigation of underwater vehicles," *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[66] Arambel, P. O., Rago, C., and Mehra, R. K., "Covariance Intersection Algorithm for Distributed Spacecraft State Estimation," *Proceedings of the American Control Conference*, Arlington, VA, USA, June 2001, pp. 4398–4403.

[67] Julier, S. J. and Uhlmann, J. K., "Using covariance intersection for SLAM," *Elsevier Robotics and Autonomous Systems*, Vol. 55, 2007, pp. 3–20.

[68] Lazarus, S. B., Tsourdos, A., Silson, P., White, B., and Zbikowski, R., "Unmanned aerial vehicle navigation and mapping," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 222, No. 4, 2008, pp. 531–548.

[69] Farrel, J. A. and Barth, M., *The Global Positioning System and Inertial Navigation*, McGraw-Hill, 1998.

[70] Bryson, M. and Sukkarieth, S., "Observability Analysis and Active Control for Airborne SLAM," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 44, No. 1, January 2008, pp. 261–280.

[71] Fleischer, S., Wang, H., and Rock, S., "Video Mosaicking Along Arbitrary Vehicle Paths," *Proceedings of the Symposium on Vehicle Technology*, 1996, pp. 293–299.

[72] Gracias, N., Costeira, J., and J.Santos-Victor, "Linear Global mosaic For Underwater Surveying," *Symposium on Intelligent Autonomous Vehicles*, Portugal, 2004.

[73] Shum, H. and Szeliski, R., "Systems and Experiment Paper: Construction of Panroamic Image Mosaics with Global and Local Alignment," *International Journal of Computer Vision*, Vol. 36, No. 2, 2000, pp. 101–130.

[74] Ferrer, J., Elibol, A., Delaunoy, O., Gracias, N., and Garcia, R., "Large-Area Photo-Mosaics Using Global Alignment and Navigation Data," *OCEANS*, Portugal, September 2007, pp. 1–9.

[75] Zhang, P., Milios, E. E., and Gu, J., "Graph-based Automatic Consistent Image Mosaicking," *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Shenyang, China, 2004, pp. 558–563.

[76] Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, November 2004, pp. 91–110.

[77] Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. Assoc. Comp. Mach.*, Vol. 24, 1981, pp. 381–395.

[78] Soatto, S., Frezza, R., and Perona, P., "Motion Estimation via Dynamic Vision," *IEEE Transactions on Automatic Control*, Vol. 41, No. 3, March 1996, pp. 393–413.

[79] Dissanayake, G., Sukkarieh, S., Nebot, E., and Durrant-Whyte, H., "The Aiding of a Low-Cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications," *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 5, October 2001, pp. 731–747.

[80] Goshen-Meskin, D. and Bar-Itzhack, I., "Observability Analysis of Piece-Wise Constant Systems - Part I: Theory," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 28, No. 4, 1992, pp. 1056–1067.

[81] Bryson, M. and Sukkarieh, S., "Observability Analysis and Active Control for Airborne SLAM," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 44, No. 1, January 2008, pp. 261–280.

[82] Bryson, A. E. J. and Ho, Y., *Applied Optimal Control: Optimization, Estimation and Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.

[83] Horn, R. and Johnson, C., *Matrix Analysis*, Cambridge University Press, 1992.

[84] Goshen-Meskin, D. and Bar-Itzhack, I., "Observability Analysis of Piece-Wise Constant Systems - Part II: Application to Inertial Navigation In-Flight," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 28, No. 4, 1992, pp. 1068–1075.

[85] Yu, Y. K., Wong, K. H., Or, S. H., and Chang, M. M. Y., "Robust 3-D Motion Tracking From Stereo Images: A Model-Less Method," *IEEE Transactions on Instrumentation and Measurement*, Vol. 57, No. 3, March 2008, pp. 622–630.

[86] Guerrero, J. J., Murillo, A. C., and Sagües, C., "Localization and Matching Using the Planar Trifocal Tensor with Bearing-Only Data," *IEEE Transactions on Robotics*, Vol. 24, No. 2, April 2008, pp. 494–501.

[87] Pugh, W., "Skip Lists: A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, Vol. 33, No. 6, 1990, pp. 668–676.

האלגוריתם שפותח מייצג את כל המדידות בין הפלטפורמות השונות שבוצעו עד כה בגרף, המתוחזק בצורה עצמאית על ידי כל אחת מהפלטפורמות בקבוצה. האלגוריתם מאפשר חישוב מפורש של איברי הקרוס-קווריאנס על סמך גרף זה בתרחישים כלליים המשלבים מודלים שונים של מדידות בין פלטפורמות שונות שבוצעו בזמנים כלשהם.

באלגוריתם הרביעי שפותח במחקר זה מוצע להשתמש במדידות המורכבות מאילוצים של גיאומטריית שלוש תמונות לניווט קואופרטיבי. אילוצים אלו פותחו כחלק מהאלגוריתם השני שהוזכר לעיל. כעת, שלושת התמונות בעלות אזור החפיפה המשותף, עליהן מושתתים אילוצים אלו יכולות להיות מצולמות על ידי פלטפורמות שונות ובזמנים שונים. במקרה וחלק מהתמונות צולמו בעבר, התמונות ונתוני ניווט המוצמדים אליהן, נשלפים מתוך מאגרי הנתונים המתוחזקים על ידי הפלטפורמות השונות בקבוצה. בניגוד למדידות מצב יחסי, המצלמה בגישה זו אינה מאולצת להיות מכוונת כלפי פלטפורמות אחרות. היות ונתוני הניווט מהפלטפורמות השונות המשתתפים באותה מדידה יכולים להיות תלויים סטטיסטית, איברי הקרוס-קווריאנס הנדרשים בשלב עדכון מערכת הניווט מחושבים על סמך האלגוריתם לחישוב איברי הקרוס-קווריאנס עבור מודל מדידה כללי בין מספר פלטפורמות (המתואר לעיל). אלגוריתם זה הותאם למודל המדידה המוכתב על ידי אילוצים של גיאומטריית שלוש תמונות. בדומה למקרה של פלטפורמה בודדת, אילוצים אלו מאפשרים להקטין שגיאות ניווט בפלטפורמה המעודכנת, כולל שגיאות מיקום ומהירות בכל הצירים, ללא שימוש במדידים נוספים (כגון מד טווח) וללא מידע נוסף כלשהו.

תמונות פסיפס) יכול להתבצע בתהליך רקע. האלגוריתם מאפשר לבצע עדכון מערכת הניווט בתרחישי מעגל, כמו בכל תרחיש אחר, תוך עיבוד של שלוש תמונות בלבד ובכך מציג דרישות חישוביות נמוכות ביחס לגישות מקובלות (כגון SLAM). למיטב ידיעת המחבר, שילוב אילוצים הנובעים מגיאומטריה של שלושה מבטים, כולל ה-trifocal tensor המוכר היטב בספרות, לא הוצע עד כה לצורך עדכון מערכת הניווט ובפרט לצורך טיפול בתרחישי מעגל.

חלקו השני של המחקר מתרכז בבעיית הניווט הקואופרטיבי. בהינתן קבוצת פלטפורמות המסוגלות לתקשר ביניהן, כל אחת מצוידת במערכת ניווט אינרציאלית, במצלמה ואולי במדידים נוספים, מפותחות שיטות לשיפור ביצועי הניווט של הפלטפורמות השונות על ידי שיתוף פעולה בין הפלטפורמות בקבוצה.

הגישה המקובלת בספרות לניווט קואופרטיבי הינה שימוש במדידות של מיקום ומצב זוויתי יחסי (להלן מצב יחסי) בין זוגות של פלטפורמות. מדידות אלו מניחות קיום של מד טווח (או לחילופין זוג מצלמות), אשר בלעדיו לא ניתן לעדכן מיקום בכל הצירים. גישה אחרת לניווט קואופרטיבי שהוצעה לאחרונה הינה לזהות סצנה הנצפית על ידי זוג פלטפורמות ולנצל את האילוצים הנובעים מגיאומטריה של שתי תמונות לצורך עדכון מערכת הניווט של הפלטפורמות. אולם, גם בגישה זו, בהיעדר מד טווח, המדידות אינן מאפשרות עדכון מיקום בכל הצירים. כפי שמתואר בהמשך, במחקר זה מוצעת גישה לניווט קואופרטיבי בה שגיאת המיקום מוקטנת בכל הצירים, ללא שימוש במד טווח.

סוגיה נוספת שיש לתת עליה את הדעת הינה התלות הסטטיסטית בין נתוני הניווט בפלטפורמות השונות בקבוצה. תלות זו מתפתחת לאחר ביצוע עדכוני ניווט המשלבים נתוני ניווט ממספר פלטפורמות. הזנחה של תלות סטטיסטית זאת עלולה להוביל לשערוך לא עקבי ואופטימי. במקרה של מדידות מצב יחסי בין זוגות של פלטפורמות, הגישה המקובלת לטפל בסוגיה זו הינה חישוב ותחזוקה של מטריצת קווריאנס מאוחדת, המכילה את הקווריאנס של כל אחת מהפלטפורמות ואיברי קרוס-קווריאנס בין כל זוגות הפלטפורמות בקבוצה. גישה זו אכן מתאימה עבור מדידות המושתתות על נתוני ניווט מהזמן הנוכחי של הפלטפורמות השונות (זוג פלטפורמות במקרה של מדידות מצב יחסי). אולם, במודל מדידות המושתת על נתוני ניווט של פלטפורמות שונות אשר חלקם מהזמן הנוכחי וחלקם לקוח מהעבר, גישה זו לא מתאימה: בנוסף לחוסר ידיעה של זהויות הפלטפורמות המשתתפות במדידה, גם רגעי הזמן של נתוני הניווט אינם ידועים מראש. חישוב ותחזוקה של מטריצת קווריאנס מאוחדת הכוללת בנוסף גם איברי קרוס-קווריאנס בין כל הפלטפורמות השונות בכל אחד מרגעי הזמן האפשריים אינה פתרון ישים.

האלגוריתם השלישי שפותח במחקר זה מציע פתרון לבעיה זו. האלגוריתם מציע לחשב בצורה מפורשת את איברי הקרוס-קווריאנס הנדרשים בשלב עדכון מערכת הניווט עבור מודל מדידה כללי בין מספר פלטפורמות. במודל זה המדידה מושתתת על נתוני ניווט ומדידות של מדידים ממספר פלטפורמות אשר יכולים להילקח מזמנים שונים. מדידה המורכבת מאילוצים של גיאומטריית שתי תמונות, שהוזכרה לעיל, הינה דוגמה פרטית למודל זה. במקרה של מודל מדידה כללי, נתוני הניווט מכל הפלטפורמות המשתתפות במדידה יכולים להיות תלויים סטטיסטית.

סצנה שכבר צולמה בעבר. אולם צוואר הבקבוק בגישת SLAM הינו הדרישות החישוביות הגדלות עם הזמן. למרות שפותחו מספר שיטות מקורבות להקטנת הדרישות החישוביות, פעולה בזמן אמת לאורך זמן ממושך הינה עדיין אתגר בגישת SLAM.

מחקר זה מתמקד, בחלקו הראשון, בפיתוח חלופה לגישת SLAM, המאפשרת לבצע עדכוני מערכת ניווט בזמן אמת בהנחה כי הפלטפורמה מצוידת אך ורק במערכת ניווט אינרציאלית ובמצלמה בודדת, אשר יכולה להיות מורכבת על גבי גימבלים. במחקר מוצע לבצע את שלב המיפוי בתהליך רקע (background process), לאו דווקא בזמן אמת. תמונות המתקבלות מהמצלמה נשמרות במאגר נתונים, יחד עם מידע רלוונטי ממערכת הניווט. מאגר נתונים זה מייצג את שלב המיפוי ויכול לשמש גם לצורך בנייה של תמונות פסיפס. עדכון מערכת הניווט מתבצע בזמן אמת על סמך מידע נוכחי מהמצלמה ומהתמונות, ונתוני ניווט המוצמדים אליהן, הנשלפות ממאגר הנתונים. גישה זו מאפשרת להקטין במידה ניכרת את הדרישות החישוביות בשלב עדכון מערכת הניווט לעומת הדרישות בגישת SLAM.

האלגוריתם הראשון שפותח במחקר זה מתמקד בשיפור ביצועי ניווט בתרחישים מאתגרים, בהקשר של שערוך תנועה מבוסס ראייה ממוחשבת, כגון צילום סצנה דלת מאפיינים על ידי מצלמה עם שדה ראייה צר. תמונות אשר מתקבלות מהמצלמה משמשות לצורך בנייה של תמונת פסיפס, המייצגת את מיפוי הסביבה ובו זמנית משמשת לביצוע עדכוני ניווט. בהנחה כי המצלמה מורכבת על גבי גימבלים, המצלמה מבצעת תהליך של סריקה המאפשר מיפוי של אזורים נרחבים. באלגוריתם מוצע לצמד בין תהליך הסריקה של המצלמה לבין תהליך בניית תמונת הפסיפס. צימוד זה מאפשר להגדיל את אזורי החפיפה בין התמונה הנוכחית לתמונת הפסיפס הקודמת, דבר המניב שיפור בדיוקי שערוך התנועה. בהמשך, מנוסח מסנן קלמן סתום מורחב, implicit extended Kalman filter (IEKF), לצורך עדכון מערכת הניווט עם התנועה המשוערכת. במסנן זה, משוואת המדידה נתונה בצורה לא מפורשת.

בדומה לכלל השיטות לעדכון מערכת הניווט המסתמכות על מידע המגיע משני מבטים (שתי תמונות), גם אלגוריתם זה מסוגל לבצע שערוך תנועת הטרנסלציה של המצלמה רק עד כדי קבוע, דבר המיתרגם ליכולת הקטנה של רק חלק משגיאות הניווט. למשל, שגיאות המיקום והמהירות בכיוון התנועה אינן מוקטנות עקב חוסר אובזרווביליות. בנוסף, האלגוריתם אינו מסוגל לנצל את מלוא הפוטנציאל הטמון במידע שזמין במקרה של תרחישי מעגל שהוזכרו לעיל. סוגיה זו הובילה לפיתוח של אלגוריתם חדש.

באלגוריתם השני, שפותח במסגרת מחקר זה, מוצע לראשונה לשלב אילוצים הנובעים מכך שסצנה סטטית נצפית בשלושה זמנים שונים, לצורך עדכון מערכת הניווט. פותח ניסוח חדש של אילוצים אלו, המשלב תמונות משלושת הזמנים ונתוני ניווט המוצמדים אליהן (חלק מהתמונות נשלפות ממאגר הנתונים המוזכר לעיל). האילוצים שולבו עם מערכת הניווט בעזרת ניסוח חדש של IEKF. האלגוריתם מאפשר הקטנה של שגיאות ניווט, בפרט שגיאות מיקום ומהירות בכל הצירים, בהינתן שלוש תמונות עם אזור חפיפה משותף. לא נדרשות תמונות נוספות לצורך ביצוע עדכון מערכת הניווט בעוד שטיוב המיפוי (המיוצג על ידי התמונות במאגר הנתונים או על ידי

תקציר

ניווט הינו יכולת בסיסית הנדרשת לצורך ביצוע מגוון רחב של משימות על ידי פלטפורמות ניידות. מעבר מנקודה לנקודה בעולם התלת מימדי מחייב, בין השאר, ידיעה של מיקום, מהירות ומצב זוויתי של הפלטפורמה בכל עת. במבט ראשון, חישוב פרמטרים אלו נראה כמו משימה פשוטה: בהנחה כי הפלטפורמה מצוידת במדידי ניווט אינרציאליים (IMU), פתרון הניווט בכל זמן נתון ניתן לחישוב על סמך פתרון הניווט בזמן הקודם והמדידות הנוכחיות של ה-IMU. אולם תהליך זה, המבוצע על ידי מערכת הניווט האינרציאלית (INS), מניב פתרון ניווט הכולל שגיאות שהולכות ומתבדרות עם הזמן. הסיבה העיקרית לשגיאות אלו הינה שגיאות במדידות של המדידים האינרציאליים. לאחר פרק זמן מסוים, התלוי בין השאר באיכות ה-IMU, שגיאות הניווט האינרציאליות יגיעו לרמות שיהיו בלתי קבילות במגוון רחב מאוד של משימות.

כתוצאה מכך, נהוג להיעזר במידע ובמדידים נוספים על מנת לאפס, או לפחות להקטין את שגיאות הניווט האינרציאליות, תהליך הנקרא עדכון מערכת הניווט (navigation aiding). לחילופין, מקורות מידע ומדידים אלו יכולים לשמש לחישוב ישיר של פתרון הניווט. מערכת ה-GPS, מאז כניסתה לפעולה בשנות ה-70 של המאה ה-20, היא ללא ספק השיטה המקובלת במרבית מערכות הניווט המודרניות לצורך ביצוע עדכון מערכת הניווט וחישוב ישיר של פתרון הניווט. אולם, אות ה-GPS אינו זמין, או לחילופין, אינו אמין במגוון תרחישים, כגון: פעולה מתחת למים, בתוך מבנה, בסביבה עירונית ועל פני כוכבי לכת אחרים. במקרים אלו יש צורך בשיטות חלופיות על מנת לקבל ניווט מדויק. בנוסף, שיטות אלו יכולות לשמש כגיבוי למערכת ה-GPS, למקרים בהם אות ה-GPS נהפך ללא זמין בשלב מסוים במהלך המשימה (למשל כניסה לאזור עירוני צפוף).

אבחנה זו והתפתחות היכולת החישובית בשני העשורים האחרונים תרמו לפיתוח שיטות רבות לביצוע עדכון מערכת הניווט המסתמכות על ראייה ממוחשבת. השיטות נבדלות בהנחות לגבי המידע הקיים מראש במערכת, כגון מפות טופוגרפיות ממוחשבות (DTM), ובמדידים השונים שאיתם הפלטפורמה מצוידת, כגון מצלמה בודדת או מספר מצלמות, ומד טווח.

בשנים האחרונות הושם דגש מיוחד בפיתוח יכולת לפעול בסביבה שאינה מוכרת, כלומר סביבה שלגביה לא קיים מידע כלשהו מראש. לעתים, בנוסף לחישוב פתרון הניווט, נדרש למפות את הסביבה הנצפית על ידי המצלמה במשך המשימה. ביצוע שתי המשימות האלו, ניווט ומיפוי, בו זמנית הינה גישה הידועה בתור Simultaneous localization and mapping (SLAM). המפה הנבנית יכולה להיות מיוצגת על ידי מיקומים של נקודות עניין בעולם התלת מימדי, או על ידי תמונת פסיפס (או מספר תמונות פסיפס) הנבנית על סמך התמונות שהתקבלו מהמצלמה. שיטות SLAM נבדלות, בין היתר, בהנחות לגבי המדידים החיצוניים (למשל: מצלמה בודדת, מצלמה בודדת ומד טווח, זוג מצלמות). חוזקה של גישת SLAM מתבטא, בין היתר, ביכולת מובנית לטפל בצורה עקבית בתרחישי מעגל, בהם הפלטפורמה חוזרת, לאחר פרק זמן שאינו ידוע מראש, לצלם

המחקר נעשה בהנחיית פרופ/ח פיני גורפיל, פרופ׳ אהוד ריבלין וד״ר הקטור רוטשטיין בפקולטה לאווירונאוטיקה וחלל.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

ביצועי ניווט משופרים באמצעות בניית פסיפסי תמונות בזמן אמת

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

ואדים אינדלמן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

ניסן תשע"א **חיפה** **אפריל 2011**

ביצועי ניווט משופרים באמצעות בניית פסיפסי תמונות בזמן אמת

ואדים אינדלמן