PhD Dissertation

A GEODETIC APPROACH TO PRECISE, ACCURATE, AVAILABLE AND RELIABLE NAVIGATION

M. Eulàlia Parés Calaf

Advisor:
Dr. Ismael Colomina Fosch
Geonumerics

Tutor:
Prof. Josep Gili Ripoll
Dep. d'Enginyeria del Terreny
Universitat Politècnica de Catalunya

DOCUMENT STATUS

Document title:	PhD dissertation:A geodetic approach to precise, accurate available and reliable navigation
Author:	M. Eulàlia Parés Calaf
Advisor:	Dr. Ismael Colomina Fosch
Tutor:	Prof. Josep Gili Ripoll
Centre:	This thesis was started at former Institute of Geomatics, and has been finished at Centre Tecnològic de Telecomunicacions de Catalunya
PhD program:	Aerospace Science and Technology - UPC
Date:	September 25, 2020

Als meus pares, el Parés i la Montse.
Els vostres somnis, les meves realitats.

Als meus àngels, l'Eduard i en Biel.
Els vostres somriures, la meva motivació.

Castelldefels, Juliol 2020.

Abstract

The determination of a body position, velocity and attitude is the purpose of the navigation techniques.

The most extended navigation systems (INS/GNSS), as any other HW and SW system, provide precise and accurate enough trajectory determinations under the appropriate conditions. However, when those systems acquire data on non-friendly environments the computed navigation solution suffers from unacceptable errors. Since the foundation of the Geodesy and Navigation group of the former Institute of Geomatics, now at CTTC, its researchers have to deal with three fundamental issues: solution performance (precision-accuracy), specially for low-cost systems, reliability and environmentally independent availability.

The main objective of the research presented on this dissertation is to contribute to the adoption of the geodetic method for navigation.The geodetic method is based on a proper problem abstraction, on optimal estimation criteria, on rigorous mathematical modelling, on the use of sufficiently redundant data and on the use of sufficiently heterogeneous data. Since rigorous modelling, redundancy and heterogeneous data have been widely used by the navigation community, this thesis focus on a proper problem abstraction as well as on an optimal estimation criteria.

The first step to prove the geodetic approach suitability has been to design, implement and validate the GEMMA system. The GEMMA system is a set of SW modules allowing the validation of new trajectory determination algorithms. The system is made up of measurement generators, filters and analysers, as well as trajectory generators and analysers and, as its main component, a generic platform for the optimal determination of trajectories (NAVEGA). The main purpose of the signal and trajectory generators and filters is to provide data to test and validate new navigation algorithms. Signal and trajectory analysers are used to characterize the error of data sets. NAVEGA is a software platform for the optimal determination of trajectories or paths of stochastic dynamical systems driven by observations and their associated dynamic or static models. The second step has been the design of a new optimal estimation algorithm that maximise the benefits of redundant systems. Redundancy in the number of available satellites but also redundancy in the number of sensors. The availability of several sensors allows reducing the noise and detecting possible outliers. In this thesis, a new bayesian filter implementation, named Simultaneous Prediction and Filtering (SiPF), providing access to the residuals of redundant measurements is presented. This approach allows to apply all the quality determination geodetic techniques to the navigation solution determination.

The benefits of the previous tools, together with the extended acceptance of the benefits of redundancy, rigorous modelling and heterogeneity lead us to conclude that the geodetic approach is a suitable way to face the navigation problem and to improve its performance, its availability and its reliability.

Resum

La determinació de la posició, la velocitat i l'actitud d'un objecte o ésser és el propòsit de les tècniques de navegació.

Els sistemes de navegació més estesos (INS/GNSS), com qualsevol altre sistema HW i SW, proporcionen trajectòries precises i exactes en les condicions adequades. Però quan aquest sistema es fa servir en entorns desfavorable, la solució pateix errors inacceptables. Des de la fundació del grup de Geodèsia i Navegació de l'antic Institut de Geomàtica, ara al CTTC, els seus investigadors han hagut de fer front a tres aspectes fonamentals: la qualitat de la solució (precisió-precisió) per a sistemes de baix cost, la disponibilitat d'una solució en qualsevol entorn i la seva fiabilitat.

L'objectiu d'aquesta tesi, a nivell general, és contribuir a l'adopció del mètode geodèsic per a la navegació. El mètode geodèsic es basa en una abstracció correcte del problema, en criteris d'estimació òptims, en una modelització matemàtica rigorosa, en l'ús de dades redundants i en l'ús de dades heterogènies . Atès que la comunitat de navegació ha utilitzat àmpliament la modelització rigorosa, la redundància i les dades heterogènies, aquesta tesi se centra en una abstracció adequada del problema, així com en criteris òptims d'estimació.

El primer pas per demostrar aquesta idoneïtat ha estat dissenyar, implementar i validar el sistema GEMMA. El sistema GEMMA és un conjunt de mòduls SW que permeten la validació d'algorismes de determinació de la trajectòria. El sistema està format per generadors de mesures, filtres i analitzadors de senyal, així com generadors i analitzadors de trajectòries i, com a component principal, una plataforma genèrica per a la determinació òptima de trajectòries (NAVEGA). El propòsit principal dels generadors i filtres de senyals i trajectòries és proporcionar dades per provar i validar nous algoritmes de navegació. Els analitzadors de senyals i trajectòries s'utilitzen per caracteritzar l'error dels conjunts de dades. NAVEGA és una plataforma de programari per a la determinació òptima de trajectòries impulsades per les observacions i els seus models dinàmics o estàtics associats. El segon pas ha estat el disseny d'un nou algoritme d'estimació òptim que maximitzi els beneficis dels sistemes redundants. Redundància en el nombre de satèl·lits disponibles però també redundància en el nombre de sensors. La disponibilitat de diversos sensors permet reduir el soroll i detectar possibles errors. En aquesta tesi es presenta un nou filtre bayesià anomenat *Simultaneous Prediction and Filtering*, que proporciona accés als residus de les mesures redundants. Aquest enfocament permet aplicar les eines geodèsiques d'análisis de qualitat a la determinació de la solució de navegació.

Els avantatges de les eines anteriors, juntament amb l'acceptació estesa dels beneficis de la redundància, la modelització rigorosa i l'heterogeneïtat, ens permeten concloure que l'enfocament geodèsic és un enfoc adequat per fer front el problema de la navegació i millorar el seu rendiment, la seva disponibilitat i la seva fiabilitat.

Resumen

El objetivo de las técnicas de navegación es la determinación de la posición, velocidad y actitud de un objeto o ser.

Los sistemas de navegación más extendidos (INS / GNSS), como cualquier otro sistema HW y SW, proporcionan determinación de trayectorias precisas y exactas cuando adquieren datos en entornos amigables. Sin embargo, también se introducen errores inaceptables cuando esos entornos no son tan favorables. Desde la fundación del grupo de Geodesia y Navegación del antiguo Instituto de Geomática, ahora en el CTTC, los investigadores tienen que lidiar con tres cuestiones fundamentales: la calidad de la solución (precisión-exactitud) especialmente para sistemas de bajo coste y la disponibilidad y la fiabilidad de los equipos en cualquier entorno.

El objetivo principal de esta investigación es contribuir a la adopción del método geodésico para la navegación. El método geodésico se basa en una abstracción correcta del problema, en una estimación óptima de la solución , en rigor matemático, en el uso de datos redundantes y en el uso de datos hetereogéneos. Dado que la comunidad de navegación ha utilizado ampliamente la modelización rigurosa, la redundancia y los datos heterogéneos, esta tesis se centra en la abstracción adecuada del problema, así como en los criterios de estimación óptimos.

El primer paso para probar esta idoneidad ha sido diseñar, implementar y validar el sistema GEMMA. GEMMA es un conjunto de módulos SW que permite la validación de nuevos algoritmos de determinación de trayectoria. El sistema está compuesto por generadores y filtros de señal, generadores y analizadores de trayectorias y, como componente principal, una plataforma genérica para la determinación óptima de trayectorias (NAVEGA). El objetivo principal de los generadores y filtros de señal y trayectoria es proporcionar datos para probar y validar nuevos algoritmos de navegación. Los analizadores de señal y trayectoria se utilizan para caracterizar el error de los conjuntos de datos. NAVEGA es una plataforma de software para la determinación óptima de trayectorias o de sistemas dinámicos estocásticos impulsados por observaciones y sus modelos dinámicos o estáticos asociados.El segundo paso ha sido el diseño de un nuevo algoritmo de estimación óptimo que maximiza los beneficios de los sistemas redundantes. Redundancia en el número de satélites disponibles, pero también redundancia en el número de sensores. La disponibilidad de varios sensores permite reducir el ruido y detectar posibles valores atípicos. En esta tesis, se presenta una nueva implementación de filtro bayesiano, llamado *Simultaneous Prediction and Filtering (SiPF)*, que proporciona acceso a los residuos de mediciones redundantes. Este enfoque permite aplicar todas las técnicas geodésicas conocidas de determinación de la calidad a la determinación de la solución de navegación.

Los beneficios de las herramientas anteriores, junto con la aceptación extendida de los beneficios de la redundancia, el modelado riguroso y la heterogeneidad, nos llevan a concluir que el enfoque geodésico es una manera adecuada de enfrentar el problema de navegación y mejorar su calidad, disponibilidad y confiabilidad.

Contents

Contents 11

List of figures 13

List of tables 15

Definitions 17

Notation 19
Conventions . . . 19
Coordinate frames . . . 19
Acronyms . . . 22

1 Introduction 23
1.1 Motivation . . . 23
1.2 Research objectives and contributions . . . 24
1.3 Outline of the dissertation . . . 24

2 Inertial navigation technology - Theoretical basics 27
2.1 Autonomous navigation . . . 27
2.2 Hybridization . . . 28

3 Related work 31
3.1 Navigation problem abstraction and modellization . . . 31
3.2 Optimal estimation criteria and quality assessment . . . 32

4 Problem abstraction - A generic, extensible and modular multi-sensor navigation analysis system 35
4.1 GEMMA overview . . . 35
4.1.1 Trajectory generation . . . 36
4.1.2 Signal generators . . . 37
4.1.3 Sequencer . . . 37
4.1.4 Trajectory estimation . . . 38
4.1.5 Analyzer . . . 38
4.2 GEMMA use cases . . . 38
4.2.1 Use case: technology selection or validation . . . 39
4.2.2 Use case: algorithm verification and validation . . . 40
4.2.3 Use case: navigation and positioning in real-life environments . . . 41

5 Problem abstraction - Geodetic tools for trajectory determination and sensor simulation 43
5.1 Trajectory determination tool - NAVEGA . . . 43
5.1.1 Next generation trajectory determination systems requirements . . . 43
5.1.2 Next generation systems architecture principles . . . 44
5.1.3 Proposed architecture definition . . . 45
5.1.4 A sample implementation . . . 49
5.2 Inertial sensors simulation tool - IMU simulator . . . 50
5.2.1 System requirements . . . 50
5.2.2 System architecture . . . 51
5.2.3 System verification . . . 54
5.2.4 System validation . . . 54

6 Optimal estimation criteria and quality assessment - Simultaneous prediction and filtering **57**
6.1 Theoretical approach . . . 57
6.1.1 Considerations on navigation sensors outliers . . . 57
6.1.2 Simultaneous Prediction and Filtering theoretical basis . . . 58
6.1.3 Considerations on practical implementation . . . 60
6.1.4 Application to integrity monitoring of multi-sensor navigation systems . . . 61
6.2 Concept validation . . . 62
6.2.1 Validation approach . . . 62
6.2.2 Validation data sets . . . 62
6.2.3 Validation results . . . 64
6.2.4 Discussion . . . 65

7 Rigorous modelling, redundancy and hetereogeneity - Benefits **67**
7.1 Modelling and rigorous modelling . . . 67
7.1.1 IMU characterization . . . 67
7.1.2 GNSS' Clock characterization . . . 67
7.2 (INS) Redundancy . . . 68
7.3 Hybridization . . . 68

8 Conclusions **69**
8.1 Thesis outcomes . . . 69
8.2 Thesis impact on the design of accurate and reliable navigation systems . . . 70

9 Outlook **71**

Acknowledgements **73**

GEMMA enabled projects and works **75**

GEMMA interface control **77**
ASTROLABE: A Rigorous, Geodetic-Oriented Data Model for Trajectory Determination Systems . . . 77

Bibliography **109**

List of Figures

2.1 Hybridization levels: uncoupled. 29
2.2 Hybridization levels: loosely (left) and closely (right) coupled. 29

4.1 GEMMA: components and work flow. 36
4.2 GEMMA's trajectory analyser: Example of quality plot. 38
4.3 GEMMA use case: the technology selection. 39
4.4 GEMMA use case: algorithm verification and validation. 40
4.5 GEMMA use case: real-time environments. 41

5.1 Trajectory determination SW architecture components diagram. 46
5.2 Image of a set of inertial systems: from navigation-grade to MEMS. 50
5.3 IMU_simulator: error components and subcomponents. 52
5.4 IMU_simulator: UML activity diagram. 52
5.5 IMU_simulator: Errorless simulated angular velocities for an IMU fixed at 41º lat, 0º lon, 0 m height. 54
5.6 IMU_simulator: computed stochastic errors. 55

6.1 SiPF: IMU outliers and their impact on velocity and position estimations. 58
6.2 SiPF: OUtlier in odometer data. 58
6.3 SiPF: classical prediction-correction flowchart 60
6.4 SiPF: proposed flowchart 61
6.5 SiPF: algorithm's validation reference trajectory. 63
6.6 SiPF validation: GNSS/RINS/magnetometer solution errors without (left) and with SiPF(right). . . . 65

List of Tables

5.1 Trajectory determination tool requirements vs architecture principles traceability matrix. 46
5.2 IMU_simulator: instrumental errors. 55
5.3 IMU_simulator: environmental errors 56
5.4 IMU_simulator: platform errors (PSD) 56

6.1 SiPF validation: sensors' specification. 63
6.2 SiPF validation: datasets sum up. 63
6.3 SiPF validation: Standard deviation of the residuals of outliers free testing data sets. 64
6.4 SiPF validation: GNSS/INS/odometer solution errors for different modes 64
6.5 SiPF validation: GNSS/RINS/magnetometer solution errors for different modes 65

Definitions

The first lesson the author of this thesis learned from navigation and geodetic communities is that a single concept can be named with several words and, for more entanglement, the same word can be used with quite different meanings for different scientific communities. This section tries to avoid misunderstandings regarding main thesis concepts. Thus, hereafter, the reader will find the concept behind each key word of this report.

Attitude is the orientation of a system in relation to a reference frame.

A **geodetic coordinate reference frame** is a combination of a geodetic reference frame plus a geodetic coordinate system.

A **geodetic coordinate reference system** is a combination of a geodetic reference system plus a geodetic coordinate system plus a (selected - implicit) geodetic reference frame.

A **geodetic coordinate system** is a coordinate system used to describe a position on or near the Earth's surface.

A **geodetic reference frame** is a realization of a reference system through a set of points whose coordinates are monumented or otherwise observable in or near the Earth's surface. In other words it is a "list" of points with their position and velocity coordinates (e.g. ITRF96 of ITRS.)

A **geodetic reference system** is a reference system used to define at any time a triad of axes on or near the Earth's surface (e.g. ITRS.)

Global Navigation Satellite System (GNSS) is a system of satellites that are continuously transmitting (electromagnetic) signals that allow the users to determine their position over the Earth surface.

An **Inertial Measurement Unit (IMU)** is a device composed by one clock, several angular rate sensors (usually three), several accelerometers (usually three) and a mechanical holding structure plus the needed electronics and communication systems to provide data. The data provided by an IMU is: time, angular rates (one for each angular rate sensor) and linear accelerations (one for each accelerometer.)

An **Inertial Navigation System (INS)** or **Inertial Navigation Unit (INU)** is a device composed by one IMU plus a navigation processor. The data provided by an INS is: time, position, velocity and attitude.

Inertial positioning/navigation is positioning/navigation based on inertial sensors.

An **inertial sensor** is a device such that reacts on the basis of Newton's laws of motion. There are two types of motion, translational and rotational motion, thus, there are two types of inertial sensors: **accelerometers** and **angular rate sensors** (such as gyroscopes.) The **accelerometers** measure the linear accelerations (forces) or the instantaneous rate of change of velocity suffered by the sensor (translational movement.) The angular rate sensors measure the instantaneous angular velocities suffered by the sensor (rotational movement.)

Measurement is the assignment of a number to a characteristic of an object.

Model is a mathematical description of a system or a proess.

Navigation is real time positioning.

Observation is a numerical property of a physical system that can be determined by a sequence of phisical or mathematical operations

Position is the place where something or someone is, often in relation to other things.

Positioning is to know a system position, velocity and attitude.

A **Redundant Inertial Measurement Unit (RIMU)** is a device composed by one clock, several angular rate sensors (more than three), several accelerometers (more than three) and a mechanical holding structure plus the needed electronics and communication systems to provide data. The data provided by a RIMU is: time, angular rates (one for each angular rate sensor) and linear accelerations (one for each accelerometer.)

A **reference frame** is a realization of a reference system through a set of points whose coordinates are observable.

A **reference system** is a definition, a set of prescriptions and conventions together with the modelling required to define at any time a set of axes.

The orientation parameters, orientation elements, **transformation** or transfer **parameters** are a set of values which define the relationship between two reference systems (or frames.)

Velocity is a vector that denotes the change of position within a time rate with a directional component.

Notation

The second lesson that the author learned from the above mentioned communities is that there are not a unique notation to represent navigation related physical and mathematical concepts. In this section the reader will find the notation used along this dissertation.

Conventions

Vectors are represented by lower-case letters. x

Reference frames. A superscript denote the reference frame in which the vector is represented. x^t

First derivative are represented with a dot. $\dot{x}^t$

Second derivative are represented with a double dot. $\ddot{x}^t$

Matrices are represented by upper-case letters. R

Rotation matrices. Superscripts and subscripts indicates respectively the "to" and "from" frames of rotation matrices. R_s^t

Angular velocity of the t-frame with respect to the s-frame, expressed in the t-frame may be expressed either by an angular velocity vector (ω_{st}^t) or by the corresponding skew-symmetric matrix (Ω_{st}^t).

$$\Omega_{st}^t = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

Coordinate Frames

Body frame

A body frame is one directly defined on the object/instrument of interest.

ref. system	instrumental system
ref. frame	any realization of the system
pos. coord. system	cartesian forward, right, down
vel. coord. system	cartesian forward, right, down
notation- superscript	b**-frame**
position and velocity vector	$\left(\mathbf{x}^b, \dot{\mathbf{x}}^b\right)$

Local frame

A local frame is one based on the ground, typically near the object/instrument of interest.

ref system	local
ref frame	any realization of the system
pos. coord. system	cartesian North, East, Down
vel. coord. system	cartesian North, East, Down
attitude "from" ref. frame	body frame
notation- superscript	l**-frame**
navigation vectors	$\left(\mathbf{x}^l, \dot{\mathbf{x}}^l, \mathbf{R}_b^l\right)$

Inertial frame

An inertial frame is one in which bodies follow the Newton's laws of motion.

ref system	ICRS: International Celestial Reference System
ref frame	quasars and stars
pos. and vel. coord. system	cartesian
attitude "from" ref. frame	body frame
notation- superscript	i**-frame**
navigation vectors (pos, vel, att)	$\left(\mathbf{x}^i, \dot{\mathbf{x}}^i, \mathbf{R}_b^i\right)$

Earth frame

An Earth frame is one based on the Earth.

ref system	ITRS: International Terrestrial Reference System
ref frame	any of the family of ITRFn
pos. and vel. coord. system	cartesian
attitude "from" ref. frame	body frame
notation- superscript	e**-frame**
navigation vectors (pos, vel, att)	$(\mathbf{x}^e, \dot{\mathbf{x}}^e, \mathbf{R}_b^e)$

Navigation frame

This is a mixed frame useful for navigation/positioning data representation.

	Navigation
ref system	ITRS: International Terrestrial Reference System
ref frame	any of the family of ITRFn
pos. coord. system	geodetic
vel. coord. system	cartesian
attitude "from" ref. frame	body frame
navigation vectors	$\big((\lambda, \varphi, h)^e, \mathbf{R}_e^l \dot{\mathbf{x}}^e, \mathbf{R}_b^l\big)$

where λ^e and φ^e are the geodetic latitude and longitude,

$$\mathbf{R}_e^l \dot{\mathbf{x}}^e = \begin{pmatrix} v_n \\ v_e \\ v_u \end{pmatrix}$$

and

$$\mathbf{R}_b^l = R_{b_2}^l(\gamma) R_{b_1}^{b_2}(\psi) R_b^{b_1}(\vartheta)$$

where γ, ψ, ϑ are the heading, pitch and roll Euler angles.

Acronyms

API	Application Programming Interface
AV	Allan Variance
CTTC	Centre Tecnológic de Telecomunicacions de Catalunya
COTS	Commercial Off-The-Shelf
DoCTA	Doctorat en Ciència i Tecnologia Aeroespacial
EKF	Extended Kalman Filter
GEMMA	Generic Extensible and Modular Multi-sensor navigation Analysis system
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HEPF	Hybrid Extended Particle Filter
HW	Hardware
ICRS	International Celestial Reference System
IEKF	Iterated Extended Kalman Filter
IG	Institut de Geomàtica
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
INU	Inertial Navigation Unit
ITRF	International Terrestrial Reference Frame
ITRF	International Terrestrial Reference System
KF	Kalman Filter
LS	Least Squares
MEMS	Microelectromechanical systems
NA	Network Approach
PF	Particle Filter
PIKF	Predictive Iterated Particle Filter
PSD	Power spectral Densisty
RF	Radio Frequency
RIMU	Redundant Inertial Measurement Unit
SiPF	Simultaneous Prediction and Filtering
SSA	State-Space Approach
SW	Software
UAV	Unmanned autonomous Vehicle
UKF	Unscented Kalman Filter
UPC	Universitat Politècnica de Catalunya

Chapter 1

Introduction

1.1 Motivation

During the last decade navigation and orientation have become an indispensable technology for most of the developed countries citizens. With the popularization of smartphone devices and location based applications, the technology becomes not only a pre-requisite for the exploitation of remote sensing but also for the citizens daily needs. Nowadays, navigation is applied in such different applications as airborne photogrammetry and remote sensing, airborne and marine bathymetry, autonomous driving and smart mobility, railway track geometry surveying, motorcycle and car racing trajectory analysis, stabilization of moving video cameras in sports' journalism or pedestrian localization. This heterogeneous demand translated in higher requirements to navigation systems. As stated in [30] and [31], the positioning community have to deal continuously with new sensors and new performances. Furthermore, citizens and professionals demand positioning everywhere. But, the widely-used INS/GNSS (Inertial Navigation Systems / Global Navigation Satellite Systems) solution is not able to provide acceptable trajectories in all environments.

For example, the situation in two completely different applications is the following:

- **Low-cost surveying - UAV navigation.** Although medium and high altitude autonomous flight is a fact [14], low-altitude autonomous flights are still a threat for citizens security [56] because navigation integrity for drones is an open issue. The European Commission, at the ending of H2020 programme, is still funding research and development projects focused on this topic (e.g. the REALITY project [26]). According to the Commission [80], in 2050 there will be more than 400000 commercial drones operating on the European Union. Thus, special attention must be paid to system reliability and to system robustness. Up to now, almost all the efforts have been devoted to GNSS integrity in any of its approaches [37]: Stellite-Based Augmentation systems, Ground-based augmentation systems or Vehicle-based Augmentation systems. But infinitely less attention has been paid to IMU integrity, even when this is the cause of the Ariane 5 [19] or the Long March 3B [62] failures. Regarding reliability, currently, the main weakness is due to non rigorous functional and stochastic modelling. As far as IMU modelling is concerned, stochastic modelling of low-cost Inertial Measurement Units (IMUs) is still an open issue [75] [11]. Robustness problem is mainly related to the lack of enough and complete system information.

  Thus, improvements on system reliability and robustness are still a must.

- **Geodetic surveying - Kinematic airborne gravimetry.** cm-level GNSS requires worldwide cm-level geoid. This is an unachievable goal unless aerial measurements are taken. Recent technological advances in INS/GNSS allow to carry out precise successful geoid determination kinematically [9][40][84]. For those campaigns to be accurate and precise enough very accurate and precise GNSS trajectory determination is needed. Thus, the use of Differential GNSS techniques involving observations from static/master and kinematic receivers is still a must. In non developed countries, were master receivers are not widely spread, this is a requirement difficult to fulfil.

  Therefore, improvements in INS/GNSS accuracy and precision are still a must [84].

And also, the situation in two completely different scenarios is the following:

- **Denied environments - Seamless navigation.** Actual demands of environmentally independent position solution highlighted the indoor/outdoor continuity problematic [6]. Indoor navigation accuracy should be consistent with outdoor GNSS-based positioning. Navigation research groups work with several hybridization approaches, involving, autonomous and non-autonomous systems, but there is not yet an approach providing fully available, robust and with an acceptable performance solution for all environments and applications [2].

To sum up, improvements on availability, robustness and performance are still a must.

- **Semi-denied environments - positioning in terrestrial corridors.** In semi-denied environments INS/GNSS hybridization main problem to face up with is the reliability of data [22] [97]. Specifically, the outlier detection while working with GNSS pseudoranges or positions. Working with simulated data shows promising results, but simulated data does not take into account the full complexity of GNSS real data. Thus real data presents several problems not taken into account in the classical methods for INS/GNSS hybridization.

  Therefore, improvement in gross error detection are still a must.

An analysis of the previous examples lead us to consider the following navigation challenges for the near future: solution **performance** (precision-accuracy) for low-cost systems, environmentally independent **availability** and **reliability**. Although they may seem basic problems they are complex enough to be the research target of tenths of groups around the world.

1.2 Research objectives and contributions

On the general level, the research presented in this dissertation aims at contributing to the adoption of the geodetic method for navigation and to its application/validation to navigation.

The geodetic method is based on a properly problem **abstraction**, on **optimal estimation criteria and quality assessment**, be it by internal or external means, on **rigorous mathematical modelling**, on the use of **sufficiently redundant data** and on the use of **sufficiently heterogeneous data**. General improvement of the navigational performance, availability and reliability is a consequence thereof.

- Abstraction means the identification of the common traits defining the essence of the information to be deal with. Once those traits are identified the problem can be modelled as a generic one. The generic approach allows to analyse all the navigation problems as a single one. This simplification has a direct impact on algorithms design and facilitates the integration of several solution approaches.
- Optimal estimation, for given optimality criteria, is an algorithmic and numerical issue; namely, a matter of finding and implementing an algorithm that provides the "optimal" solution. Quality assessment is inevitably related to the optimality criteria but can go beyond it by further concept elaboration. An example is the geodetic theory of internal and external reliability, related to regression diagnostics of statistical testing. Other examples are the observability and controllability concepts of optimal sequential estimation.
- Rigorous modelling refers to the identification of models that approximate the physical reality beyond the sensitivity of current technology within the appropriate application context. Mathematical modelling must be both functional and stochastic. They are mutually dependent as correctness in the stochastic aspect uses to be related to simplicity in the functional formulation. we note here that the closer the model is to raw measurements, the simpler the modelling. Thus, a general objective, is to build models upon independent measurements (uncorrelated) that can be stochastically modelled with simple/well known distributions.
- The concept of redundant and heterogeneous data includes a number of principles from the actual redundancy in measurements, to the geometry of these measurements to the heterogeneity and complementarity of the measurement devices or sensors.

This work aims to prove that the full geodetic method is a suitable approach to face current navigation challenges. Since rigorous modelling, redundancy and heterogeneous data have been widely used by the navigation community, this thesis focus on a proper problem abstraction as well as on optimal estimation criteria and quality assessment.

1.3 Outline of the dissertation

Within this report the reader will firstly find an introductory chapter (chapter 2) describing the fundamentals of inertial and hybrid GNSS/INS navigation. The chapter, which is not a review of the state of the art, is intended to help further reading of the rest of the report. The next chapter (chapter 3) is devoted to review works related with this thesis' focus, this is, a review on navigation tools and optimal estimation models. Again, this chapter is not a review of the state of the art of the Kalman filter, but rather an exercise to put in context the work reported in chapters 5 and 6. After that, in (chapter 4) the principles behind the GEMMA system, a system for trajectory determination algorithms verification and validation are presented. The GEMMA tools have been instrumental in this research. The two relevant components of GEMMA, the simulation and the trajectory determination tools are described in the next chapter (chapter 5.) Later on, a new geodetic estimation method, the Simultaneous Prediction and Filtering

(SiPF) method is described in detail (chapter 6.) The SiPF is the main contribution of this research as it introduces a novel way to detect outliers in the dynamic measurements —i.e. in the measurements that participate in SDEs— for the upcoming low-cost and redundant low-cost sensor systems. Subsequently, several case studies on rigorous modelling, redundancy and hybridisation (chapter 7) are presented. The report ends with a summary of the main conclusions from the research (chapter 8) and several recommendations for future works are presented (chapter 9.) In the appendices, the reader will find materialization cases of the previous work, from a list of beneficiary projects and works to a generic data model which is also a generic data interface model for sequential estimation methods based on measurements and observation equations. The later is a published journal article.

Chapter 2

Inertial navigation technology - Theoretical basics

This chapter presents the theoretical basis of inertial navigation technology. First the navigation principles are presented, later on, the reader will find the fundamental concepts related to inertial hybridization with other sensors.

2.1 Autonomous navigation

Autonomous navigation is the process of estimating one's current position based upon a system non depending on external aiding, this is, all the processed information is collected by the system itself. Some examples of autonomous navigation systems are altimeters, barometers, odometers or inertial sensors.

Most of the classical autonomous navigation methods are based on dead reckoning methods. This is, processes of estimating position based upon a previously determined position and advancing that position based upon known or estimated speeds over elapsed time. A disadvantage of dead reckoning is that since new positions are calculated solely from previous positions, the errors of the process are cumulative, so the error in the position fix grows with time.

While traditional methods of dead reckoning are no longer considered primary means of navigation, modern Inertial Navigation Systems (INS) are widely used.

The INS base their solutions in data provided by inertial sensors. An inertial sensor is a device such that reacts on the basis of Newton's laws of motion. There are two types of motion, translational and rotational motion, thus, there are two types of inertial sensors: accelerometers and angular rate sensors (such as gyroscopes.) The accelerometers measure the linear accelerations (forces) or the instantaneous rate of change of velocity suffered by the sensor (translational movement.) The angular rate sensors measure the instantaneous angular velocities suffered by the sensor (rotational movement.)

Typically, a combination of three orthogonally mounted angular rate sensors and three orthogonally mounted accelerometers is used to acquire the data for inertial positioning. These devices use to be quite standard and are known as Inertial Measurement Units (IMUs). An IMU is composed by one clock, several angular rate sensors (usually three), several accelerometers (usually three) and a mechanical holding structure, plus the needed electronics and communication systems to provide data. The data provided by an IMU is: time, angular rates (one for each angular rate sensor) and linear accelerations (one for each accelerometer.)

From the knowledge of linear accelerations, angular velocities and an initial position, velocity and attitude, it is possible to compute the position, velocity and attitude of the system in a posterior time.

According to Newton-Euler's second Law:

$$\ddot{x}^i = f^i + \overline{g}^i(x^i) \tag{2.1}$$

where f^i are the sensed linear accelerations in an inertial frame, x^i is the position and $\overline{g}^i$ is the acceleration due to the gravitational field at the point x^i [14].

But, due to the fact that the measures are referred to the body frame and taking into account coordinate ref-

erence frames transformation basis, 2.1 is expressed as

$$\begin{aligned} \ddot{x}^i &= R_b^i f^b + \overline{g}^i(x^i) \\ \dot{R}_b^i &= R_b^i \Omega_{ib}^b \end{aligned} \tag{2.2}$$

or equivalently:

$$\begin{aligned} \dot{x}^i &= v^i \\ \dot{v}^i &= R_b^i f^b + \overline{g}^i(x^i) \\ \dot{R}_b^i &= R_b^i \Omega_{ib}^b \end{aligned} \tag{2.3}$$

where f^b is the sensed linear accelerations, Ω_{ib}^b is the matrix containing the sensed angular velocities, x^i is the position, v^i is the velocity, R_b^i is the rotation matrix between the i-frame and the b-frame (parametrized by the angles heading (γ), pitch (ψ), roll (ϑ)) and $\overline{g}^i$ is the acceleration due to the gravitational field at the point x^i [14].

Due to the problem nature, is it clear that the solution will be affected by integration drifts. But this is not the only error affecting the solution, since he IMU sensor's data is affected of errors such as white noise, biases or scale factors, the delivered solution will be also affected by this type of errors.

2.2 Hybridization

In order to determine the IMU sensors drifts and biases the INS solution can be compared with other positioning sensors solution [90] or with the carrier vehicle dynamic model [41]. This way, once the INS errors are deduced, the INS solution can be corrected. Thus, the equation 2.3 are modified as follows:

$$\begin{aligned} \dot{x}^i &= v^i \\ \dot{v}^i &= R_b^i \hat{f}^b + \overline{g}^i(x^i) \\ \dot{R}_b^i &= R_b^i \hat{\Omega}_{ib}^b \end{aligned} \tag{2.4}$$

where $\hat{f}^b$ is the linear accelerations obtained after correcting the sensed ones with the estimated errors and $\hat{\Omega}_{ib}^b$ is the matrix containing the corrected sensed angular velocities.

In order to determine the INS error, altimeters and the Global Positioning System (GPS) were the main auxiliary sensors until the first decade of the XXI century. Recently, due to the appearance of new positioning constellations and the increasing demand of indoor positioning, GNSS (Global Navigation Satellite Systems), RF (Radio Frequency) systems and visual-based positioning solutions (like RGB stereocarmeras or LIDARs) are also common tools to solve INS drifts.

Depending on which auxiliary sensor observation is used to update the solution and how the output parameters are used to correct input data, we talk about different hybridization levels. Thus, an uncoupled algorithm takes into account IMU data plus auxiliary sensor computed position, velocity and/or attitude and determines position, velocity and attitude.(Figure 2.1.) A loosely coupled system also uses IMU data plus auxiliary sensor computed position, velocity and/or attitude. It computes not only position, velocity and attitude but also IMU systematic errors (Figure 2.2.) A closely coupled system uses IMU data and auxiliary sensor raw data. It also computes position, velocity, attitude, IMU systematic errors and auxiliary sensor systematic errors (Figure 2.2.).

The most extended approach to merge INS and other sensors solutions is the Bayesian approach [51], in any of its embodiments. This solution is based on the fact that the navigation problem can be written as:

$$\begin{aligned} x_k &= f_k(x_{k-1}, v_{k-1}) \\ z_k &= h_k(x_k, w_k) \end{aligned} \tag{2.5}$$

Where the error model is described by (1) an initial distribution $p(x_0)$, (2) a transition equation $p(x_k|x_{k-1})$ and (3) an initial distribution $p(z_k|x_k)$. Bayesian theory says that:

$$E(x_k) = E(f_k) = \int_{t_{k-1}}^{t_k} f_k(x_k) p(x_k|z_k) dx_k \tag{2.6}$$

Independently of the involved sensors and on the hybridization criteria, the navigation problem is as simple and as complex as being able to properly compute this value.

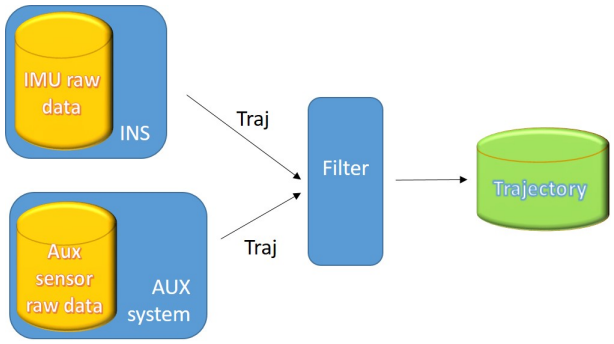

Figure 2.1: Hybridization levels: uncoupled.

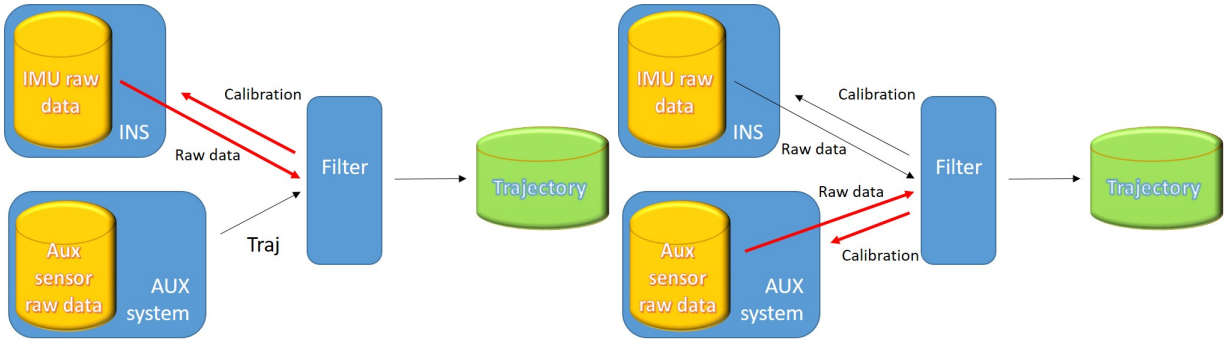

Figure 2.2: Hybridization levels: loosely (left) and closely (right) coupled.

In the next chapter the reader will find how navigation community has typically faced this problem and in the following ones the reader will find an abstraction of the overall problem that simplifies model implementation and development and a new numerical approach to solve this equation taking into account the geodetic principles.

Chapter 3

Related work

In the previous chapter, a review of the technological basis of navigation technology has been presented. In this section, some of the most recent advances in the navigation algorithms and methods related to the present reseach are summarized.

3.1 Navigation problem abstraction and modellization

From a research point of view, multisensor navigation field is not only trajectory determination systems but ancillary tools such as reference trajectory generators, signal generators and trajectory analysis/validation tools. Currently, a researcher can find a wide range of helpful tools available both in the "traditional" market as in the open-source / free license market. But all those tools are designed as individual tools and are not conceived as a part of a whole. Thus, for example, we can easily find matlab code for the generation of synthetic trajectories [52]; for IMU measurements generators [53]; we can find high grade software for GNSS signal generation, like the proviedd by IFEN [39]; Open-source GNSS research tools, like the CTTC's GNSS software defined radio receiver, named GNSS-SDR [18]. Researchers can also find reliable and wordwide trajectory determination and analysis systems like the Applanix or the Waypoint systems [17]. But, even with this wide range of tools, even in the best of the scenarios, the researcher would need to deal with interfaces incompatibilities, and when trying to develop new algorithms he/she can even need to develop a full trajectory determination system by him or herself.

The research presented in this dissertation aims to develop a geodetic-based, generic, useful (modular and extensible,) and complete chain of tools for multisensor navigation research purposes (see chapter 4.)

Focusing on trajectory determination systems, an overview on current available solutions reveals that the market for traditional surveying applications is dominated by a few players, mainly: Applanix (Canada), OXTS (UK), IGI GmbH (Germany), uBLOX (Switzerland), Civitanavi Systems (Italy) and NovAtel (Canada). The first four companies provide integrated systems, including both HW and SW components that may not be acquired separately. The software components are, moreover, monolithic real-time navigators or post-mission orientators. NovAtel provides with integrated HW and SW solutions as well (the SPAN family), but also offers independent SW systems (GrafNav and Inertial Explorer). Yet, even in this case, these SW components are just navigators/orientators. The market segment covering new devices like smartphones, car driving assistance or pedestrian navigation is currently dominated by Garmin (Switzerland), Tom Tom (Netherlands), Magellan (Japan), Broadcom (USA) and Invensense (USA). Once again, these are closed systems that do not allow the integration of new sensors. The consolidation and evolution of smartphones paves the way to a new navigation environment. Handheld devices provide the HW and the SW APIs while SW developers provide tools specifically developed for this kind of systems. The efforts to develop such a wide range of tools (from professional, to car-oriented or smartphones devices) may be measured in thousands of person/months for each company, and will keep being like that unless a common platform is developed for all of them. Beyond the traditional market, it is possible to find now open source tools like RTKlib and GPStoolkit library, able to provide good solutions for a single specific sensor, the GNSS receiver.

All those systems rely on a Kalman Filter family estimation criteria, they implement what is named a State-Space Approach (SSA). One of the main research lines of the former Institute of Geomatics was to develop a fully different approach, the Network Approach (NA), based on geodetic concepts and principles. In the geodesy and geomatics field, a network is a set of instruments, observations and parameters that are inter-related through mathematical models. A time-dependent network is a network where some of its parameters are stochastic processes. To solve a network is to perform an optimal estimation of its parameters in the sense of least-squares. Analogously, to solve a time-dependent network is to perform an optimal estimation of its parameters which include some stochastic process. In the network approach, the navigation equations system are considered as a time-dependant network and solve them

by means of the least-squares method. Similar approaches to NA can be found in other research communities [15]. For example, in the robotics and computer vision community, this approach is known as bundle adjustment or sliding window bundle adjustment [46].

The main pros and cons of both approximations are presented hereafter.

- State-space approach
  - Advantages
    - Real-time parameter estimation capability.
    - The state vector dominates the scene. That is, there is a clear definition of what the system is.
  - Disadvantages
    - Connectivity of parameters through static observation equations is not supported.
    - Filter divergence.
    - Computation of covariance matrices for all the state vectors can not be avoided.
- Network approach
  - Advantages
    - Support for connectivity of parameters regardless of time.
    - Support for both traditional networks and for dynamic systems.
    - Possibility to compute the covariance of a limited number of selected parameters.
    - Variance component estimation.
  - Disadvantages
    - Large system of linear equation.
    - Real-time parameter estimation not feasible in general.

The research presented in this dissertation aims at demonstrate the validity of the NA philosophy but within a SSA (see section *5.1*.)

3.2 Optimal estimation criteria and quality assessment

Nowadays, trajectory determination for navigation, geodetic positioning and remote sensing orientation is mainly based on the Bayesian approach, mainly represented by the Kalman filtering and Smoothing published by R.E. Kalman [42]. Their methodology is to solve navigation problem by prediction, filtering and smoothing. The optimal solution to the prediction-filtering smoothing is obtained through one of the recursive algorithms of the Kalman filter type. In the prediction-filter cycle, the most important entity is the state-vector, in some situation this could represent a problem, due to problems unobservability. High end systems are commonly implemented using Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). The EKF assumes linear process and measurement models while UKF generates sigma points using the real mean and standard deviation of data. To overcome these limitations, a non-parametric filter, the Particle Filter (PF) was proposed lately. The PF can easily deal with non-linearity and non-Gaussian noises.In [1] Aggarwal et al. propose a Hybrid Extended Particle Filter (HEPF) as an alternative to the EKF to achieve better navigation data accuracy using low-cost MEMS sensors. HEPF combines the advantages of Extended Kalman Filter and Extended Particle Filter by alternating between the two filters, based on system dynamic. In [21], Fang et al. propose the Predictive Iterated Kalman Filter (PIKF). The PIKF estimates the model error by the PF and uses it to compensate the state estimation.

Although optimal on friendly environments,a weakness of the Kalman filters is its behavior in the presence of outliers. In such situations the performance of those filter degrades. In INS/GNSS integration, well-known examples of those outliers are GNSS signals affected by multipath but there are several scenarios, less known, where IMU measurements are also affected by outliers. For example, the signals outside the dynamic range of the IMU will not be collected properly. Another example of incorrect measurements will be those signals whose frequency are higher that the IMU acquisition frequency. A method widely extended to face these situations are the adaptive filters. Adaptive filters are based on the robust least-squares philosophy. Those filters reweigh the measurements in real time allowing to down-weight an outlier when it arrives [89]. Adaptive filters have been proven very useful for detection of outlying measurements involved in the filtering step but not the ones involved in the prediction step.

All those algorithms are real-time oriented, where data is processed sequentially in time. The quality assessment tools for those algorithms are based on innovation analysis and observability and controllability analysis. They do

not take full advantage of modern mission configurations, where redundancy and heterogeneity are common. In [15], Colomina et al. proposed the network approach, a post-processing oriented algorithm where all available data is processed together in a batch mode and where this information is really taken into account. In this approach the geodetic analysis tools (internal and external reliability, outlier detection) have been fully exploited and have demonstrated its suitability to improve solution reliability. Unfortunately, since this is a bath approach, it implies enormous computational burden.

The research presented in this dissertation aims to develop a new sequential optimal estimation criteria that takes benefits of problem geometry and able to use geodetic analysis tools (see chapter 6).

Chapter 4

Problem abstraction - A generic, extensible and modular multi-sensor navigation analysis system

In this chapter a full navigation research framework named GEMMA (Generic, Extensible and Modular Multi-sensor navigation Analysis system) is presented. GEMMA is a generic, extensible and modular system that allow to implement all the navigation research required tools, in particular all the algorithms presented in this research, in a quick and easy way. GEMMA consist on a set of tools designed and developed under geodetic principles and aimed to simplify navigation research process. Those tools have been proven useful not only for the development of the work presented herein but also in a number of projects and research works (see Appendix 1 for a complete list of beneficiaries of GEMMA).

4.1 GEMMA overview

The architecture (and components) of a framework willing to provide with a useful set of tools and procedures to facilitate the research tasks related to a specific set of disciplines may obviously vary depending on the actual experience of the people involved first in its inception and, later on, in its design and implementation stages. From the author's point of view the components included in a positioning and navigation research framework should be able to offer, at least, the following set of features:

- Generation (densification) of realistic trajectories for any kind of platform: from spacecrafts, to aerial, terrestrial or marine vehicles, or even alive organisms;
- simulation of signals (measurements) for several types of sensors, as, for instance, IMU sensors or GNSS receivers,
- sequencing of sensor measurements,
- trajectory estimation and
- (trajectory) quality assessment.

Each of the features (and related tools) in the list above is important by itself. For instance, the ability to generate and densify trajectories avoids having to obtain such trajectories by other methods; when signal (measurement) generators are available, there is no need to organize data collection campaigns. In short, these tools help to save time and reduce costs, thus facilitating the work of researchers. However, it is the combination of different subsets of these features (tools) what reveals the versatility of the concept, and how it responds to different research use cases (see section 4.2).

Last but not least, data play a central role in such a framework. Impedances derived from the use of different data formats or even data models must be avoided to facilitate a correct flow of information among the tools available in the framework. In short, the less file format converters are used, the better; obviously, converters will be needed to import data collected by other systems—or conversely, to export results to these.

The natural solution to this was to devise, implement and use a common data model (and related interfaces).

Chapter 4 was presented as a conference paper at XXIII ISPRS Congress. *Navarro,J.A., Parés,M.E., Colomina,I. "GEMMA: a generic, extensible and modular multi-sensor navigation analysis system" , ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume III-3, pp. 433-440. XXIII ISPRS Congress, 12-19 July 2016, Prague (Czech Republic), July 2016.*

This data model, ASTROLABE [61], supports the generic and extensible architecture already implemented in the aforementioned tools. A full description of this data model can be found in Appendix 2.

GEMMA implements these features, providing a tool for each feature. In particular, it includes four signal generators, for IMUs, magnetometers, odometers and GNSS receivers.

Figure 4.1 depicts all the features just discussed (implemented as a set of tools in the GEMMA system). Additionally, *all* the possible data flows between these are shown. Depending on the actual scenario being tackled (see section 4.2 for some examples), only a subset of these tools and data flows will be used. For instance, when not working in a real-time environment, the trajectory estimation tool will not be connected to an acquisition system but will use either logged (pre-recorded) data (measurements) coming from one or more of the available signal generators.

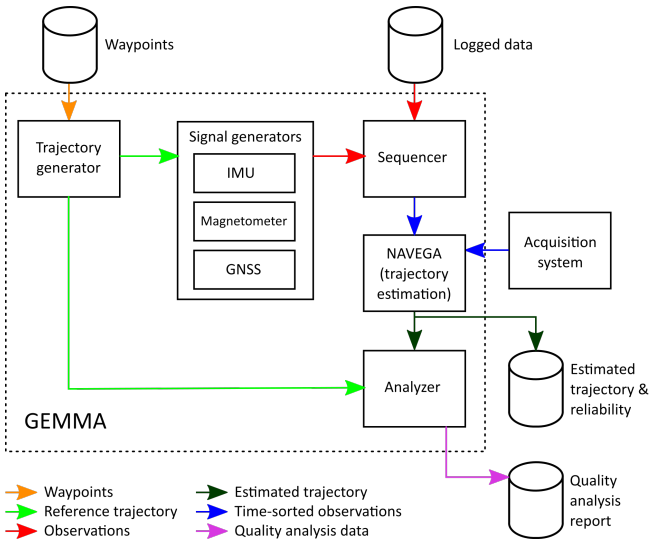

Figure 4.1: GEMMA: components and work flow.

The following sections describe briefly the different components of the architecture and show how GEMMA implemented these.

4.1.1 Trajectory generation

The ultimate goal of the trajectory generator tool is to create *reference trajectories* to compare with when these do not exist or it is too expensive to obtain them—for instance, flying a real plane. Strictly speaking, this tool is not about full trajectory *generation* but about trajectory *densification*. Given a set of waypoints (time-tagged positions, and eventually attitudes) an infinite number of trajectories can be derived. The tool should be able to produce a dense *realistic* trajectory including not only the previous way points but a much more extensive list of them as well as related velocities and orientations.

For these trajectories to be as realistic as possible they should be continuous and infinitely derivable. Moreover, since not all the vehicles follow the same dynamic equations, the tool should be designed in such a way that the densification of the way points and the related orientation definition follow the user required platform dynamics. This translates into an extensible piece of software where the densification is independent of the selected vehicle dynamics.

For versatility (and flexibility) reasons, the generator must be able to accept way points generated by different sources, as for instance, KML files or trajectories output by (realistic) flight simulators—file format converters may be needed to do so. The parameters controlling the behaviour of the tool (as the output frequency) must be also configurable by the user.

GEMMA includes a trajectory generation tool complying with the functional requirements discussed above. Specifically, it achieves the infinitely derivable position and orientation by convolving with a C^∞ function and later on interpolating the resulting trajectory at the user selected rate; the realistic orientation for different platforms is obtained from the dynamics formulas of those vehicles [68, 73].

4.1.2 Signal generators

A (software) signal[1] generator simulates the observations that would have been delivered by a sensor if it would have actually been used in a real campaign.

Signal generators take as input a reference trajectory—however obtained—and use the *models* defining the behaviour of the sensor to be simulated to deliver the simulated observations. Note that these models must include the *errors* affecting the sensors, so output data mirrors this *defective* behaviour.

Instrumental errors behave as a composition of stochastic processes (as random walks, white noise or Gauss-Markov ones). Additionally, errors may be originated by different sources, beyond the instrument itself, the environment (temperature, magnetic fields, humidity among many others) or the platform the instrument is mounted on (vehicles, human or animal carriers). These kinds and sources of errors must be taken into account by a signal generator so reality is properly mirrored.

Signal generators may be either software or hardware components and the presented architecture makes no distinctions between these. Software components, apparently, are less prone to obsolescence since it is possible to include new features just by maintaining the code.

There are different situations where the use of such generators prove to be useful. These are some, among others:

- To check how the quality of an estimated trajectory is improved by incorporating a sensor that is not available—but correctly modelled.
- To select the appropriate quality of a sensor according to the requisites that the estimated navigation solution must meet. For instance, a tactical-grade GNSS receiver may not be necessary in all situations.

At the moment of writing this paper, GEMMA includes four types of signal generators fully compliant with the functional description above, namely odometers, magnetometers, GNSS receivers and IMUs (see a detailed description of this one in chapter 5.) Others may be included in the future when needed. A hardware GNSS simulator, the IFEN NavX-NCS PROFESSIONAL GNSS simulator [39], is also part of the system.

4.1.3 Sequencer

The goal of the sequencer is to sort a series of heterogeneous observations in ascending time order. In this context, *heterogeneous* stands for *originating in different kinds of sensors*. The need to sort input data is motivated by the fact that the estimation of the successive values of the states integrating a trajectory is based on the use of observations that are *close* in time.

A sequencer takes as input a series of files or TCP / IP socket connections containing or transmitting time-tagged observations measured by different sensors and outputs a single stream of data including these same observations conveniently sorted. This output stream may be either written to a file (for local, batch processing) or sent through a TCP / IP socket connection (remote, real-time processing).

When processing data received through TCP / IP socket connections, the signal sequencer must be able to discard delayed data. A time tolerance parameter to decide if such data is accepted in spite of being too old is a very convenient feature to have; normally, reading data through network connections imply real-time processing and, in this situation, some delays or shifts related to time stamps are to be expected. Therefore, such a tolerance parameter may be used to alleviate these discordances. Obviously, the trajectory estimation tool (see section *4.1.4*) must be able to cope with such non-perfectly time-sorted stream of data for this feature be of any use at all.

GEMMA includes a sequencer tool implementing almost all the features described above. The sequencer's ability to send its output through a network opens the way to *distributed research*, which has been put into practice in several projects, as ATENEA [22]: observation data is obtained in one place, sequenced there, and sent via sockets to a second place, where it is processed.

[1] The words *signal*, *measurement* and *observation* are incorrectly used as synonyms here for the sake of simplicity. For instance, in some situations, additional processing may be needed to obtain a measurement out of a signal, but such distinction is not made in this paper.

4.1.4 Trajectory estimation

The goal of this tool is to estimate trajectories (navigation solution) from of a series of time-sorted, heterogeneous sensor observations. A detailed description of this tool is presented at chapter 5

4.1.5 Analyzer

The last tool of the system definition is the trajectory analyser. As its name indicates, the aim of the tool is to perform several quality analysis over an estimated trajectory. The main features it should include are:

- Given a reference trajectory, which is considered as the absolute truth, the system should be able to "translate" one trajectory to the other one and to compare them. By "translation" we mean to apply user requested offset and boresight matrix. The analyser should be able to compute statistics (like, mean, standard deviation and RMSE) fully describing the difference between these, which is the cornerstone to assess the quality of the estimated trajectory.
- Given a reference trajectory, the system should be able not only to compare estimated and reference trajectory as previously stated but also to check its coherence with the predicted error, and to determine if those last values are under or overestimated.
- Given the residuals or the innovations of an estimated trajectory the system should be able to determine the stochastic process beneath them, and to check if it is coherent with the expected one (typically white noise).

GEMMA implement the aforementioned features and additionally, a series of plots depicting some of the statistics is produced to help to better understand the results produced by the analyser.

Figure 4.2 is an example of one of those plots: In it the reader can see the actual error of an estimated trajectory (comparison between estimated and reference trajectory - in red) and the estimator predicted error standard deviation (in green). From the figure it can be directly seen that the estimated error does not correspond with the actual error, thus the trajectory determination system is being optimistic.

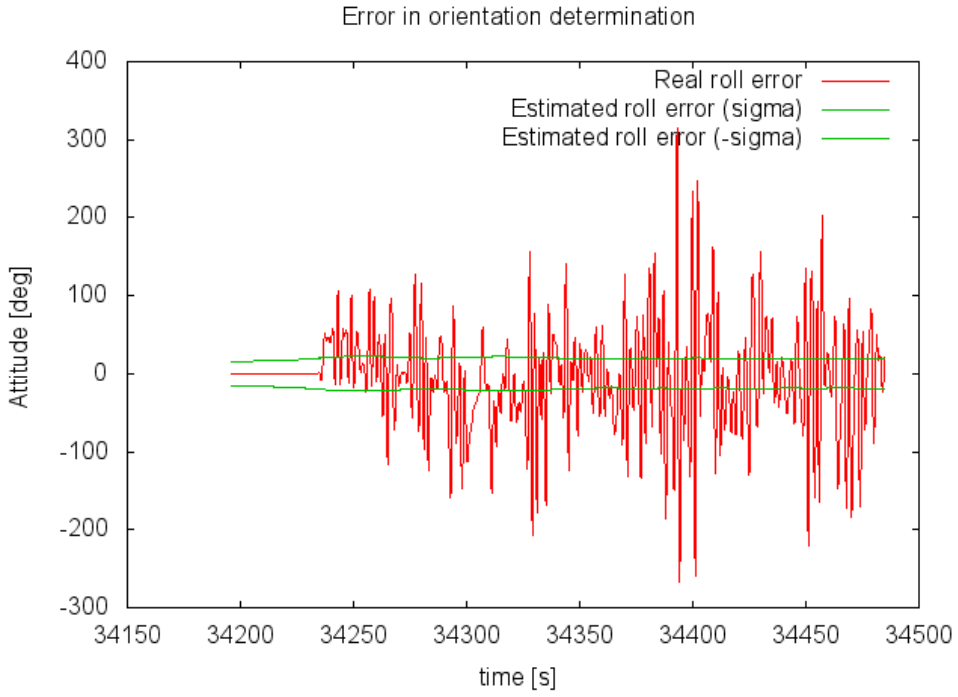

Figure 4.2: GEMMA's trajectory analyser: Example of quality plot.

4.2 GEMMA use cases

A framework such the one presented here may be used, and has already been used (see Appendix 1) in a variety of research use cases, just combining in different ways the components that integrate it. The most usual use cases are:

- Technology selection or validation.
- Algorithm verification and validation.
- Navigation and positioning in real-life environments.

The following sections describe these use cases and the role that play the different components of the tool in each of these situations. All the presented use cases have been successfully addressed with GEMMA.

4.2.1 Use case: technology selection or validation

This use case takes into account two possible situations:

- It is necessary to select a sensor whose quality is good enough—but not necessarily better—to estimate output trajectories that must match some precision and accuracy requirements. For instance, not all applications need a navigation-grade IMU, but may be costly enough to determine what is the actual quality level that will guarantee the required output.
- Assessment of the quality of a single sensor, that is, determining whether it is appropriate for the requisites affecting the estimated trajectory.

The advantage of using GEMMA in this case is that it is possible to *simulate* the signals produced by these sensors and then estimate and evaluate, for each candidate, the respective output trajectories. Needless to say, proper modelling of the behaviour of the intervening sensors is required.

The components of GEMMA involved in this use case are shown in Figure 4.3. In this figure, those GEMMA components that do not intervene in the use case are shown in pale grey.

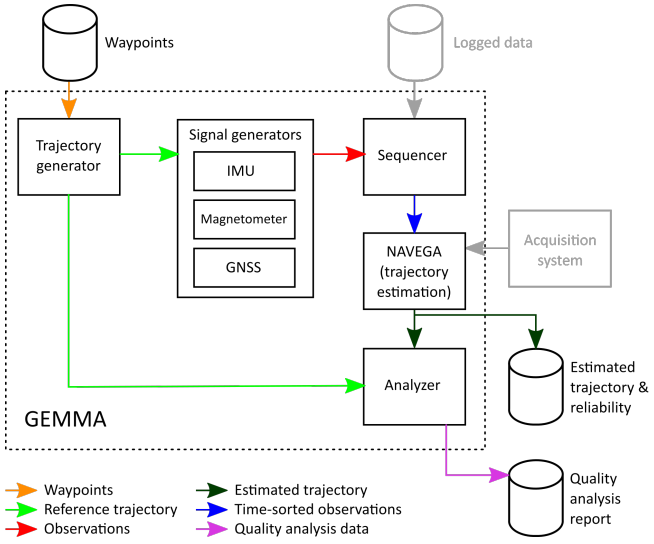

Figure 4.3: GEMMA use case: the technology selection.

The following would be the typical work flow to follow:

- First, a reference trajectory is needed to compare it with the output estimated ones. It may be either synthetically generated by means of the trajectory generator component or taken from any field campaign.
- The signal generator corresponding to the sensor(s) to evaluate would then be used to simulate the signals (measurements, observations) that would have been produced if the sensor would have actually travelled along the reference trajectory.
- Then the sequencer would time-sort all the observations produced by the signal generators.
- The trajectory estimator component (NAVEGA) would be used to estimate the output trajectory, and, finally,
- the quality of the aforementioned trajectory would be assessed by the covariance analysis provided by the analyser component.

When comparing several sensor candidates to select the most appropriate one, the process above should be repeated for every sensor being evaluated. If the goal is just to determine whether a sensor is good enough to fulfil the stated goal (trajectory quality) then a single run should suffice.

A real project where this use case was put to the test was GINSEC [60].

4.2.2 Use case: algorithm verification and validation

It has been stated several times in this chapter that GEMMA is generic and extensible. This means that new sensors may be included into the system—or those that have been modified easily adapted— at almost no cost.

Including a new sensor means writing the mathematical equations modelling its behaviour and generating a loadable library including the new model. There is no need to change already existing software to make the new model available to GEMMA; that is, when adding new sensors there is no need to maintain, change or adapt the existing code base. The new model will be loaded dynamically upon request—that is, when data related to such model is processed.

As it happens to any kind of algorithm, it is necessary to verify and validate new models in order to guarantee their correctness (that is, the new code contains no bugs) and performance (the new sensor is correctly modelled). This is the algorithm verification and validation use case.

Figure 4.4 show the components in the GEMMA system playing a role in this situation (note that, in this figure, those components not intervening in the process are shown in pale grey).

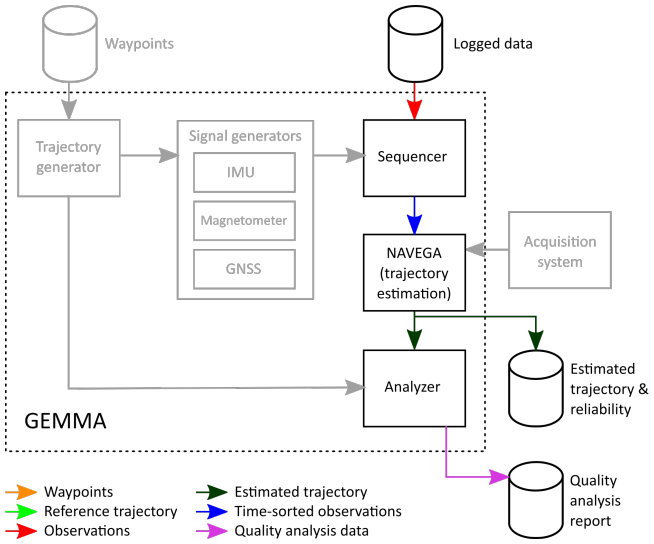

Figure 4.4: GEMMA use case: algorithm verification and validation.

The sequence of steps to take when verifying and validating a new algorithm is defined below. A reliable reference trajectory, however, must be generated beforehand. To create such trajectory, a data collection campaign involving not only a set of *trusted sensors*—that is, already correctly modelled—but also the new one must take place.

The reference trajectory must be estimated using only trusted sensors, so it is a reliable one to compare with (note that Figure 4.5 describes this procedure).Data coming from the new sensor *is just logged*, and will be later used to verify and validate the new model being developed.

Once the reference trajectory and the new sensor data are available, the procedure to follow is:

1. Using the sequencer tool, data coming from *all* the sensors involved in the data collecton campaign must be time-sorted.
2. The sensor mathematical models must be written. This, essentially, means filling the gaps in a template provided in GEMMA's developer's kit; such template adheres to the API (*A*pplication *P*rogramming *I*nterface) defined by the system. In this way, developers need to have no prior knowledge about the internals of GEMMA; their knowledge may be just limited to sensor modelling
3. A loadable library is then created and included in the GEMMA system.
4. NAVEGA is used to estimate a new trajectory.
5. The trajectory analyser is then used to compare the reference trajectory with the newly estimated one, and the results is also checked with the estimated covariance thus assessing its quality.

Steps 2 to 5 may have to be repeated several times until the quality of the mathematical model is satisfactory.

Several already concluded projects were clear examples of this use case: GAL [84], ENCORE [16] and ATENEA [22].

4.2.3 Use case: navigation and positioning in real-life environments

This is the use case where less components of GEMMA are involved: NAVEGA is used as a standalone tool (see Figure 4.5). In this situation, NAVEGA is used as a real-time trajectory estimator.

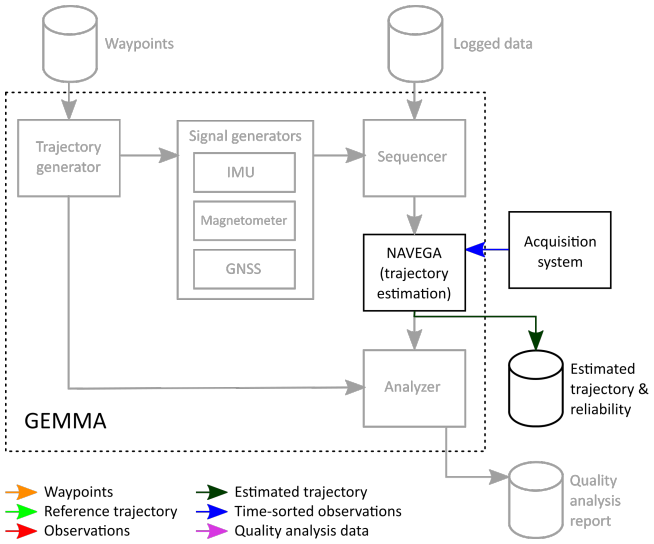

Figure 4.5: GEMMA use case: real-time environments.

Loaded with the appropriate set of models (those characterizing the sensors integrating an external acquisition system) NAVEGA might be used as a real-time server providing successive position and attitude values to some other subsystem(s). For instance, an autopilot system might decide to delegate the task of positioning the aircraft to NAVEGA, feeding this component with data coming from the different sensors on board, instead of performing this process itself [60].

A variation of the example above would compute position and attitude in real-time but log these data to permanent storage instead. This situation is typical in post-processing environments, when higher accuracy and precision are required. Processing the data collected with NAVEGA by means of techniques as PPP (*P*oint *P*recise *P*ositioning) would improve the quality of the estimated trajectory to the level required by professional applications (as network bundle adjustment, for instance) [56].

Chapter 5

Problem abstraction - Geodetic tools for trajectory determination and sensor simulation

This chapter delves into two of the most relevant tools of GEMMA: NAVEGA and the IMU simulator. NAVEGA is the central engine of the GEMMA system. In this chapter the reader will find the architecture of two innovative generic, modular and extensible tools (a trajectory determination tool and an IMU signal simulator tool) based on architectural principles successfully applied by the photogrammetric community [20], [12], [15].

5.1 Trajectory determination tool - NAVEGA

The geodetic principles for SW development, where NAVEGA principles are inspired, rely on the abstraction of the algorithms in a way that including new elements to the system only requires the development of a toolbox. The architecture proposed hereafter describe a SW platform for the optimal determination of trajectories or paths of stochastic dynamical systems driven by observations –or measurements– and their associated dynamic or static models. The reader will find firstly the requirements of the future Bayesian-based navigation systems, and a brief snapshot of the navigation algorithm structure. Later on, we present the principles that, from our point of view, should rule the next generation of navigation system architectures. Finally, we present a particular implementation of these architectural principles together with a short summary of its performance.

5.1.1 Next generation trajectory determination systems requirements

Although the main requirements (quality and robustness) of navigation systems have not varied significantly since the appearance of the first tools, the technology available in the market having to comply with these requirements has. On the other hand, the availability of lower-cost micro-processor and the miniaturization of technology introduce new requirements to the system (performance, integrability and extensibility.) Finally, the popularization of location-based applications has also had a large impact on navigation requirements evolution (usability):

- **Quality** (precision/accuracy). These terms refer to the quality of the solution. Back in the early times of navigation systems, high quality solutions were expected – in direct correspondence to the high quality of the measurement systems used by those. Due to the irruption of COTS (Commercial Off-The-Shelf) systems in recent years, the quality of the solutions must rely more on algorithms and less in the characteristics of sensors. Thus, more and more sophisticated inference algorithms are needed to obtain similar results with cheaper sensors.
- **Robustness**. This requirement is about the system resilience to the presence of outliers, the lack of information or bad initialization. New users or non-specialized on the field expect for solutions valid in any environment, and will not be satisfied with system that behave incorrectly due to issues that they are not even aware of, such as multipath, magnetic fields or satellite occlusions. New systems should consider all this scenarios and their related problematics.
- **Performance**. The owners of handheld devices demand navigation systems able to position them in the most singular places and always in real time. This implies to use a wide range of sensors to be able to provide a

Section 5.1 was presented as a conference paper at ION GNSS+ 2015. *Parés, M.E.,Colomina, I., On software Architecture Concepts for a Unified, Generic and Extensible Trajectory Determination System , in Proceedings of the ION GNSS+, 08-12 September 2015, Tampa, Florida (USA).*

continuous solution but also algorithms with a low computational cost able to deliver results in short time. Even when dealing with the most sophisticated sensors, users expect to have a solution, computed by a microprocessor, in real time.

- **Integrability**. New users also expect that location-based applications have access to location service providers. The new navigation systems should be easily accessible from external users in order to be easily integrated in non-geo applications. This translates in clear and manageable interfaces.
- **Extensibility**. The last requirement, related to the continuous irruption of new technologies, refers to the capability of the system to integrate new sensors without increasing exponentially the development effort. Either for professional applications or for mass-market ones, more and/or new sensors are used every year. New navigation systems should be able to cope with that, focusing only on sensor modelling and not in the estimation algorithms.
- **Usability**. The users of navigation system have evolved from airplane or boat crews and photogrammetry specialists to millions of citizens requesting for geo-located services in their daily routines. These new users have never received (and will never receive) training on navigation techniques, so the interfaces of navigation systems should be clear and intuitive. For professional users and researchers, interfaces should allow the modification of states controlling the behavior of the software as well as a deep interaction with the application. To sum up, the system should be able to deal with a wide range of users and, at the same time, guarantee the quality of the solution and its performance.

5.1.2 Next generation systems architecture principles

In order to achieve the current requirements of extensibility and modularity we have revisited the previous algorithm, which has gone through a process of abstraction and generalization. As stated in [15], simple and extensible software design requires correct abstraction levels. Abstraction is the process of expressing a quality or characteristic apart from any specific object or instance. Thus, we have identified the navigation software characteristics apart from its actual implementations. According to [15] *insufficient or needless abstraction leads to complex systems, wrong abstraction leads to non extensible systems but correct abstract models are, therefore, the key to simple and extensible systems.* Because of this abstraction exercise on navigation systems, we have arrived to some principles that will drive the definition of our new architecture:

- **Separation between estimation and modelling**. Bayesian estimation processes ingredients are data (measurements/states), the relations between data and the specific algorithm used to estimate the states [81]. Current navigation systems implementations do not take into account this process conceptual segmentation, leading to very efficient but hardly reusable SW tools. We propose to define an architecture where **estimation** algorithms (procedure) and **modelling** (data and its relation) are implemented as separated components. The separation of the "numerical cruncher" and the equations allow a quick extension of the software when new sensors appear. Roughly speaking, "configuring" the system implies "loading" components related to the specific navigation sensors and setting a number of mission related states. Still roughly speaking, "configuring" the system for a specific instrument has no effect on the computational kernel even if new sensors come into play. The actual implementation of this concept could take benefit from dynamic libraries. While the common elements of the state-space approach should be implemented directly in the "number crunching" kernel, the uncommon ones should be materialized in dynamic libraries. An implementation like this one, has a lot of benefits such as minimizing the required amount of RAM in run time, since only the needed sensors are loaded; minimizing the development time, since the inclusion of new sensors imply the development of relatively small fragments of code and, finally, minimizing the time needed to train developers, since these should only have a deep knowledge about the sensor but not about Bayesian filters. Furthermore quality performance and robustness will not only not decrease but can even increase since more efforts can be placed on the development of the "number crunching" kernel.
- **Sensor/measurements abstract reference model**. Focusing on the information related to the navigation system, we distinguish four elements: (input) measurements, (input) auxiliary instrument constant values, (output) states and (equations) models relating all those elements. Available technology allows us to have several sensors providing different kind of observations which purpose is to estimate same states type (e.g. magnetometers and star trackers both aims to determine orientation in space). Technology evolution, provide users with same principle sensors but a wide range of quality performances, the purpose of all this sensors, however, is to estimate same states type (e.g. navigation-grade to MEMS inertial sensors for position and orientation estimation). The modelling of the information component in four categories (observation/measurement, state, model and instrument) avoids code repetition and allows the use of object-oriented modelling, its inheritance, encapsulation and polymorphism mechanisms [12], [15]. A clear stateization of measurements, states, equations and auxiliary values elements is a powerful mechanism that allows the simplification of the system and consequently will help developers to implement quickly new sensor measurement and state models in the SW.

- **Computational strategy object**. Navigation solution performance depends not only on the equipment but also on the environmental conditions [30] [31]. Some years ago, this last factor was not taken into account in navigation systems, mainly focused on airborne environments. Nowadays, since the number of non-airborne users is growing exponentially, such systems should include a mechanism able to deal not only with sensors but also with scenarios, a context awareness *strategy object*. Users should be able to inform the system about its environmental context through an options file. In situations where the user is not able to provide with this kind of information, the strategy object should be capable to determine such context by itself (e.g. the qualitative analysis of the data provided by inertial sensors mounted in a car should allow the system to determine this environment by itself). Strategy should also be aware of the environment to be able to add to the system, if needed, contextual information (e.g. vertical velocity restrictions for terrestrial platforms or sudden direction changes restrictions in fixed-wing airplanes.) Strategy must also decide at any moment of the process if the system is able to provide the required solution out of the available information and, if it is not possible, decide how to proceed. If the object detects that the provided information is not enough to compute a solution, it must be able to choose between delivering a partial solution or warning the user that the system is not able to provide an acceptable one and that more information is required. Once again, the software must reflect the conceptual difference of this object and the estimation and modelling objects; thus, the strategy object should not be integrated in the "numerical cruncher", but implemented as a separate component. An approach like this will allow the use of the same sensor models for several platforms and environments without having to implement them repeatedly. This implementation have multiple benefits like minimizing the system RAM requirements and simplifying the development process, since developers can tune its strategies independently of the estimation process.

- **Generic/adaptable user interface**. The definition of the interface (be it file or network) of modern navigation systems shall be funded in the fact that the interface should not complicate the extensibility of the system. Therefore, the definition of such interfaces should take into account future extensions of the system. An incorrect interface definition leads to an under-utilization of the capabilities of the system and to an increase on development efforts. Currently, when integrating new sensors into a navigation system, not only the source code must be modified but also the user interface definition and layer. As the eyes are the mirror of the soul, the interface should be the mirror of the system. For this reason, we propose that the definition of the interface take into account all the abstraction processes presented in previous premises. Analogously to what has been discussed until this moment, we will separate the user interface –control of the estimation process— and the data interface –generic description of the rules to provide and read data from the system. [12] — Nowadays, trajectory SW can be used by such different users like geophysics, photogrammetrists, wedding recorders or pedestrian shoppers. We can classify trajectory determination SW users in three groups: lab, workshop, factory [59]. By lab users we refer to the ones that intend to develop new models for new sensors. They need to have access to all SW configuration states in order to find the best configuration to solve a specific problem. On the other side, a civilian user that intends to use the navigation system to go for a dinner in a restaurant is not worried about what are the sensors used by the navigation system or how these have to be calibrated. Between both users we can find the workshop ones, typically professional navigation system users that want to get the best available performance of the system. A successful SW should be useful for all of them. It must be able to deal with several dialects. This translates into the fact that system should be completely configurable (for lab users) but should allow external interfaces with restricted configuration access (for factory users). Concerning data interface, it should reflect the abstraction exercise performed in previous sections and its foundations should be the sensor, measurement, state and model concepts. With a definition like that, the long-term development costs will be reduced since the interface, once defined, will be suitable to incorporate any new sensor.

Table 5.1 summarize the contribution of each principle to the fulfilment of the system requirements, through the traceability matrix confronting requirements and principles. Note that the performance requirement is not specifically considered by any principle. Since we are proposing an object-oriented implementation it could be stated that performance will suffer a heavy with respect to the use of sequential programming approaches. This is true but it can be demonstrated that the penalty to pay for the use of an object-oriented approach is almost negligible. Furthermore, the strength of new computer processors makes this overhead less significant when compared to the overall system performance.

5.1.3 Proposed architecture definition

Based on the design principles presented in the above section hereafter we propose an architecture that complies with that design.

	Estimation vs modelling	Sensors abstract modelling	Adaptable user interface	Strategy object
Precision/accuracy	X	X		X
Robustness	X			X
Performance				
Integrability	X	X	X	
Extensibility	X	X	X	X
Usability		X	X	

Table 5.1: Trajectory determination tool requirements vs architecture principles traceability matrix.

System components and activity diagram

Our system is based on four components (Figure 5.1): computation kernel (or "numerical cruncher"), trajectory (this component is the one includes modelling and supervisor), interfaces and a driver to control all the process. These components should be implemented using both static and dynamic libraries -when operative system allows it. Static libraries should be used to implement the procedure ("number crunching" kernel) while the modelling/strategy should be resort to dynamic libraries - thus, only the classes related to the requested sensors will be loaded in run time. To extend the system, incorporating new sensors, is as simple as writing new classes – that should be included in new dynamic libraries - adhering to the API defined by the system.

The system inputs are not only sensor observations and generic context, as in common trajectory determina-

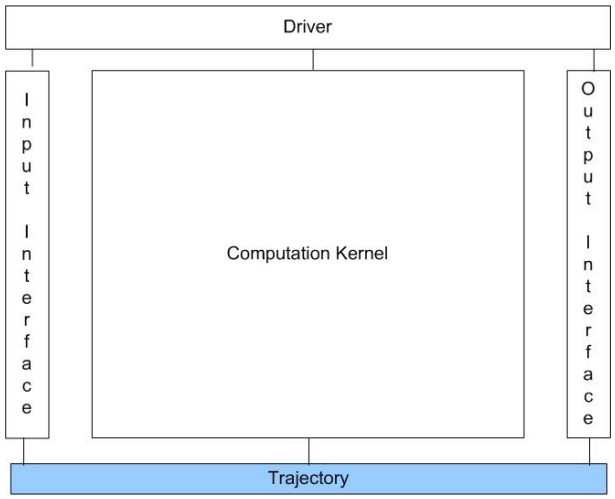

Figure 5.1: Trajectory determination SW architecture components diagram.

tion tools but also a list of requested states/states, a list of desired models/equations to compute these states and a specific computation context or strategy. All this information should be collected through the interface layer. The control component is in charge of loading all the requested objects (the aforementioned dynamic libraries.) The strategy object should analyse the feasibility of computing the target states from the set of given observations. If the computation is possible all the data should be sent to the estimation subcomponent to process it; otherwise a warning message should be sent to the user. There will be situations where only a subset of the desired states can be computed; should this happen, the strategy object should decide whether to compute what is possible to compute or to send a warning message. The output of the estimation subcomponent is delivered to the user through the interface component.

It is important to note here that the process is valid either for forward, backward and smoothing processes. Thus, a few (and simple) modification of both (1) the (unique) source code implementing this processing algorithm (the State-Space Approach) and (2) the interface component, make possible to use the same algorithm for either real-time or post-processing work modes.

Processing component

The processing component includes not only the driver subcomponent but also the two filtering steps subcomponents: the prediction component (e.g. stochastic dynamical systems solver for Kalman filter family estimators) and the update component (e.g. correction step for Kalman filter family estimators). The processing component is responsible for loading computation options and start the desired trajectory estimation processes (forward, backward and/or smoothing.) These processes mainly rely on the two numerical estimation subcomponents, prediction and correction.

The API must allow the inclusion of new numerical methods that may be of interest to the final user (extensibility criterion). In this way, both the prediction and the update components should be able to incorporate any related numerical method implementation. Through the interface layer, the user should be able to select among the available methods. Furthermore, it is important to remind that, as stated in previous sections, each of these implementation should be independent of the actual set of equations to solve; these should be designed to interface efficiently with the modelling objects (observations, equations, states and constant values) to retrieve values, to allocate and properly fill in vectors and matrices that will allow state estimation.

Modelling component

We propose a sensor/measurements model like the one presented in [15], where the fundamental reference model is based on four abstract categories (i.e. classes): instruments, observations, states and mathematical models. It is important to remark here that when we talk about observations and states we refer not to a unidimensional observation/state item but a multidimensional one. Every new kind of sensor to be integrated in the system will probably collect measurements not available for other sensors. These measurements will be related to some unknown states, probably position or orientation. The relation between the observations and the states, maybe with the participation of the instrument auxiliary constants, is materialized by the mathematical models (stochastic equations). In order to extend the system, is necessary to identify the instrument and its characteristic values, to list its related measurements, to identify the states and the mathematical models that relate them. Once this is done, specific classes that inherit from the fundamental reference model can be reused or newly developed.

EXAMPLE – *Magnetometer modelling*

- **Instrument**. The magnetic declination should be taken into account to correct the measurements. This value can be entered as a constant to the system through the instrument object. Also the boresight matrix between magnetometer and trajectory reference frames.
- **Observation**. Magnetic field intensity values.
- **States**. Attitude parameters at a rotation matrix.
- **Model**. Rotation matrix that relates magnetometer and trajectory orientation. Note that the model should take into account that trajectory and magnetometer reference frames could not be the same.

Strategy component

When designing the strategy component three issues arise: context awareness, context conditioning and computation feasibility.

Context awareness refers to the capability of the system to be aware of the environment and to add constrains to the system if required. The context awareness can be provided by the user or automatically detected by the SW itself. In the second case, a qualitative analysis of available data should be performed. Thus, for example if an IMU is involved in the computation and the signal output by the IMU matches (in a qualitative sense) the typical movement patterns related to the behaviour of human beings [55], a pedestrian environment can be assumed. This context awareness analysis should be done periodically to assure correct environmental assumptions during the processing of the whole trajectory.

Context conditioning refers to the capability of the system to include conditions to the system that depend on the environment. In our approach, instead of implementing condition equations, we add observation equations to the system. This reduces computation complexity and allows reusing all the implemented estimation algorithm without any additional changes. In order to do this, condition equations should be transformed to observation equations by assuming a reduced noise instead of a "perfect" condition. The smaller the noise, the most restrictive the condition will be. For each environment, a set of additional condition equations are defined and added to the intrinsic sensor observation equations.

Last issue refers to *computation feasibility*. In order to decide if the system is able to provide a solution out of the available information we should check the degree of freedom of the problem and the update observations' matrix. If not enough observations are provided, the strategy object checks the suitability of performing a reduced adjustment with (pre-selected) fundamental states. If this is not possible, computation is finished and control is returned to the user. On the other hand, it could happen that there are enough measurements but the system is not robust enough to converge properly. This can be detected by means of covariance matrix analysis. Each strategy object, should include a set of relevant and non relevant states, with covariance thresholds to determine whether those states can be estimated or not.

The strategy component should be implemented, if possible, in classes stored in dynamic libraries. These classes are closely related to the sensors involved in a particular computation. All strategy specific implementation classes will be derived (inherit) from the same class, so all of them should have the interface. This component is aware of several strategy issues, highlighting context awareness determination, context conditioning and computation feasibility. In this way, several strategies (strategy classes) may be implemented and be able to deal, in different ways, with the same kind of data (e.g. IMU and GNSS), taking into account issues like the quality of the sensors, its performance or the environment.

EXAMPLE – *Strategy for IMU/GNSS/MAGN*

- **Context awareness**. IMU signal qualitative analysis. Fit with predefined patterns to deduce, pedestrian, terrestrial vehicle or fixed-wing airplane environment.
- **Context conditioning**. For pedestrian navigation include observation equations to limit maximum velocity. For terrestrial vehicles include observation equations to limit vertical and lateral velocities. For fixed-wing airplane include observation equations to avoid backward movement
- **Computation feasibility**. Initial state estimations will be derived from GNSS (position and velocity), and magnetometer (orientation). IMU calibration values will be derived form a combination of magnetometer data and IMU data. Position, velocity and attitude are the fundamental states; the system will not try to solve IMU calibration values until the standard deviations of the first ones are below predefined thresholds.

Interface component

The interface component based on three clearly distinguishable concepts: the Application Program Interface, the Data Interface and the User/Options Interface.

Application Program Interface design is totally driven by the system extensibility requirement. This translates into object oriented principles, inheritance capability and the use of dynamic libraries for the modelling and strategy components. Since the modelling and strategy component classes inherit from five basic classes (instrument, observation, state, model and strategy) the interfaces are the ones defined by these classes. New classes must completely adhere to this API. Regarding the platform interface with other systems, its design is as simple as possible. The tool should executed just with a "go" command and loading a configuration options file (explained at the end of this subsection).

Since the design principle of the Data Interface is based on is adaptability - to incorporate easily and painlessly new instruments, measurements, models or states - we have defined an interface (equivalent for network TCP/IP sockets and file interfaces) based on the generic description of observations and models (see Appendix 2). Each observation is described through a time tag, its value (expectation), its estimated error (standard deviation) and some auxiliary values that can be useful for computation purposes. When using a file interface – based on the XML standard - the way to introduce these values in the SW is through the $< l >$ tag shown below:

$$< l > obs_{code}\ obs_{id}\ time\ a_1\ \ldots\ a_n\ e_1\ \ldots\ e_n\ s_1\ \ldots\ s_n < /l >$$

where obs_{code} refers to the unique identifier for a specific observation type (for example, GNSS position) and obs_{id} refers to the sensor identifier. time refers to the acquired data time tag, a_i refers to auxiliary values, e_i refer to the acquired set of values and s_i refer to its standard deviation.

The observation equation to be used relates observations and states. Again, when using the XML file interface, the way to introduce these values in the SW is through a the $< o >$ tag:

$$< o > model_{code}\ time\ par_1\ ...\ par_n\ obs_1\ \dots\ obs_n < /o >$$

where $model_{code}$ refers to the unique identifier of the model class that relates a certain set of state types with observation types. Time refers to the time to perform the calculus, par_i refers to the identifiers of the states to solve (the type of this state is known by the model class) and obs_i refers to the identifiers of the observations to involve (again, the type of this observations is known by the model class). This data interface is common (uniform) for the two different data channels implemented by the system: (disk) files or TCP/IP sockets. This allows for remote (in the cloud) execution, being possible to capture data in whatever the place providing that a network connection is available. The advantages of this approach are manifold: perhaps the most important one is ubiquity (data may be captured everywhere); another advantage to consider is the reduction of the computational power needed by the processor in charge of computing the navigation solution, since it will not be responsible of managing the sensors acquiring data. For extended details on the interface definition, see Appendix 1 of this thesis report.

The user/options interface is designed as a two-layer interface. The lower one is based on files and interacts directly with the platform component. On top of this one, a user oriented interface (generally a command line or a graphical user interface) is developed. This second interface allows the SW dealer to modify determined configuration states depending on the user needs. Thus, for example, in a lab environment, the GUI should allow the configuration of all the computation states, while, in a factory environment, these options should be restricted to the minimum. In this way the software can be used in lab/research/factory environments by just changing the most external layer, the GUI.

Architecture strengths and weakness

Since the proposed architecture fully relies on the principles presented in previous sections, we can state that it satisfies the requirements of the new generation of navigation systems concerning quality assessment, robustness, performance, extensibility, integrability and usability. In the next section we present an implementation of this architecture where we have been able to demonstrate empirically that these requirements are fulfilled.

As any other approach, the one proposed in this thesis has some drawbacks. In this case, at the implementation level. For example, since we want to accommodate an undetermined number of sensors in each execution we have to take into account that the amount of memory used will change from execution to execution, leading to a high demand of RAM memory. Moreover, we can not fine-tune the Kalman filter to improve matrix management, since we do not know their dimensions a priori. Thus, special attention on the computational burden of the algorithms implemented must be paid.

5.1.4 A sample implementation

We have implemented the aforementioned architecture in a SW system called NAVEGA. It is coded in C++ and provides real time and post-processing trajectory estimation under Windows and Linux operating system running in different processor architectures like x86, LEON3 and ARM. Currently the system includes a family of Kalman filter algorithms and provides with Gaussian states, this is the output are expected values and covariance matrices. Furthermore, it also provides the user with residuals expected and standard deviation values, suitable for a posteriori trajectory analysis.

Until this moment, NAVEGA has been proved to be **extensible** since it has already been used to process data provided by several types of *sensors* like IMUS [56], GNSS [83], redundant IMUS [56], LIDAR [57], camera images [4] and odometers [3]. Since not all the implemented sensor models are available in commercial systems we validate NAVEGA solutions using indirect methods. Using NAVEGA as positioning provider to remote sensing platforms (like photogrammetry systems); we compare the NAVEGA georeferenced data with topographic measurements. Up to now, these comparisons have shown the success of NAVEGA, that is, the predicted state error estimation corresponds to actual state error, or, in other words, the expected quality is achieved. NAVEGA has also been successfully used for processing data collected in a wide range of *environments*: airplanes, helicopters [56] and terrestrial vehicles, either outdoors and indoors [3].

Regading **robustness** issues, the implemented analysis tools has been proven able to deal with odometers and camera outliers. When working with GNSS modelling, the system is able to detect outliers when these affect just one satellite. At this moment, more than one outlier cannot be properly detected.

NAVEGA has also been proven to be **integrable** since it has been used in several projects as the location provider of different services like a mobile mapping system [22] or a GNSS receiver with tight coupling capabilities [82].

With the previous examples we have been able to verify that NAVEGA's **performance** is suitable for a wide range of

platforms. NAVEGA is able to process INS/GNSS solutions at 5000 Hz in Real Time when running on an Odroid-XU3 with a negligible latency.

Last but not least, due to its file interface the system is completely **usable** either for Kalman filter experts to unexperienced users, through several GUIs (Graphical User Interface), adapted to theirs knowledge of the problem.

5.2 Inertial sensors simulation tool - IMU simulator

Since 2010, and thanks to the introduction into the market of chip-scale inertial sensors, inertial technology users have grown exponentially. There are situations where the performance of these new chip-scale IMUs is not suitable for the requirements of the target applications. The validation of new methods related to inertial technology could become really expensive. For example, to determine which is the suitable IMU grade for a specific purpose there are basically two options, to try all available IMUs (which could be extremely expensive), or to do simulations until the required IMU grade is determined. At any rate, the economical and time efforts needed to perform feasibility analysis may be reduced with the use of inertial simulators able to reproduce a variety of scenarios.

The main purpose of the work presented in this section was to implement a software tool capable of simulating the outputs of any IMU, in particular, from navigation grade to MEMS ones. In Figure 5.2 the reader can see a picture of the CTTC's IMUs. From left to right: iMAR FJI (navigation grade), IMU-IIe (navigation grade), Northon Grumman LN200 (tactical grade), ADIS 16488 (MEMS) and EPSON S4E5A0A0A1 (MEMS). These IMUs have been used to assess the degree of "realism" of the simulation tool (see section *5.2.4*.)

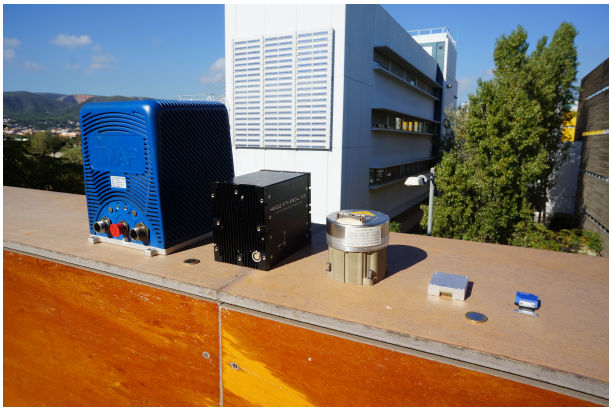

Figure 5.2: Image of a set of inertial systems: from navigation-grade to MEMS.

In the simulation process, not only the reference trajectory is relevant; the correct identification of all the involved error sources allows their proper combination to recreate a complete set of trustable scenarios. Given a specific reference trajectory, users may simulate the behaviour of as many combinations of sensors, platforms and environmental factors as needed.

5.2.1 System requirements

The first step to take when developing a software tool is to describe carefully the system requirements. This process is of the foremost importance to assure a useful and durable tool. Hereafter, a summary of the IMU simulator main requirements is presented:

- **Functionality**. Given a set of waypoints, the tool should be able to derive from them a trajectory composed of positions, velocities, linear accelerations, attitudes or orientations and angular velocities at the user desired frequency. Linear accelerations and angular velocities (IMU outputs) should include noise corresponding to user selected IMU model, environment and platform.
- **Quality**. The simulated data should be realistic. This means that the output signal should not only fit to the trajectory dynamics but also to the stochastic error signature of each simulated IMU.

Section 5.2 was presented as a conference paper at ISS Gyro Symposium 2015. *Parés, M.E., Navarro, J.A., Colomina, I. "On the generation of realistic simulated inertial measurements", in Proceedings of the ISS Gyro Symposium, September 2015, Karlsruhe (Germany)*

- **Performance**. For a tool like this no real time performance is requested. Even though, the tool should be useful for feasibility analysis replacing field tests campaigns, thus the input/output ratio should not be excessive.
- **Integrability**. One of the purposes of the tool is to avoid preliminary field tests when validating new algorithms and methodologies. This means the system should be able to provide inertial data with the appropriate interfaces.
- **Extensibility**. The last requirement, related to the continuous evolution of the technology, refers to the capability of the system to integrate new error models and environmental features with a moderate development effort. The simulation system should be able to cope with that.
- **Usability**. The users of inertial systems have evolved from navigation system developers to specialists in a wide range of fields like medicine, nursing, Internet of Things, among others. These new users have never received training on inertial stochastic modelling, so the interfaces of the navigator should be clear and intuitive. For professional users and researchers, interfaces should allow the modification of states controlling the behavior of the software as well as a deep interaction with the application. To sum up, the system should be able to deal with a wide range of users and, at the same time, guarantee the quality of the solution and its performance.

5.2.2 System architecture

The architecture of a software system is a description of the system which comprises components, the main properties of those components, and the relationships between them. Usually, in order to describe those relations, some standard models are used. Among others there are the UML Object Model (where the main components and their relationship are presented) and the UML Activity Model (where the different actions done by the system during an execution are presented.)

Object Model

The simulator is designed under the Object-Oriented (O-O) paradigm and is coded in the C++ language. The main reason for choosing the O-O paradigm instead of procedural programming is that allows the system to be modular and extensible, two basic characteristics for a system whose main purpose is to validate methods.

The tool includes three main components, one for trajectory densification, one for errorless data generation and a third one for noise data generation.

The design of this last component, the noise data generator, has undergone a rigorous data modelling process, which has led to the identification of the essential data entities involved in this process. Error data have been thus classified in platform (vehicle), instrument and environment error data, each of these showing their particular characteristics. Observational data have been classified in two separate entities: accelerometer and gyroscope data.

These entities define the common traits for all observation and error data variations. That is, capture the essence of such information. Any observation or error data source will adhere to these specifications. This abstraction makes the system general – that is, able to cope with any kind of observation or error data types.

Particular occurrences of observation or error data are just instances of these basic entities. The use of object orientation and C++ classes allows for the actual implementation of such instances, adhering, as stated above, to a common data frame. The use of shared libraries (also known as Dynamic Load Libraries or DLLs in the Microsoft Windows environments) to implement these different classes (objects) opens the door to extensibility: each time a new type of data – whatever its, type – is needed, a new shared library is created and included into the system. Dynamic loading guarantees this automatically.

Figure 5.3 presents the taxonomy of the different aforementioned data entities related to the noise data generator. It shows, as well, how one of these entities (gyroscope) may be implemented to cope with different, actual gyroscopes, by means of different C++ classes and shared (dynamically loadable) libraries.

Activity Model

In what concerns the data processing, the simulator reads the waypoints data sequentially and creates a dense trajectory data set. Once the dense trajectory data set is available (this is, time-tagged positions, velocities and attitudes) the simulator computes errorless linear accelerations and angular velocities taking into account the planet gravitational and rotational effects. The last step is to add noise to those signals.

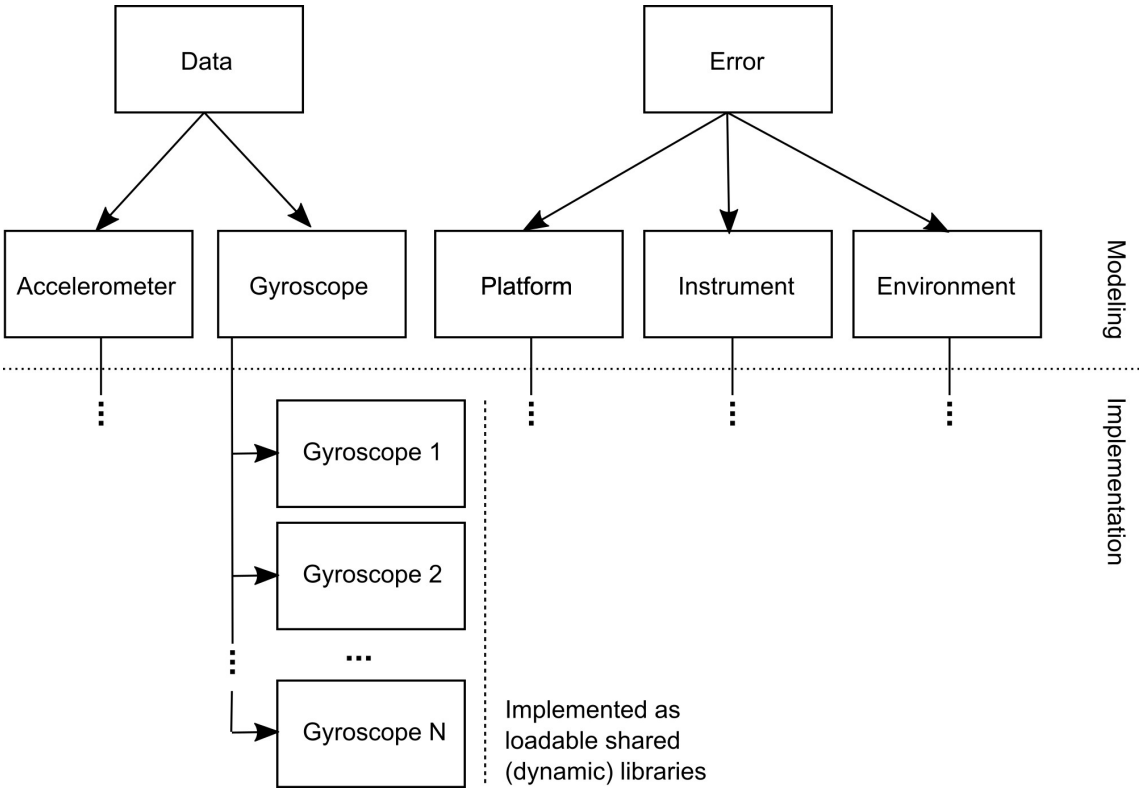

Figure 5.3: IMU_simulator: error components and subcomponents.

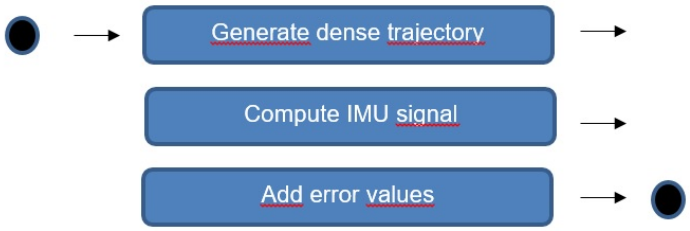

Figure 5.4: IMU_simulator: UML activity diagram.

Trajectory densification and errorless signal generation

Given a set of waypoints (time-tagged positions, and attitudes) an infinite number of trajectories can be derived. In order to generate a realistic trajectory, the simulator first densifies a trajectory and then computes angular velocities and linear accelerations. In order to generate proper linear accelerations a C^4 position function is required, that is, the function defined by positions, as well as its derivatives from first to fifth order, must be continuous functions. An initial set of waypoints is densified by interpolating positions until the desired output frequency is reached. Later on, in order to fulfill the requirements, the data set is convolved with a C^∞ function, producing a compliant reference position function. The generation of angular velocities relies on the same premises: a dense set of attitudes is first generated and later on convolved with a C^∞ function.

In order to simulate an errorless signal, the mechanization equations are rewritten in such a way that angular velocities and linear accelerations are isolated. In the next paragraphs the models used are described.

Angular velocities Given a trajectory, the simulator computes angular velocities for an orthogonal triad of angular rate sensors. The model implemented is

$$\Omega_{ib}^{b} = C_{n}^{b}(\dot{C_{b}^{n}} + \Omega_{in}^{n}) \tag{5.1}$$

where Ω_{ib}^{b} is a skew-simmetric matrix relating the body frame (b) with the inertial frame (i) obtained from the angular local velocities measured by the rate sensors; Ω_{in}^{n} is a skew-simmetric matrix relating the inertial frame (b) with the navigation frame (n) and obtained from the position given by the reference trajectory; and C_{n}^{b} is a rotation matrix relating the body frame (b) with the navigation frame (n) and obtained from the attitude values.

For details on equation 5.1, the reader is referred to any textbook on inertial navigation such as [14] or [40].

Note that the matrix $\dot{C_{n}^{b}}$ can be either computed numerically from C_{n}^{b} or analitically after numerically differentiating its components. After analysing the results for both methods, even though both are equally valid, the first one was selected because of its low computational burden.

Linear Accelerations.Given a trajectory, the simulator computes linear accelerations for an orthogonal triad of accelerometers. The model implemented is

$$f^{b} = C_{n}^{b}(\dot{v^{n}} + (2\Omega_{ie}^{n} + \Omega_{en}^{n})v^{n} - g^{n}(x^{n})) \tag{5.2}$$

where f^{b} is the vector of linear accelerations in the body frame (b); v^{n} is the vector of velocities of the sensor in the navigation frame (n); C_{n}^{b} is a rotation matrix relating the body frame (b) with the navigation frame (n); Ω_{ie}^{n} is a skew-simmetric matrix relating navigation frame (n), inertial frame (i) and terrestrial frame (e); Ω_{en}^{n} is a skew-simmetric matrix relating navigation frame (n) and terrestrial frame (e); and $g^{n}(x^{n})$ is the gravity vector expressed in the navigation frame (n).

Noise signal generation

There is a wide range of noise sources that interfere in sensors, for example environmental factors (temperature, radiation, magnetic fields [45]) platform inner vibrations [87] or, of course, the sensor electronics and internal mechanics [90]. The solution proposed in this dissertation models error sources as an entity on its own and implement these as separate, pluggable components. Platforms (vehicles) and environmental factors are treated in the same way. This concept leads to a modular, generic and extensible tool capable of generating realistic inertial measurements in a wide range of environments. This approach let users define and use their own models for noise sources, platforms and environmental factors easily.

Independently of the error source, the errors affecting the data could be additive (bias and white noise), multiplicative (scale factor and quantization) or distributive (misalignment). The proposed simulation tool implements the error according to the following formula:

$$\begin{aligned} d_o &= [((R((1+S)*(d_i+b)+w))/quant]*quant \text{ if } d_0 \in [-Range, Range] \\ d_o &= Range \text{ if } d_0 \geq Range \\ d_o &= -Range \text{ if } d_0 \leq -Range \end{aligned} \tag{5.3}$$

where S is the scale factor, b the bias, w the white noise, $quant$ the quantization value. $Range$ is the range where the IMU provide values, d_i is the ideal simulated signal and d_o is the output final signal.

Bias refers to the offset in the measurement provided by an inertial sensor. For bias effects, the model used is

$$b = b_i + b_p + b_e + b_d \tag{5.4}$$

where b_i, b_p, b_e and b_d are the instrumental, platform, environmental and dynamics biases respectively. The instrumental bias uses to be a combination of Gaussian noises [90] (IMU manufacturers provide with its stochastic properties) platform bias refers to the accelerations and angular velocities induced by the vibrations where the simulated IMU is to be mounted in; the environmental scale factor value is a temperature and pressure dependent value and the dynamic scale factor is a velocity (linear and angular) dependent value [95].

The **scale factor** of an inertial sensor is the relation between the output signal and the quantity which is measuring. For scale factor effects, the model used is

$$S = S_i + S_e + S_d \tag{5.5}$$

where S_i, S_e and S_d are the instrumental, platform, environmental and dynamics scale factors respectively. The instrumental scale factor uses to be a random constant value (IMU manufacturers provide with its stochastic properties); the environmental scale factor is a temperature dependent value and the dynamic scale factor is a velocity

(linear and angular) dependent value.

Misalignment R refers to the errors due to the non-orthogonality of the actual assembly mechanism and are implemented as a rotation matrix.

In order to add the non-systematic errors, white noise w is added to the signal. IMU manufacturers provide with its stochastic properties.

The system quantize the signal to simulate the A/D conversers with the following model:

$$[x/quant] * quant \tag{5.6}$$

where x is the data to correct, $quant$ is the number of quantization and $[.]$ stands for the integer part of a real number. The quantization value is also provided by IMU manufacturers.

5.2.3 System verification

An IMU does not deliver the same data twice even in the "same" conditions, so the verification had to be done by means of indirect methods [38].

In order to verify the errorless signal computed with this tool, static acquisitions across the planet were simulated firstly. The output of one of those simulations can be seen in Figure 5.5.

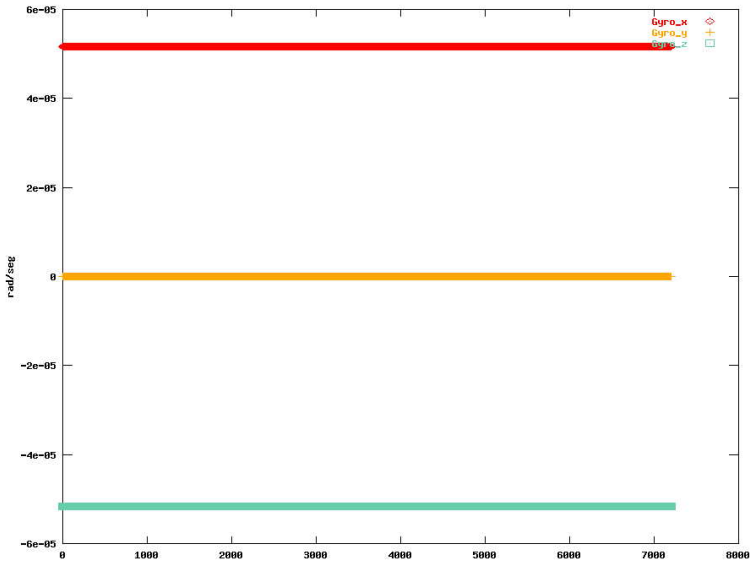

Figure 5.5: IMU_simulator: Errorless simulated angular velocities for an IMU fixed at 41º lat, 0º lon, 0 m height.

After checking that the results are the expected ones (according to the Earth's characteristics) dynamic tests were performed: from straight lines and circular trajectories to more complex ones such as a typical aerial surveying trajectories. For the acceptance of these outputs, the generated signal was processed with an INS/GPS navigation software, as the results were the reference trajectory, the errorless signal generated with the IMU simulator was verified.

As stated previously, IMU simulator add to the errorless signal a combination of errors mainly obtained from different stochastic processes (constant, random constant, Gauss-Markov, random walks, etc.) See Figure 5.6. The analysis of the stochastic variables generated by the simulator proves that effectively the processes fit with the requested ones.

5.2.4 System validation

The validation of the system revealed itself as one of the more difficult tasks of the project because of the lack of references. First step to be able to reproduce real data is to analyze it and properly characterize it. For the validation

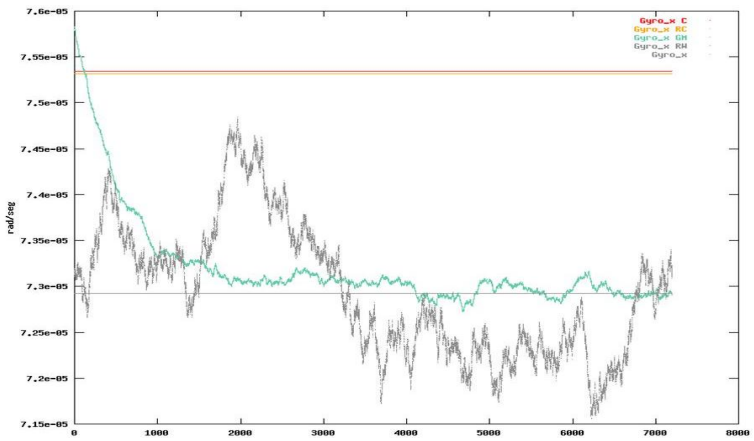

Figure 5.6: IMU_simulator: computed stochastic errors.

of the tool we selected four IMUs (a navigation grade IMU - iMAR FJI, a tactical grade one - Northon Grumman LN200 and two MEMS IMUs - EPSON S4E5A0A0A1 and Invensense MPU6500), two environmental scenarios (constant and changing temperatures) and three platforms (static acquisition, van and small airplane platforms.)

To characterize instrumental error, IMUs were submitted to long periods of static acquisition at different controlled temperatures. The acquired signal is then analyzed through Allan Variance (AV) techniques to obtain the results presented in tables 5.2 and 5.3. The Allan Variance of a time series of data is a characterization of the noise and other processes in the data as a function of averaging time. It is defined as one half of the time average of the squares of the differences between successive readings of the frequency deviation sampled over the sampling period. The Allan variance is a functions of the sampled period, as well as the distribution being measured, and it is displayed as a graph rather than a single number. For more information on AVs refer to [38].

On the other hand, to determine the platform characterization values several acquisitions were performed with the iMAR FJI in static mode, in a van and in a small airplane with the engines on. It is important to note here that the IMU selected to acquire this characterization data is the navigation-grade one, since its error are small enough to not interfere with platform vibrations. In the current approach, the platform signal has been modelled as sinusoidal vibrations whose main parameters are the frequency and the amplitude. In table 5.4, the characterization for the three scenarios is presented. In order to properly evaluate this nominal frequency Power Spectral Density (PSD) analysis of data has been performed. The PSD of a time series of data is a characterization of the noise and other processes in the data as a function of frequency. The PSD is defined to be the Fourier transform of the autocorrelation function of the data, where the autocorrelation is the expected value of the data multiplied by itself delayed. For more information on PSD refer to [38].

		Navigation "FJI"	**Tactical "LN200"**	**MEMS "EPSON"**	**MEMS "MPU6500"**
Gyros	Bias run-to-run	0.05 º/h	1 º/h	0.5 º/s	5 º/s
	Bias instability (º/h)			6	7
	Scale factor (ppm)	30	100	-	3%
	Misalignments (º)			0.1	0.02
	White Noise ($/\sqrt{h}$)	0.001	0.07	0.2	0.3
	Quantization (º/h)	0.0007	157	32	54.9
	Range (º/s)	± 450	± 816	± 300	± 500
Accelerometers	Bias run-to-run (mGal)	60	300	800	7000
	Bias instability (mGal)	-	-	10	9
	Scale factor	-	300 ppm	-	2%
	Misalignments (º)	-	-	0.1	0.02
	White Noise (mGAL/$\sqrt{Hz}$)	8	200	600	300
	Quantization (mGal)	1	1200	4.57	6.10
	Range (g)	± 5	± 20.4	± 3	± 4

Table 5.2: IMU_simulator: instrumental errors.

		Navigation "FJI"	Tactical "LN200"	MEMS "EPSON"	MEMS "MPU6500"
Gyros	Temperature	-	-	$0.03(^o/s)/^oC$	$30(^o/s)/^oC$
	Pressure	-	-	-	-
	White Magnetic field	-	-	-	-
Accelerometers	Temperature	-	-	$0.02mG/^oC$	$1.5mG/^oC$
	Pressure	-	-	-	-
	White Magnetic field	-	-	-	-

Table 5.3: IMU_simulator: environmental errors

		Static	Van	Plane
Gyro	Sinusoidal vibration freq. (Hz)	0	10	60
	Sinusoidal vibration amplitude (^{o}/h)	0	15	100
Accel	Sinusoidal vibration freq.(Hz)	0	10	60
	Sinusoidal vibration amplitude (mGal)	0	500	1e5

Table 5.4: IMU_simulator: platform errors (PSD)

In order to check that realistic simulations are performed, AV and PSD signal analysis techniques were used. With this analysis we were able to prove that the spectral characteristics of the simulated data are similar to the real ones.

To validate the system five acquisition campaigns with the real IMUs were carried out: static with and without temperature changing; van with and without temperature changing and plane with temperature changing. The static acquisition length is about 6 hours, while for van and airplane tests is around 15 minutes. Later on minimum of twenty simulations were carried out for each IMU in those scenarios. In all tests quantization noise is perfectly determined; white noise errors and inertial bias instability is computed with a precision of 10% and the combined analysis of all the tests proved that the run-to-run bias is coherent with the nominal one. For the non-static acquisitions, in the PSD analysis of both real and simulated datasets a sinusoidal behavior can be detected with the frequency and amplitude parameters differing less than 1%. These results lead us to conclude that the tool is properly mimicking IMUs.

Chapter 6

Optimal estimation criteria and quality assessment - Simultaneous prediction and filtering

In this chapter a new optimal estimation criteria is presented. This approach relies on the sequential Least-Squares (LS) approach [88]. The reader will find here the algorithm as well as how the geodetic tools, commonly applied in in geodetic estimation problems –geodetic network adjustment— can also be used in the navigation problem. The verification and validation process of this tool as a navigation tool is also presented here. Note that this validation process has been carried out by implementing this algorithm in the NAVEGA system presented in the previous chapter.

6.1 Theoretical approach

6.1.1 Considerations on navigation sensors outliers

In the school of geodetic applications, sensor errors can be decomposed in two parts [35]: a deterministic one and a stochastic (or random) one. The deterministic part can be computed and corrected through laboratory calibration and, during the trajectory determination, through the calibration process. On the contrary, stochastic errors can not be removed and the only thing that can be done is to try to minimize their effects. This is done using a correct stochastic characterization of the sensor data when using it in the trajectory determination filter. Obviously an incorrect stochastic model will lead suboptimal to unacceptable solution but, even with a correct model, the appearance of outliers can threat the filter performance, and even worse, they can threat the system reliability.

In autonomous sensors the outliers are mainly due to an incorrect relation between signal/sensor bandwidth and dynamic range and also to environmental interferences (see section 2). When speaking about sensors, three kinds of bandwidths should be taken into account: the sensor bandwidth (this is the sensor spectral response); the noise bandwidth that depends on the electronics integrated in the sensor and is different from the sensor bandwidth; and the sampling frequency. In order to reduce the noise level, it is usual to oversample the signal (hundreds or thousands of Hz in the case of IMUs) and then compute the average of some contiguous samples. This average smoothes the signal noise and downsamples the data when it is going to be transmitted through the interface. However, it can be assumed without loss of generality in this case that the sampling frequency is the output data rate. In order to keep all the sensor features in the acquired signal, it is recommended to follow the Nyquist criterion that says that the sampling frequency should be the double of the sensor bandwidth. This rule is not always kept, overall for low cost sensors. The dynamic range is the ratio between the largest and smallest values that the sensor can assume. Thus, when the signal intensity is higher than the maximum acquirable value, the sensor saturates and, many times, an outlier appears. Finally it may happen that the sensor is inappropriately used in an environment for which it is not designed. If, at a certain moment during the acquisition process, the frequency of the signal to collect is higher than the sensor acquisition frequency or the intensity of the signal is higher than the sensor dynamic range or there's some kind of environmental interference the noise error characterization will be no longer valid.

Examples of the errors presented above are: when an IMU is submitted to a shock, in general, both the dynamic range and the IMU signal bandwith are not big enough to "capture" that signal, leading to an erroneous set of accelerometer measurements that will have a direct impact firstly on the estimated velocity and secondly on the estimated position. Furthermore, since the error is not detected as a such, the estimated error of the predicted position would be also erroneous leading to an unreliable system (see Figure 6.1).

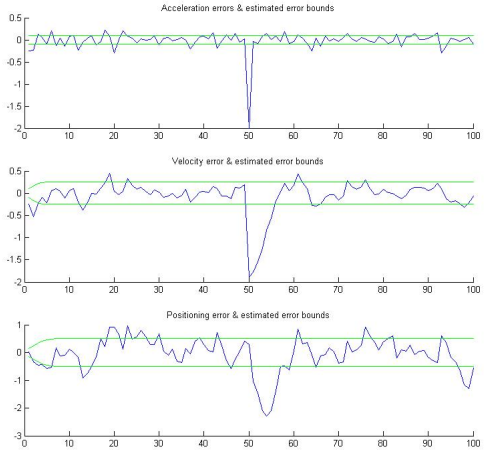

Figure 6.1: SiPF: IMU outliers and their impact on velocity and position estimations.

On the other hand, local magnetic fields affecting magnetometers or slippery roads interfering the normal behaviour of odometers(see Figure 6.2) are typical examples of environmental interferences.

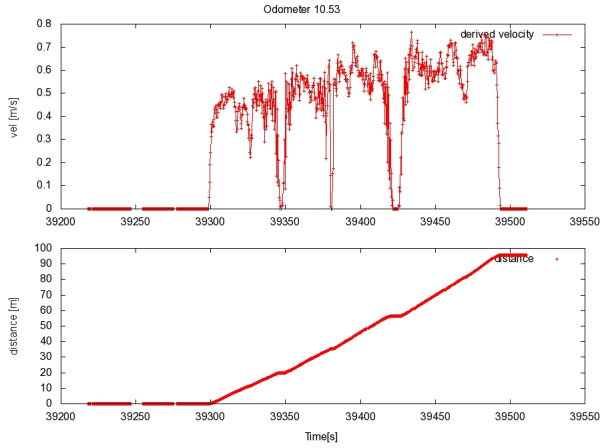

Figure 6.2: SiPF: OUtlier in odometer data.

6.1.2 Simultaneous Prediction and Filtering theoretical basis

In the Kalman filter and its many versions, in the prediction step, the observations ℓ of the dynamic models and their covariance matrices at time t are used to predict estimates of the parameters or states $x(t)$ from $x(t-\Delta t)$. Once this is done, the estimated states and their covariance matrices can be used for a new prediction step to obtain $x(t+\Delta t)$ or can be updated —improved— in a filter step if other observations ℓ_s are available. Sometimes, this two steps are computed simultaneously in what is knowns as one-step Kalman filter [36]. The one-step Kalman filter use simultaneously the prediction and the filtering step leading to a filter dynamics written in tems of the state predicted estimate.

Consider the system

$$x(t+\Delta t) = A(t)x(t) + B(t)u(t) + Gw(t) \tag{6.1}$$

$$y(t) = C(t)x(t) + v(t) \tag{6.2}$$

where $x(t) \in R^n$,$u(t) \in R^m$,$w(t) \in R^n$,$v(t) \in R^r$,$y(t) \in R^r$,$u(t)$ and $w(t)$ are sequences of white, zero mean, Gaussian noise with zero mean and covariance matrices $Q(t)$ and$R(t)$ respectively. Then, we can estimate $x(t+\Delta t)$

like follows:

$$x(t+\Delta t) = A(t)[I - KC(t)]x(t-\Delta t) + B(t)u(t) + A(t)Ky(t) \tag{6.3}$$

with initial condition

$$x(0) = x_0 \tag{6.4}$$

where,

$$\begin{aligned} K &= P(t-\Delta t)C(t)^T[C(t)P(t-\Delta t)C(t)^T + r]^{-1} && (6.5)\\ P(t+\Delta t) &= A_k P(t-\Delta t)A(t)^T - AKC(t)P(t-\Delta t)A(t)^T + G(t)QG(t)^T && (6.6)\\ P(0) &= P_0 && (6.7) \end{aligned}$$

where $P(t+\Delta t)$ is the predicted error of the state, $P(t-\Delta t)$ is the estimated error at time $t-\Delta t$ and I is the Identity matrix. From a least-squares point of view, in a filter step at time t, there are two types of observations: the predicted states $x(t)$ and the update observations. The predicted states —we will write them as $x(t)^-$— become, at time t, observations that are related to the unknown states $x(t)$ through a simple functional model $x(t)^- + v = x(t)$ and a complex stochastic model described by the [full] covariance matrix of the predicted $x(t)^-$. In general, the additional observations or update measurements ℓ_s are related to the unknown states by functional models $g(\ell_s + v_s, x(t))$ that can be more or less complex and by stochastic models that use to be simple because their covariance matrix uses to be diagonal or block diagonal. After the filter step, optimal estimates $\hat{x}(t)$ for $x(t)$ are obtained together with residuals $\hat{v}$, $\hat{v_s}$ for the observations.

An outlier in the measurements ℓ_s has the chance to be detected by statistically testing its residual $\hat{v_s}$. Analogously, an outlier in $x(t)^-$ has the chance to be detected through its residual $\hat{v}$. However, if we assume that $x(t-\Delta t)$ was an error free state, an outlier in $x(t)^-$ can only be produced by an outlier in ℓ. Unfortunately, the observations ℓ do not receive residuals and therefore, an outlier in ℓ has no chance to be detected through a residual. In other words, after the prediction step, the least-squares engine that performs the filtering does not see ℓ. Of course, since $x(t)^-$ is obtained by solving a stochastic differential equation with initial values $x(t-\Delta t)$ and observations ℓ, an outlier in ℓ will result in an outlier in $x(t)^-$; however, the influence of a single error in ℓ will be spread into a multiple error affecting various components of $x(t)^-$ which makes error detection and identification a difficult task.

Simultaneous Prediction and Filtering is a natural extension of the one-step Kalman filter that aims at avoiding the above situation by making ℓ and ℓ_s simultaneously participate in the estimation of the state $x(t)$ without going through $x(t)^-$. In Figure 6.3 and Figure 6.4 the flowcharts of the classical approach versus the one-step approach presented in this research are shown.

Hereafter, the prove that the estimated solution of both approaches will be the same is presented. Let us write the stochastic differential equation that models the evolution of $x(t)$ as

$$\dot{x} = f(t, x, \ell + v) \tag{6.8}$$

where ℓ is the measurements' or observations' vector and $-v$ its unknown error. We define now, the transition function Φ that maps $x(t)$ into $x(t+\Delta t)$

$$\begin{array}{ccc} & \Phi & \\ \mathcal{F} \times R^u \times R^n \times R & \longrightarrow & R^u \\ (f, x(t), \ell, \Delta t) & \rightarrow & x(t+\Delta t) \end{array} \tag{6.9}$$

or $\Phi(\Delta t)$

$$\begin{array}{ccc} & \Phi(\Delta t) & \\ \mathcal{F} \times R^u \times R^n & \longrightarrow & R^u \\ (f, x(t), \ell) & \rightarrow & x(t+\Delta t) \end{array} \tag{6.10}$$

so we can write

$$x(t+\Delta t) = \Phi(\Delta t)(f, x(t), \ell) = \Phi(f, x(t), \ell, \Delta t). \tag{6.11}$$

The update of the classical two-step kalman filter at time $t+\Delta t$ can be formulated as a least-squares parameter estimation problem as follows:

$$\begin{aligned} g1:0 &= x(t+\Delta t)^- - x(t+\Delta t)\\ g2:0 &= g(x(t+\Delta t), \ell_s + v_s) \end{aligned} \tag{6.12}$$

with observations $x(t+\Delta t)^-$, ℓ_s and unknown parameters or states $x(t+\Delta t)$. The covariance matrices of the observations are $C_{x(t+\Delta t)^- x(t+\Delta t)^-}$ and $C_{v_s v_s}$. Where

$$C_{x(t+\Delta t)^- x(t+\Delta t)^-} = \frac{d\Phi}{dl} C_v \frac{d\Phi}{dl}^T + \frac{d\Phi}{dx(t)} C_{x(t)} \frac{d\Phi}{dx(t)}^T \tag{6.13}$$

In the same way a simultaneous prediction and filtering step at time $t + \Delta t$ can be formulated as a least-squares parameter estimation problem as follows:

$$\begin{aligned} g1:0 &= x(t+\Delta t) - \Phi(f, x(t) + v, \ell + v, \Delta t) \\ g2:0 &= g(x(t+\Delta t), \ell_s + v_s) \end{aligned} \tag{6.14}$$

with observations $x(t)$, ℓ, ℓ_s and unknown parameters or states $x(t + \Delta t)$. The covariance matrices of the observations are $C_{x(t)x(t)}$, C_{vv} and $C_{v_s v_s}$.

It can be easily proven that applying the least-squares parameter estimation formulation to both systems

$$\begin{aligned} x(t+\Delta t) &= N^{-1} A^T (D C_{ll} D^T)^{-1} g \\ C_{x(t+\Delta t)x(t+\Delta t)} &= N^{-1} \\ N &= A^T (D C_{ll} D^T)^{-1} A \end{aligned} \tag{6.15}$$

where A is $\frac{dg}{dx}$, D is $\frac{dg}{dl}$ and C_{ll} is the overvations covaiance matrix, results in exactly the same result:

$$\begin{aligned} C_{x(t+\Delta t)x(t+\Delta t)} = N^{-1} &= \left(C^{-1}_{x(t+\Delta t)^- x(t+\Delta t)^-} + \frac{dg2}{dx}^T \left(\frac{dg2}{dv_s} C_{v_s v_s} \frac{dg2}{dv_s}^T \right)^{-1} \frac{dg2}{dx} \right)^{-1} \\ x(t+\Delta t) &= N^{-1} \begin{pmatrix} C^{-1}_{x(t+\Delta t)^- x(t+\Delta t)^-} & 0 \\ 0 & \frac{dg2}{dx}^T \left(\frac{dg2}{dv_s} C_{v_s v_s} \frac{dg2}{dv_s}^T \right)^{-1} \end{pmatrix} g \end{aligned} \tag{6.16}$$

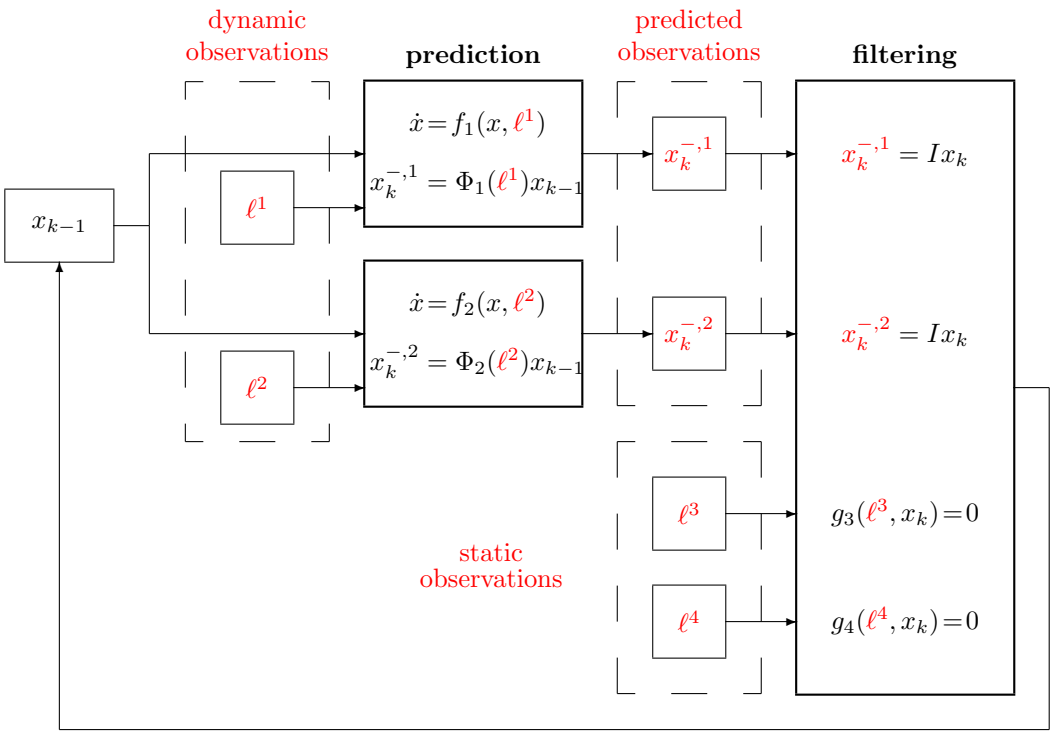

Figure 6.3: SiPF: classical prediction-correction flowchart

6.1.3 Considerations on practical implementation

In order to implement the previous idea, it is recommended to use a numerical approximation of the transition matrix.

Two considerations have to be taken into account when stating the problem of the numerical integration of equations: the first one is related with the numerical method and the second one with the nature of the observations. The ideal numerical method would be a method of high accuracy, step-size independent and fast. Unfortunately, usually high accuracy requirements increase the computational burden, making the method slower. Hence, a compromise

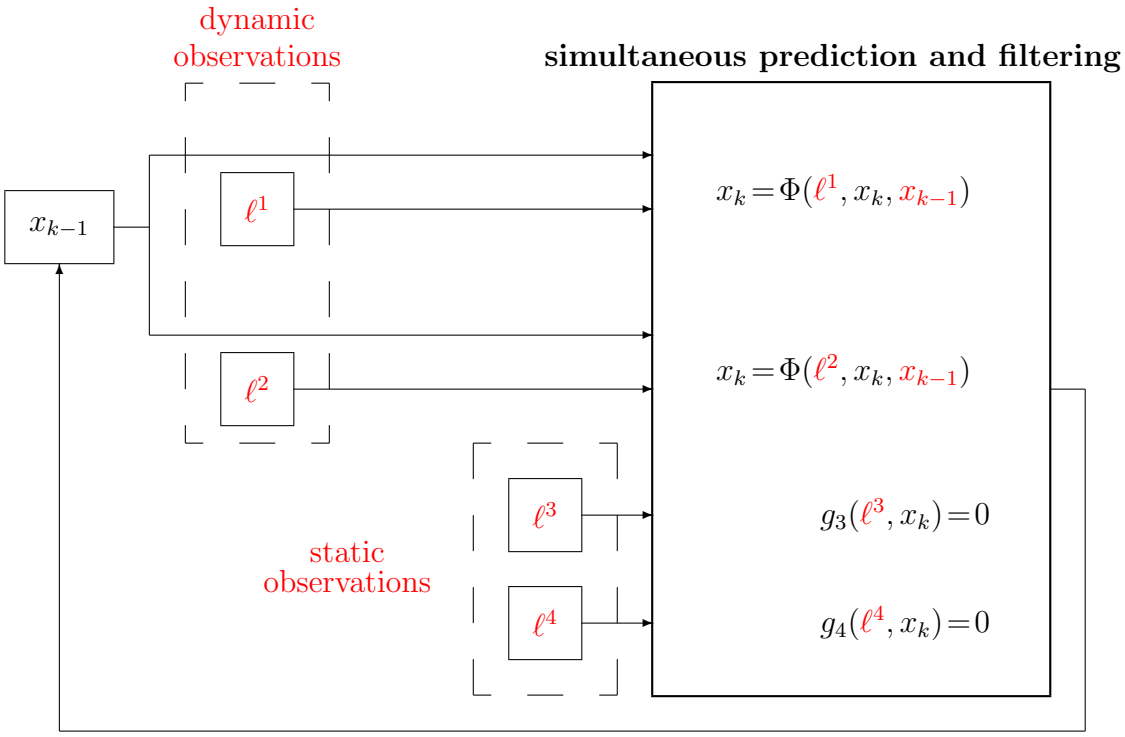

Figure 6.4: SiPF: proposed flowchart

between accuracy and velocity is required. For navigation purposes, fourth-order methods are a good compromise and usually they fulfill the accuracy requirements. A step-size independent method would not be essential if the field defining the differential equation could be evaluated at any epoch [71]. For example, if the field had an analytical expression not depending on external observations, this condition would not be necessary. Unfortunately, this is not the case of the INS mechanization equations. The mechanization equations depends on the inertial measurements provided by the IMU. Again, step-size independency would not be required if the frequency at which the IMU provides inertial measurements would be constant (typically IMUs output inertial measurements between 50Hz and 400Hz, what is enough for integration purposes). But, as said previously, although there is a nominal frequency, the inertial observations may slightly depart from it; even worse, data gaps might occur. Hence, the method becomes step-size dependent.

Multistep methods use the approximations $\hat{x}_k, \hat{x}_{k+1}, ..., \hat{x}_{k+r-1}$ to calculate $\hat{x}_{k+r}$, $r \geq 2$. The order of these methods is strongly related to the number of steps r. Details on its implementation can be found in [71]. It is important to remark that, in order to start the method, r initial values are needed, that is, r estimations of the solution. These will not be usually known; just a single initial condition. But this problem is avoidable. To achieve r initial values, it is worth to start with the Euler method from the initial condition to get a second value. Then, with this two values, second-order prediction-correction can be used to get a third one, and so on, until r values to start the rth order Predictor-corrector method are achieved.

We note that the algorithm proposed here implies a heavy computational burden compared with any version of the Kalman filter. This is mainly due to the propagation of the covariance matrix at high rate, specifically to the involved matrix inversion. In any case, the strength of new computer processors makes this overhead less significant when compared to the overall system performance.

6.1.4 Application to integrity monitoring of multi-sensor navigation systems

Generally speaking, multi-sensor navigation systems combine two types of observations; those that are related to the navigation [and calibration] states through differential equations and those that are related to the states by non-differential ones. In the previous section we called them dynamic and static observations simultaneously. Thus, for instance in terrestrial navigation, typically one or more inertial measurement units, baroaltimeters, odometers, and GNSS receivers are used. In this case, the introduced simultaneous prediction and filtering can be applied to configurations where there are only dynamic observations, only static ones or combinations of both them.

It can be easily demonstrated that SiPF offers the same parameter estimation performance that the two-step LS

filtering and that the EKF. The difference lies in in the modeling capability and in the available quality analysis tools.

When considering the estimation as a control problem, the dynamic -control- equations are integrated in time until an external observation is available. This observation allows to improve parameter estimation by applying restrictions on this parameter. The user can not apply restriction equations on the measurements controlling the system, only on the parameters. Considering the problem as a least squares adjustment problem the user can easily include restrictions over any of the measurements involved in the computation -even over the dynamic ones. This is a specially useful capability when working with systems involving redundant sensors with known relative positions and attitudes among them.

The main advantage of considering the previous process as a sequential least squares adjustment instead of a Kalman filter process is that we can apply the powerful quality analysis tools established by the Dutch geodesist Willem Baarda after the II World War [8]. Considering the estimation as a least squares adjustment implies having access to the residuals of all the observations -even the dynamic ones- and allows to apply to them all the quality analysis tools extensively developed in the geodetic community, not only the ones typically used by the control community. Thus, in addition to controllability and observability analysis -that allow to understand the parameter's determinability and their capability to be influenced- more analysis can be done based on the residuals of the computation. The use of estimators like variance component or the sigma naught could imply an improvement of the stochastic modeling. Moreover, the redundancy matrices would allow the evaluation of the system design and geometry and would allow to perform analysis of suitable mountings for multisensor systems. The analysis of those matrices would also allow to compute the minimum detectable errors for each sensor and would also allow to know the impact of a malfunction in each sensor on the solution. Finally, it is important to remark that the LS approach would allow to detect outlier measurements in systems with redundant sensors - even outliers in dynamic measurements like the ones presented on section *6.1.1*.

6.2 Concept validation

6.2.1 Validation approach

The Simultaneous Prediction and Filtering algorithm has been implemented in the NAVEGA platform (section 5.1.) It is important to note here that the NAVEGA platform also includes an Extended Kalman filter and a Sequential Least Squares filter.

The validation process can be divided in three phases:

- **Solution correctness.** On the first place, the test data sets have been processed with the three approaches (Kalman Filter, Sequential Least Squares and SiPF) and the results have been compared in order to check that without outlier detection and exclusion techniques the results are the same.
- After that an **analysis of the residuals** has been performed. As in any least squares adjustment, the distribution function of the estimated residuals should correspond to the distribution function of observations noise. If not, it means that observations error modelling is not correct threatening system reliability. In a classical sequential least squares adjustment, the residuals analysis could be applied only to the measurements involved in the updating step, while in a SiPF approach we will have access to the residuals of the measurements involved in the updating step - they should be the same that in a LS approach- and also to the measurements controlling the dynamic process.
- After checking that the error of all the measurements involved in the computation are correctly characterized, geodetic techniques for **outlier detection and removal** are applied to the LS approach and to the SiPF approach. Number of outliers correctly detected has been listed and checked.

6.2.2 Validation data sets

The design of the validation data sets aims to cover the situations presented in section *6.1.1* where outliers are an issue. In this study two system configurations, in a terrestrial scenario including three simulations mode were envisaged. All data sets consist on simulated measures for a terrestrial trajectory following the path presented in Figure 6.5. The data sets were prepared using the simulation tools (section 5.2.)

The first system configuration consists on data from a time synchronized system including a GNSS receiver, an IMU and an odometer. The second one includes data from a time synchronized system including a GNSS receiver, three IMUs and a magnetometer. The specification of all those sensors are presented in Table 6.1.

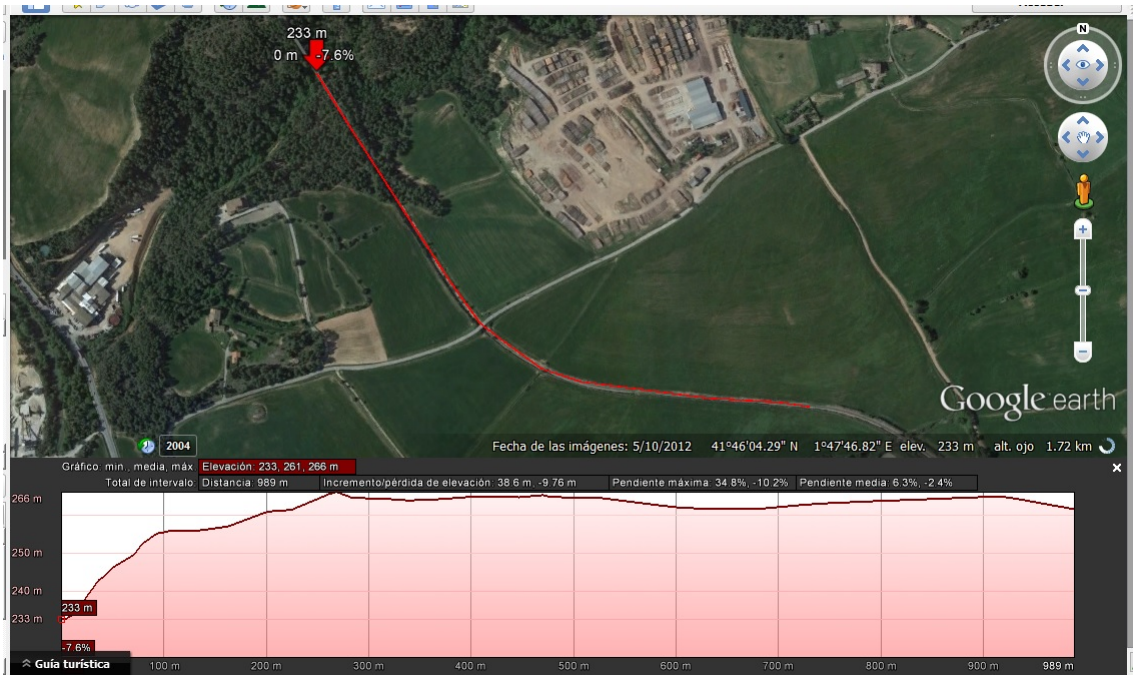

Figure 6.5: SiPF: algorithm's validation reference trajectory.

For each configuration three modes of simulation have been used: outliers free, punctual outliers and realistic outliers. The outliers free mode assumes that there are no gross errors in data; in the punctual outliers mode punctual errors have been introduced to the data sets in order to evaluate the algorithm performance; and finally in the realistic outliers mode errors similar to the ones that may happen in the field are introduced. For the first system configuration (GNSS/IMU/odo) the punctual outliers mode simulation consists on adding single gross errors in the odometer data; while the realistic outliers mode consists on adding "slide" measures in the odometer data. For the second system configuration (GNSS/RIMU/magneto) the punctual outliers mode simulation consists on adding single gross errors in one of the IMUs and also in the magnetometer; while the realistic outliers mode consists on adding "shock" measures in the RIMU data. Please note, that since the aim of this research is to see the capability of SiPF to detect ouliers in dynamic observations, no outliers have been introduced in GNSS data sets.

	Frequency	Data	Type of observation	Noise
GNSS receiver	10 Hz	pos.	static	10 cm
IMU-TG1	50 Hz	angular vel., linear accel.	dynamic	0.3 deg/$\sqrt{h}$,0.034 mg/$\sqrt{Hz}$
IMU-TG2	50 Hz	angular vel., linear accel.	dynamic	0.4 deg/$\sqrt{h}$,0.064 mg/$\sqrt{Hz}$
Odometer	50 Hz	vel.	dynamic	2 cm/s
Magnetometer	1 Hz	magnetic field intensity	static	1e-4 Teslas

Table 6.1: SiPF validation: sensors' specification.

	Sensors	Outliers free	Punctual outliers	Realistic outliers
Conf 1	GNSS, IMU-TG1, odo	No gross errors	Errors in odometer	Sliding odometer
Conf 2	GNSS, magnet IMU-TG1(x2), IMU-TG2	No gross errors	Errors in one imu and magnet	Shocks in IMUS

Table 6.2: SiPF validation: datasets sum up.

6.2.3 Validation results

Solution correctness

The outliers free data sets and models were tested with the three processing algorithms leading to the same results. No difference was observed between KF, sequential LS and SiPF.

Analysis of the residuals

In table 6.3 the standard deviation of the sensors' residuals in outlier free tests are presented. These residuals are obtained from SiPF algorithm. In particular, the analysis of the residuals in the SiPF test allow us to check the IMU data consistency. In both data sets the residuals are distributed as white noise with a distribution coherent with the simulated data.

	GNSS receiver (cm)	IMU-TG1 ($deg/\sqrt{h}$, $mg/\sqrt{Hz}$)	Odometer (cm/s)	Magnetometer (Teslas)
Conf 1	8	0.15, 0.020	1.2	-
Conf 2	8	0.12, 0.018	-	8e-3

Table 6.3: SiPF validation: Standard deviation of the residuals of outliers free testing data sets.

Outlier detection

All data sets have been processed with IEKF and SiPF. In order to check improvement on outlier detection and removal, time windows of 5 seconds centred on the gross errors have been analysed.

For the GNSS/INS/odometer configuration the results are the ones presented in table 6.4. The results demonstrated that odometer punctual gross errors have been detected and correctly removed. Moreover, it can be seen that odometer "slides" are detected but can not be fully removed.

	Position cm	Velocity cm/s	Attitude deg
Outlier free mode - IEKF	0.1	0.07	0.05 / 0.02 / 0.03
Outlier free mode - SiPF	0.1	0.07	0.05 / 0.02 / 0.03
Punctual outlier mode - IEKF	0.3	0.12	0.05 / 0.02 / 0.03
Punctual outlier mode - SiPF	0.1	0.07	0.05 / 0.02 / 0.03
Realistic outlier mode - IEKF	5.0	3.02	0.05 / 0.02 / 0.03
Realistic outlier mode - SiPF	0.7	0.13	0.05 / 0.02 / 0.03

Table 6.4: SiPF validation: GNSS/INS/odometer solution errors for different modes

For the GNSS/RINS/magnetometer configuration the results are the ones presented in table 6.5. The results demonstrated that IMU punctual gross errors can be detected but can not be removed with the SiPF approach. Moreover, it can be seen that IMU "shocks" can be detected but can not be removed with the SiPF approach. In figure 6.6, actual versus estimated errors is plotted. It can be seen that using classical filters the outliers lead not only to an erroneous trajectory estimation, but to "chitting" the user about the quality of the results. SiPF allows to, at least, warn the user, about problems in the data set.

	Position cm	Velocity cm/s	Attitude deg
Outlier free mode - IEKF	0.2	0.07	0.05 / 0.02 / 0.03
Outlier free mode - SiPF	0.2	0.07	0.05 / 0.02 / 0.03
Punctual outlier mode - IEKF	3.0	0.12	0.07 / 0.05 / 0.06
Punctual outlier mode - SiPEF	0.1	0.05	0.07 / 0.05 / 0.06
Realistic outlier mode - IEKF	3.0	0.12	0.07 / 0.05 / 0.06
Realistic outlier mode - SiPF	0.5	0.08	0.07 / 0.05 / 0.06

Table 6.5: SiPF validation: GNSS/RINS/magnetometer solution errors for different modes

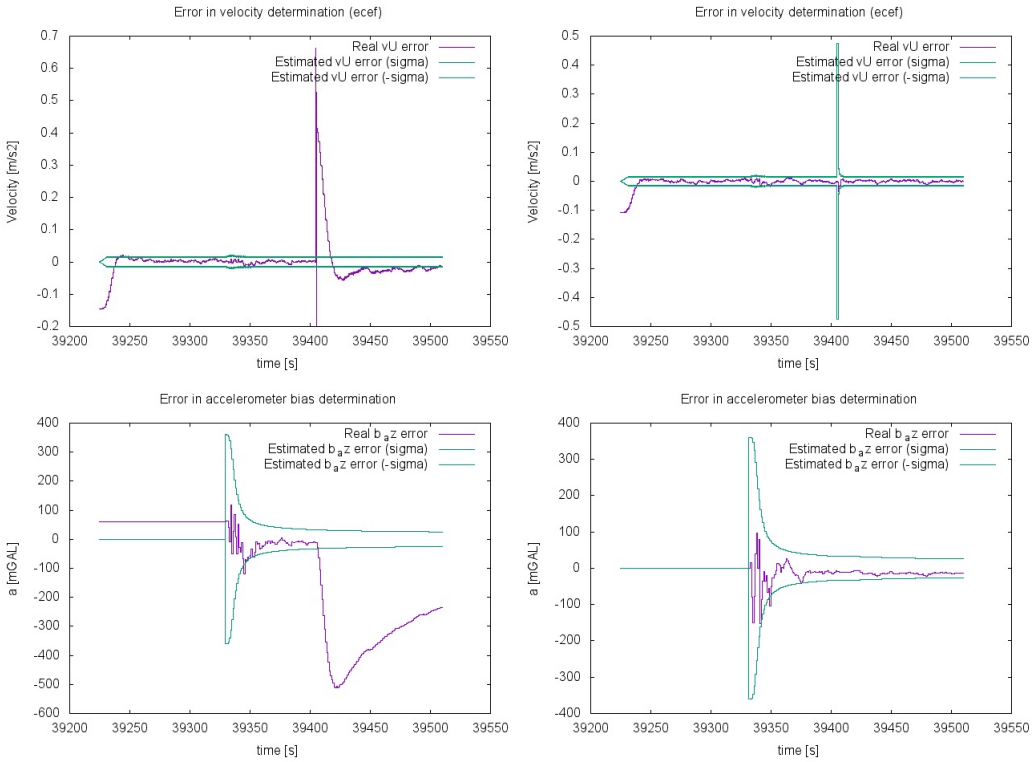

Figure 6.6: SiPF validation: GNSS/RINS/magnetometer solution errors without (left) and with SiPF(right).

6.2.4 Discussion

As it has been theoretically proved in seccion 6, the results proves that there are no difference between KL, sequential LS and SiPF when working with data sets without gross errors. The tests also prove that SiPF can be used to properly characterize the error of the IMU measurements. The analysis of the IMU residuals shows that those residuals behaves properly. In case some inconsistency appear, this would be due to an improper characterization of the residuals. SiPF has been proven as a useful tool for outlier detection when redundancy in dynamic measurements occur. Thus, when punctual errors happen with favourable problem geometry (this is, several different sensors) sensors, SiPF is able to detect and to remove the errors. With unfavourable geometries (several sensors of the same type) errors can only be detected. The same thing happens with "real" error, this is, not punctual errors but errors that happen during short periods of time. SiPF is able to detect them although is not able to remove them. It may seem that this is not enough, but that's not true. Being able to detect those problems allow the system to properly characterize the solution error leading to an improvement on solution reliability and also allowing the system to recover quicker, this is improving the solution performance earlier.

Chapter 7

Rigorous modelling, redundancy and hetereogeneity - Benefits

Although as stated in chapter 1 this thesis does not focuses neither on rigorous modelling, redundancy nor data hetereogeneity, a brief comment on those topics is presented hereafter in order to highlight the relevance of this Geodetic pillars in an improved navigation approach. The reader will find here former IG's and current CTTC' Geodesy and Navigation group research works, in which the author has been directly involved, and where geodesy principles are the fundamental driver.

7.1 Modelling and rigorous modelling

While modelling tries to represent something through a description of its properties, rigorous modelling tries to go a step beyond an to represent it good enough that the only thing that is left is noise. In other words, rigorous modelling refers both to functional and to stochastical sensor modelling. These models represent the relation of the data acquired by the sensor and the parameters to be estimated and also the behaviour of sensor errors. In order to show the relevance of properly modelling these, hereafter, some works in which the author of this thesis has been involved are presented.

7.1.1 IMU characterization

For the first high grade sensors, classical mechanical calibration methods were designed requiring specialized platforms [90]. For example, the six-position method or the angular rate tests [90]. But these tests required a high cost specialized platform. Recent works at at CTTC's group aims to bring the high-grade IMU calibration approach for low-cost sensors using simple 2 degree-of-freedom platforms. In that frame, a very low cost platform ($< 1000€$) has been developed allowing for frequent IMUs characterization, and thus for improved in-lab calibration of low-cost systems [25].

Another approach for low-grade sensors, involve analytical methods: special discrete parametric models such as Auto-Regressive (AR) models, Moving Average (MA) models, Mixed AR and MA (ARMA) models or harmonic models can be used [65]. In this context, Guerrier et al. [32] propose a Mahalanobis distances based complex approach to detect outliers that takes into account the shape and size of multivariate data. In this topic, the author participated in a research lead by A. Waegli [93] that proposed to use a statistical method employed for studying the volatility of financial markets (GARCH). The method was adapted and tested for the usage with inertial redundant data.

Both tools have allowed the researchers to improve the characterization of the available equipment and thus to be able to 1) generate better simulations and 2) better characterize stochastic models in optminal estimation algorithms.

7.1.2 GNSS' Clock characterization

GNSS technology relies mainly on clocks, (satellite) atomic clocks and (receiver) quartz oscillators. Although the modelling of the first ones was precisely done, the second ones were roughly modelled, and their errors have been typically considered as parameters to be estimated together with receiver position. Lately, with the apparition of low-cost ($< 3000€$) atomic clock's this could change. One of the research lines in which the author has been involved lately [24] proved that the Chip Scale Atomic Clock (CSAC) makes it possible to estimate position within specifications even when only three satellites are available. It has also been observed that the impact of reducing

and simplifying the clock unknown has almost no impact in planimetry performance neither for static nor dynamic trajectories. However, the results demonstrate that having access to these precise clocks has a relevant impact on the performance of height estimations in both scenarios.

Again, this research has allowed the researchers to improve the characterization of the available equipment and thus to be able to 1) generate better simulations and 2) better characterize stochastic models in optminal estimation algorithms.

7.2 (INS) Redundancy

In 2004 the former Institute of Geomatics presented two new algorithms for the integration of GPS with redundant IMUs. In the first one the inertial measurements were combined in the observation space to generate a synthetic set of data which is the integrated with GPS by the standard algorithms. In the second approach, the method of strapdown navigation was adapted in order to account for the redundant measurements [13] [49]. Then, later on, in 2008 the È cole Politechnique Fédéral de Lausanne, in close collaboration with the research group where the author of this thesis belongs, presented the results of an evaluation of the previous algorithms. Both methods were evaluated in experiments where redundant MEMS-IMUs were fixed in different geometries: orthogonally, redundant and skew-redundant IMUs. For the latter configuration, the performance improvement using a synthetic IMU was shown to be 30% on the average. the extended mechanization approach provided slightly better results (about 45% improvement) as the systematic errors of the individual sensors were considered separately rather than their fusion when forming compound measurements. The maximum errors were shown to be reduced even by a factor of 2 [92].The redundant approach has been consolidated at our geomatics group in later european projects [56] and internal developments [76] as a key factor for performance improvement at reduced cost.

The use of redundant IMUs together wth the use of SiPF would allow to reduce outliers impact in navigation solution and thus, to improve system reliability.

7.3 Hybridization

Research teams of DEIMOS and the former Institute of Geomatics had been working closely on the analysis of the performance of different INS/GNSS hybridization levels for several years. In 2007 the results achieved using tightly coupled architectures, also termed deeply coupled, under different scenarios, using aiding and coasting techniques for Galileo signals were presented [82]. According to this research the superior performance of such architectures is specially suited for receivers operating in demanding environments; for instance, urban scenarios, where signal blockage is frequent, and high dynamics, such as accelerating trains. Also important is the contribution in the reduction of the total error budget and increased robustness of satellite signal tracking, ultimately allowing to explore new applications and new possibilities of GNSS. The hybrid receiver was implemented in the GRANADA software receiver. In the patent PCT/PT2007/000021 these results are described in detail.

Both optical and LiDAR sensors are recently used for navigation purposes. This hybridization is generally based on features analysis. A specific target is localized in consecutive images and the different points of view allow us to correct the IMU drifts. Multisensor integrated solutions that combine the complementary features of the GPS, laser scanner feature-based navigation, and inertial navigation for urban scenarios is a solution under investigation at our group [23], being the preliminar results very promising. Features (such as lines) are extracted from laser scan images and then used for position and attitude determination [5].

The use of heterogeneus system, properly characterized, together wth the use of SiPF would allow to reduce outliers impact in navigation solution and thus, to improve system reliability.

Chapter 8

Conclusions

The main thesis contribution has been to show that all the geodetic pillars: abstraction, optimal estimation, rigorous modelling, redundancy and data hetereogeneity are keystones in the achievement of precise, accurate, available and reliable navigation.

Abstraction allows the design, implementation and validation of new trajectory determination algorithms in a quick and easy way. An inertial-based navigation abstraction model has been defined and implemented successfully (GEMMA) proven the benefits of such an approach instead of devoted software and tools.

Optimal estimation. EKF and their multiple variations allow state estimation, some of them propagate the uncertainties with error models more accurate than the Gaussian one and some of them include algorithms that are robust against outliers in the measurements (updating step) or against mis-modeling of the dynamic process noise. But in INS/GNSS systems the dynamic processes are controlled by inertial observations and to the authors' knowledge none of the current bayesian implementations allow to monitor and thus to protect the system against outliers in all -dynamic and static- observations. The filter proposed in this dissertation considers the estimation problem not as a control problem but as a least-squares adjustment problem. Thus, all observations play the same role in the estimation and all of them could be treated in the same way. The method allows to have access, in updating epochs, not only to the innovations (and residuals) of the observations used for the updating but also for the ones controlling the dynamic process. Moreover, there are not a posteriori densities to be considered, an thus the effect of the non-linearities will be less dramatic. The proposed approach also benefits from the optimized least squares algorithms-in the computational burden aspect- and allows to work with problems where the number of states varies along the time. Thus, a direct consequence of using this approach is an improvement on sensor error modelling, end thus leads to a more reliable solution estimation. This approach is of special interest when working with redundant systems like the ones available in robust and reliable UAV navigation system [7] or in the robust and reliable medium-cost terrestrial navigation systems [27].

Rigorous modelling, **redundancy** and **data hetereogeneity** have been widely used by the navigation community beyond the examples provided in this report. This proves its fundamental role in advanced navigation solutions.

8.1 Thesis outcomes

The work carried out in the frame of this thesis results in a series of useful tools, algorithms and methodologies that proves that the geodetic approach is a suitable approach for the navigation community.

Firstly it has allowed to develop the GEMMA system. GEMMA is currently an essential tool for the research of the Geodesy and Navigation group of CTTC. In particular the simulator allows, and in fact has already allowed, feasibility analysis saving initial investment in equipment and campaigns. The tool has already been used in several European projects to perform feasibility analysis for kinematic airborne gravimetry (GAL project) [84], urban mobile mapping performance analysis (ATENEA project) [22] and the design of the payload navigation system of a drone (CLOSE-SEARCH project) [56].

The main component of GEMMA is NAVEGA. NAVEGA is a modular, generic and extensible trajectory determination tool requiring only dynamic libraries for its extension to new sensors. This approach responds to some of the modern system design challenges as described in [30] and [31]. The design starting point is the abstraction of the traditional state-space approach navigation algorithm in two main components: estimation and modelling. This concept translates into a SW module that allows for fast and incremental modelling of multi-sensor navigation systems. It is expected that the tool will benefit both navigation and geomatic communities, particularly the Earth

Observation (EO), mapping and surveying teams that perform primary data acquisition in kinematic mode be it from airborne, terrestrial and marine manned and unmanned vehicles.

GEMMA also includes an IMU simulation tool that has been proved extremely useful for researchers in the navigation field. The simulator allows, and in fact has already allowed, feasibility analysis saving initial investment in equipment and campaigns. The tool has already been used in several European projects to perform feasibility analysis for kinematic airborne gravimetry (GAL project) [84], urban mobile mapping performance analysis (ATENEA project) [22] and the design of the payload navigation system of a drone (GINSEC project) [60]. Next steps to be done in the tool development are the improvement of the IMUs characterization instrumental errors in dynamic conditions, and a fine characterization of platform induced noise.

In its current version, NAVEGA includes a traditional EKF but also a new geodetic oriented algorithm: the Simultaneous Prediction and Filtering filter.

8.2 Thesis impact on the design of accurate and reliable navigation systems

The main lesson learned after the end of this thesis refers to the way of approaching the design of an accurate and reliable navigation system. Before this work, the main approach would have been to select an IMU fulfilling the specifications, select the best GNSS available, type an adhoc Kalman filter and done. After this work the author of this dissertation proposes to do it differently, first step should be to select the a priori most suitable sensors (this may include one or several IMUS, one or several GNSS receivers and other sensors). Then to carry out a complete set of simulations based on those sensors performance. Once validated the sensors, the advice is to develop a generic Kalman-filter based SW that allows to replace the sensors at any moment and even increase the number of them if needed. Such a SW should include input/output inspectors to ensure solution performance and reliability, and if several IMUs are available it should include an estimator like SiPF, that take into account all the information provided by them.

Chapter 9

Outlook

The thesis presented in this dissertation is only a brick in the path of navigation related research, many more is still pending to be faced. In this chapter a short overview on this tasks is presented.

In this research a major departure from the mainstream research lines of KF (UKF, PF, ...) has been presented; its inspiration originating in the methods of geodesy and its motivation in the limited practical use of the mentioned KF heirs. This departure, the SiPF, allows for the application of the geodetic–a.k.a. residual-based– methods to the analysis, modelling and, in general, outlier detection and isolation to all measurements participating in the trajectory determination. An obvious example is the Fault detection, isolation, and recovery of mission critical applications where SWAP-C (Size, Weight, Power and Cost) matters. Moreover, new technology is being produced and ad-hoc methods let to cumbersome, intricate software which is difficult, expensive and length to be developed, and then even much more expensive to be certified. It goes without saying that a thorough, systematic, even massive, empirical testing is required. Also, the numerical implementation of SiPF needs further investigation:whats the best numerical integration method or how to compute efficietly the LS among other. All this is worth doing as, for the first time since the invention of KF, about 60 years ago, we will be able to see te residuals of our dynamic measurements and this opens new line of research and practical processing.

Acknowledgements

The research reported on this thesis has been partially funded by the Spanish Ministry of Education and Science, through the GENIA project of the Spain National Space Research Programme (reference: ESP2005-07599.)

En primer lloc m'agradaria donar les gràcies al Dr. Ismael Colomina, per tot el teu suport i la teva infinita paciència amb mi. Tot i que aquesta tesi ha estat més llarga que un dia sense pa, sempre hi has estat quan t'he necessitat. Voldria agrair-te un cop més haver-me ensenyat el significat de la paraula Geomàtica i haver-me obert les portes a aquest món tant apassionant. Darrera totes les coses ben fetes d'aquest treball hi ha la teva mà. Darrere els errors només la meva tossuderia.

Aquest projecte ha estat un camí que mai he fet sola. En tots aquests anys he tingut la oportunitat de treballar amb molta gent i tots i cadascú de vosaltres m'heu ajudat a avançar una mica. A tots els companys de les empreses amb les que he tingut el plaer de treballar, gràcies; als companys del CTTC, gràcies; a tots els companys de l'Institut de Geomàtica, GRÀCIES!!!. De tots, voldria destacar el suport d'en Michele Crosetto, per ser tant insistent; d'en Guido, l'Eva, la López, el Jonathan, la Núria, la Maria i l'Anna, per l'optimisme i la vitalitat; de l'Enric i en David, perquè amb vosaltres no paro d'aprendre coses noves cada dia; de l'Edgar i en Pepe perquè van ser els primers a confiar en mi i sempre hi han estat quan els he necessitat; d'en Pere per les llargues discussions i d'en Pep, per estar sempre disposat a escoltar les meves tribulacions i per ensenyar-me la màgia del software.

Per a poder dur a terme un projecte com aquest, no n'hi ha prou amb bona voluntat i una sèrie de coneixements previs. Es necessita, com a la vida, gent que et doni suport incondicional i que et dediqui temps, temps de recerca però també temps personal, d'esbarjo, d'alegria, d'amistat. Sóc afortunada, tinc aquesta gent: Uri, Swing-Woman, Flacuchina... GRÀCIES, el vostre suport és el 22.35% d'aquesta tesi! A tu, Marta, la més insistent de totes, que m'has cuidat, m'has protegit, m'has renyat, m'has ensenyat i per sobre de tot m'has estimat i t'has deixat estimar, tant en els moments fàcils com en els difícils, MIL GRÀCIES!!

No puc oblidar als amics no-geomàtics, perqué el vostre punt de vista sobre el que fem sempre m'ha ajudat a tirar endavant. Moisès, Carol, Olga, Dani, Marc, David, Jordi i Esther. No canvieu mai.

Aquest text també és possible gràcies a la meva família , la que venia de sèrie i la nova. Tiet, tieta, Marc, Gina, Oncle, Pilar, Benet, Laura i Àngel, gràcies per tot el vostre suport, el logístic i el moral.

Diuen que la "vida te da sorpresas, sorpresas te la vida" i jo afegiria que la Geomàtica també! Gràcies Eduard, per ser-hi, per escoltar, per opinar, per respectar-me, per demanar-me, per donar-m'ho tot, perqué cada cop que he tirat la tovallola l'has recollit i me l'has tornat a donar. Per recordar-me qui sóc, per recordar-me qui som. Ets la meva llum, ets la meva alot.

I finalment el reconeixement més important, el dels principals responsables d'aquest treball: els meus pares. El meu pare em va ensenyar a ser curiosa, a preguntar-me el perqué de tot i a buscar sempre respostes a les meves preguntes. La meva mare em va ensenyar a ser persistent i a no rendir-me mai davant de les dificultats de la vida. El seu exemple m'ha dut a ser qui sóc i fer el que faig. El seu exemple ha permès que avui, tu, estiguis llegint aquestes línies. Per vosaltres papes. Us estimo.

GEMMA enabled projects and works

GEMMA or some of their components have been used in several european, national and regional research projects. Hereafter a list of the main relevant ones is presented.

Research Projects

- GENIA, "Galileo Enhanced Navigation with Inertial Aiding", Programa Nacional del Espacio, 2005-2008
- IADIRA, "Inertial Aiding – Deeply Integrated Receiver Architecture", FP7-GALILEO-2007-GSA-1, 2009-2010
- IEGLO. "Infrastructure-Augmented EGNOS/Galileo Receiver for personal Mobility", FP7-GALILEO-2007-GSA-1, 2009-2010
- CLOSE-SEARCH. "Accurate and safe EGNOS-SoL navigation for UAV-based low-cost SAR operations", FP7-GALILEO-2008-GSA-1, 2010-2011
- ATENEA. "Advanced Techniques for Navigation Receivers and Applications", FP7-GALILEO-2008-GSA-1, 2010-2011
- GENEVA. "Galileo / EGNOS Enhanced Driver Assistance",FP7-GALILEO-2008-GSA-1, 2010-2012
- ENCORE. "Enhanced Code Galileo Receiver for land management in Brazil".FP7-GALILEO-2008-GSA-1.2010-2012
- ATENEA +. "Prototipo de receptor avanzado multisensor GNSS-INS-LiDAR-Cámara para cartografía urbana", INNPACTO, 2011-2013
- DECIVEL. "Determinació Cinemàtica de Vies i Estructures Lineals", PROVA'T, 2012-2013
- GAL. "Galileo for Gravity", FP7-GALILEO-2011-GSA-1, 2012-2014
- GINSEC. "Enhanced GNSS-BF-INS Solution for Un-manned Vehicle Control", SME-2013-1, 2013-2016

Other research works

- Wis, M. "Dynamic stochastic modeling for inertial sensors." PhD thesis. UPC 2016.
- Tèrmens, A. "A network approach for strapdown inertial kinematic gravimetry." PhD thesis. UPC 2014.

International Journal of Geo-Information

MDPI

Article

ASTROLABE: A Rigorous, Geodetic-Oriented Data Model for Trajectory Determination Systems

José A. Navarro [1,*], M. Eulàlia Parés [1,*], Ismael Colomina [2] and Marta Blázquez [2]

[1] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Parc Mediterrani de la Tecnologia, Av. Carl Friedrich Gauss 7, Building B4, 08860 Castelldefels, Spain

[2] GeoNumerics S. L., Parc Mediterrani de la Tecnologia, Av. Carl Friedrich Gauss 11, 08860 Castelldefels, Spain; ismael.colomina@geonumerics.com (I.C.); marta.blazquez@geonumerics.com (M.B.)

* Correspondence: jose.navarro@cttc.es (J.A.N.); eulalia.pares@cttc.es (M.E.P.); Tel.: +34-93-645-29-00 (J.A.N. & M.E.P.)

Academic Editor: Wolfgang Kainz
Received: 20 December 2016; Accepted: 25 March 2017; Published: 28 March 2017

Abstract: The constant irruption of new sensors is a challenge for software systems that do not rely on generic data models able to manage change or innovation. Several data modeling standards exist. Some of these address the problem from a generic perspective but are far too complex for the kind of applications targeted by this work, while others focus strictly on specific kinds of sensors. These approaches pose a problem for the maintainability of software systems dealing with sensor data. This work presents ASTROLABE, a generic and extensible data model specifically devised for trajectory determination systems working with sensors whose error distributions may be fully modeled using means and covariance matrices. A data model relying on four fundamental entities (observation, state, instrument, mathematical model) and related metadata is described; two compliant specifications (for file storage and network communications) are presented; a portable C++ library implementing these specifications is also briefly introduced. ASTROLABE, integrated in CTTC's trajectory determination system NAVEGA, has been extensively used since 2009 in research and production (real-life) projects, coping successfully with a significant variety of sensors. Such experience helped to improve the data model and validate its suitability for the target problem. The authors are considering putting ASTROLABE in the public domain.

Keywords: trajectory determination systems; data model; genericity; extensibility

1. Introduction

In the context of this paper, Trajectory Determination Systems (TDS) [1] are software components that compute *trajectories* using a variety of sensor measurements as input. The word trajectory usually refers to the path of a flying object. In this article, we will first generalize its use to the path of any moving object through the 3D space as a function of time. In addition, we further generalize it to paths of entities—objects or mathematical abstractions—whose coordinates change with time in some *navigation*, *state* or *parameter* space N; e.g., the actual 3D path plus the velocity and attitude angles of a 3D rigid body. Clearly, this concept is closely related to the mathematical one of a stochastic process since we describe the coordinates of the object as a time-dependent N-valued random variable. More to the point, a TDS estimates optimal—in some sense—trajectories from sensor measurements and mathematical models. A TDS may be either a real-time or post-processing tool.

Today, the applications of trajectory determination are becoming increasingly relevant and demanding: unmanned aircraft navigation [2], precision agriculture [3,4], indoor mapping, indoor positioning [5], robotics—including autonomous vehicle driving— [6] are blooming examples among many others. Moreover, the trend is not bounded by the traditional relative small size of the

professional markets since accurate trajectory determination is also an enabler of mass market applications [7]. In general, these applications cannot be served by mono-sensor systems due to the complexity and variety of the navigation environments. In parallel, sensing technology—including satellite navigation technology—is evolving fast and contributing new sensors either in their very concept or in their features (size, weight, error properties and other) [8,9]. To leverage their potential for trajectory determination in the context of extremely competitive markets, the measurements of new sensors shall be modelled and integrated in TDS as quickly as possible.

The challenges discussed above are not only scientific, but also software engineering ones, both at internal computational and external interface levels. The described situation, large variety and rapid evolution of input data, is not particular to TDS but a general condition of modern software systems. In fact, coping with change and evolution of software systems is an old challenge of software engineering. Many software development methods like Extreme Programming [10] or, more in general, the family of agile software development methods [11] were, to a significant extent, motivated and influenced by the need to cope with unforeseeable change and rapid evolution. Many programming languages were also shaped by these challenges, one of the most popular being object-oriented languages [12].

One solution to the problem is to develop systems—TDS in particular—that are easily extensible. Over the past years we have developed a number of TDS [13–15] that achieved extensibility through genericity. For this purpose, we designed internal estimation engines and external interface engines, for a wide class of data and mathematical models in a way that dealing with new data types or new models neither requires to modify the estimation engines nor the input/output ones [14,15], following a plug-and-play paradigm. As opposed to the simultaneous approach in [13,14], ASTROLABE targets sequential estimation and is based on seed concepts anticipated in [15]. More specifically, under "wide class" we understand an abstraction that supports, in principle, all existing and future sensor models and data for TDS. A similar approach is reported in [16] for a "Reconfigurable Integration Filter Engine" (RIFE). While [16] describes a generic TDS, in ASTROLABE we concentrate on a generic interface independent of the internal navigation engine and target all types of trajectories. A related article [17] mentions the Unified Aiding Information Drives (UAID) generic sensor interface for RIFE. However, in [16] the use of a new sensor requires that the sensor type is available in a system library whereas we pursue a different level of abstraction where new types of sensors can be also integrated in a plug-and-play fashion.

We note that there are other approaches to the extensibility of TDS. One possibility is to develop, for each sensor type, a "maximalist" model and interface with the hope that it is general enough. Another approach is to use sensor replacement models (RPM) [18]. This technique is popular in small-scale photogrammetry and remote sensing. It is based on general analytical formulations, usually unrelated to the physics of the sensor, like rational functions (algebraic fractions where both the numerator and the denominator are polynomials). Its benefits notwithstanding, a RPM is nothing else than a model that works for a wide variety of imaging sensors. A true generic interface shall allow for a seamless use of any type of model and entirely leave to the decision of the sensor specialists which particular model shall be used.

As with software engineering, extensibility is an old issue in geomatics with some early contributions that had influenced our work: in [19] early attempts to classify geodetic entities under a common umbrella were made; in [20] a pioneering generic adjustment engine was presented; in [21] the advantages of the object-oriented approach to generic adjustment systems were discussed; and in [22,23] the underlying general structure of network adjustment systems was analysed. Though loosely related to our problem, we acknowledge the research on ISO standards for geomatics as reported in [24] that have the potential to generate generic interfaces for TDS. These are contributions from geomatics relevant to TDS as they equally apply to real-time and post-processing tools.

Generally speaking and surprisingly to some extent, the literature on navigation and orientation, either in broad general surveys like [25] or even for specific generic systems like [26,27] or [28] do not

tackle generic interfaces. It has been only recently, that the design of plug-and-play systems has made apparent the need for generic interfaces; automation—e.g., in robotics—and harmonisation—e.g., large organisations—are drivers of this. Thus, as already referenced, [17] mentions the UAID generic sensor interface for their universal plug-and-play navigation system. [29] refers to the Defense Advanced Research Projects Agency (DARPA) All-Source Positioning and Navigation (ASPN) programme and its set of generic interface documents known as the "ASPN Interface Control Documents (ICDs)". ASPN was launched by DARPA in November 2010 with the aim of developing low cost sensor fusion technologies and achieving a plug-and-play architecture; i.e., that sensors can be added, removed, or replaced on-the-fly and that their measurements can be included in the input data streams with minimal delay [30]. The DARPA initiative is an independent confirmation of the relevance of the topic of this article.

To at least alleviate this problem, this research work proposes a data abstraction. The identification of the common traits defining the essence of the information provided by sensors leads to a data model general enough to support such diversity, that is, able to represent any kind of sensor whose error distributions may be fully modelled by means of a mean and a covariance matrix (see Section 2) including metadata. Data (values) by themselves are meaningless; a rigorous data model must take care of including the metadata that unequivocally characterize data if it means to be complete. Ignoring metadata has been a constant source of problems in the past—and it still is [31]. Unfortunately, applications in production environments regularly tend to ignore metadata for the sake of performance (The absence of metadata implies a series of assumptions on default values concerning data, as the physical units or the expected reference frame and coordinate system used, among others. As a consequence, it should not be necessary to perform conversion tasks (units, coordinate systems) on input data, thus reducing the computational effort required to provide a solution and, consequently, improving performance.).

Standards covering sensor and observation data models already exist [32,33] and efforts to bring those closer to geomatics also exist [24]. Although these data models would have been powerful (and general and extensible) enough for the purposes of our work, we focus, however, *in a much narrower field of applications* than those targeted by the aforementioned standards; a suitable data model for TDSs needs to specify a fewer number of characteristics to be sufficient, which leads to a more concise specification. In short, ASTROLABE may be seen as a subset of [32,33]. Additionally, terseness is an advantage from several standpoints, as for instance: (1) the amount of information to store when a TDS is working in real time is noticeably reduced, thus decreasing the time needed to save data; (2) transmitting such data through a network connection reduces both bandwidth requirements and transfer time; (3) the burden related to the preparation and management of data and metadata is significantly reduced in actual production environments and (4) the implementation of an Application Programming Interface (API) managing such data model is much simpler and compact.

On the other side, standards designed to represent data from specific types of sensors exist and are widely used. A well known example is the Receiver INdependent EXchange (RINEX) format [34], specifically designed for GNSS range measurements, and its relatives, the Antenna Exchange (ANTEX) format, the IONosphere map EXchange (IONEX) format and the Standard Product 3 (SP3) orbit format. We note that each of these formats are defined independently and that the inclusion of new GNSS systems requires modification and recompilation of the IO software in most cases. Another popular example is the LASer file format (LAS) [35] for point clouds derived from laser scanning. The main advantage of such kinds of standards, being tailored and optimized for specific sensors, is, at the same time, the reason to avoid them; one of the goals of the research work presented in this paper is to devise a generic data model able to manage diversity and evolution.

This paper presents ASTROLABE, developed since 2009 at the former Institute of Geomatics (IG) and since 2014 at the Geomatics Division of the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) with the cooperation of GeoNumerics. Its goal is to create a generic, extensible, complete and efficient data model suitable especifically for TDSs. This work is inspired in the generic data models

and interface specifications of two generic and extensible network adjustment tools, GeoTeX [13] and GENA [14]. At the time when the decision to develop the generic standard was made, to the best knowledge of the authors, there was nothing comparable, academic or industrial. It is true that the standards defined in [32,33] already existed but, as stated above, these were too ambitious for our purposes.

ASTROLABE is the generic interface of CTTC's NAVEGA [15] and GeoNumerics' NEXA trajectory determination systems. In the rest of the article, for the sake of clarity and readability, we will use NAVEGA examples and refer to NAVEGA functions whenever helpful.

This work is structured as follows: Section 2 identifies and introduces the data entities involved in the process of trajectory estimation; these data entities are the pillars on which the ASTROLABE data model relies. Section 3 describes such data model in full detail from the conceptual standpoint, including data and metadata. On Section 4 the ASTROLABE data model is translated into a formal specification comprising a file and a network interfaces, which are briefly described. An almost complete C++ library implementing these two interfaces is briefly discussed in Section 5. Some considerations on the kind of sensors (and related observations) that may be modeled using ASTROLABE are included in Section 6. To finish, conclusions are presented in Section 7. The interested reader may find a more detailed description of the ASTROLABE file interface in Appendix A. An example describing ASTROLABE data and metadata used in a real-life project (GAL) may be found in Appendix B.

2. The Foundations of the Data Model

Shortly stated, the goal of a TDS is to derive trajectories out of sensor data, whatever these are, be it in post-processing or real-time mode.

As any other kind of application, a TDS transforms input data into some kind of result using specialized logic appropriate for the problem to solve. In the particular case of TDSs, these use *observations* coming from *instruments* (sensors) as input, and apply a series of *mathematical models* specific for the intervening observations and instruments to compute the *states*. The computation of these values in consecutive points of time deliver a time-ordered series of states, the trajectory.

The most common methods used to determine trajectories in TDSs are sequential least-squares estimators. Kalman filters [36] are the best known among them. Although these filters do not specifically require Gaussian distributions for errors, they yield the exact conditional probability estimate when this happens. Further, to fix the ideas, for the data model described hereafter, a Gaussian distribution of the observation error will be assumed. Nevertheless, the data model can be used for non-Gaussian situations. In fact, the proposed data model supports a broad range of error probability distribution functions, namely those for which the first and second moments exist.

Four relevant entities arise from the previous description: states (a.k.a. parameters), observations (a.k.a. measurements), instruments and mathematical models. The data model is precisely based on them. In our context, it is convenient to add a "new" one, namely, the *observation equation*, intended to identify a particular combination of observations, states, instruments and mathematical models.

Below, a short formal description of these entities is provided to help to focus the discussion.

- **(Observable and) Observation.** An observable is a numerical property of a physical system that can be determined by a sequence of physical or mathematical operations. Technically, it is a random variable. An observation or measurement is one of the values that an observable or random variable may take; i.e., it is a random sample of a random variable. (For example, the various repeated measurements between A and B of a distance meter instrument—the measured distances—are observations and the abstract concept of distance between A and B is the observable).
  It is worth mentioning that for ASTROLABE, and this is a language abuse, an observation is, in fact, a vector of measurements for a set of observables (an array of observations as defined just above).

In the context of TDSs, typical observables are acceleration, temperature, range or angular speed. Observations for these observables might be -1.3 m/s^2, $+23.556°$, $+12{,}832.34$ m and -27.42 rad/s respectively.

- **State.** Similar to observations, a state is a time-dependent random vector whose expectation and covariance have to be estimated from known observations, instruments and mathematical models. Note that in simultaneous, least-squares estimation the states are called parameters.
  Typical states in TDS environments are position or attitude random vectors.
- **Instrument.** An instrument (or sensor) is a device used to measure some observable, thus delivering observations. From the data abstraction point of view, it is an entity which contains the constants that characterizes an actual instrument.
  Within the context of TDSs, inertial measuring units, gyroscopes or GNSS receivers are examples, among others, of instruments.
- **(Mathematical) Model.** A mathematical description of a system or process. In the context of ASTROLABE, it is a stochastic equation or a stochastic differential equation; traditionally, this has also been described as a functional model plus an stochastic model.

The last data entity in ASTROLABE's data model is the observation equation which, in our context, is defined as:

- **Observation equation.** A materialization of a given mathematical model—stochastic or stochastic differential equation— for a particular set of measurements, instruments or states.

The last part of the former sentence introduces a new facet in the definition of observation equations. From the processing standpoint these may be viewed as *commands* or *triggers* that tell the trajectory determination software that the right moment to derive a new state has come (and that the information to use is the one specified by such equation). A series of observations, all of them including measurements reported by—probably—different, cooperating sensors, may be held in stock until the best moment to derive the new state comes. This moment is reported by the observation equation. This is specially important when working with systems that hybridize a set of sensors to compute trajectories; some usual examples combine, for instance, GNSS receivers, inertial measuring units and magnetometers for such purpose. The use of the simultaneous data reported by these sensors noticeably increases the quality of the solution (see [37]).

Figure 1 depicts graphically an hypothetical observation equation. This example assumes that there exist j different kinds of models, l kinds of observations, o different kinds of states and z kinds of instruments. The observation equation in the figure relates the model whose identifier is h, two observations with identifiers 1 and k respectively, two states, whose identifiers are m and o and, finally, a single instrument with identifier y.

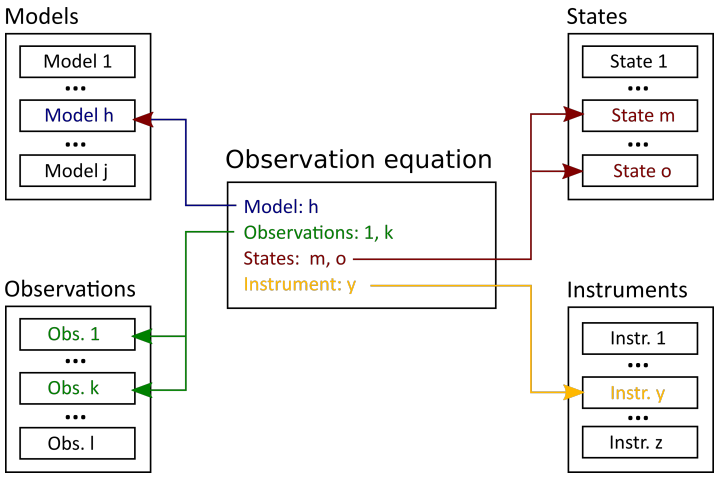

Figure 1. Example of an observation equation.

3. The Data Model in Detail

The sections to follow will describe in detail the observation, state, instrument and observation equation entities just introduced. Such a description takes place from the data and metadata standpoints; it is the result of the abstraction process that leads to the ASTROLABE data model. However, before such a description may be detailed, it is necessary to discuss how data and metadata are identified and cross-referenced. Section 3.1 will introduce the [meta]data identification mechanism used in ASTROLABE. Sections 3.2–3.5 will then describe both data and metadata.

3.1. On Data Identification and Cross-Referencing: Types, Identifiers and Instance Identifiers

ASTROLABE relies on a hierarchical, three-level mechanism to unequivocally identify and cross-reference data. Metadata employ two of these levels, namely types and identifiers, while data complete the identification triplet incorporating instance identifiers.

A type is a code tagging a unique kind of object (be it an observation, state, instrument or model) with a specific, and also unique, set of properties. An example of such object would be a certain kind of instrument, e.g., a temperature-compensated barometer. A unique type code would be assigned to this kind of barometer. Should a different kind of barometer exist, as for instance, one whose measurements were not compensated by means of temperature, a new, different type code should be used. Examples of these type codes could be `barometer_t_compensated` and `barometer_basic` respectively.

There may exist, nonetheless, many brands and models of barometers that use temperature to correct their measurements. All these would be incarnations of `barometer_t_compensated` (instrument) objects, that, in spite of sharing a common type, might behave differently—e.g., being more or less accurate or delivering data using different physical units. In other words, even though all the temperature-compensated barometers may be described by a common set of properties, the actual values of these properties may differ.

The identifier is used to take these differences into account. An identifier is a unique code used to tell apart different incarnations of objects (temperature-compensated barometers in the example above) that, in spite of being characterized by the same set of properties have different values for these.

A tuple composed by a type and an identifier unequivocally identifies any kind of object in ASTROLABE metadata. Figure A2, lines 01–29, shows the full XML specification of temperature-compensated barometer observations using the ASTROLABE file interface (Although this section takes care of describing a data model that may be implemented in many different ways, examples are presented using the XML syntax defined by ASTROLABE to materialize its file interface. Such file interface is detailed in Section 4.1.). Note the type/identifier codes in lines 02 and 04 of this example.

Actual data records use the aforementioned identifier to characterize themselves. That is, these records include the identifier as an extra field pointing to the metadata that will serve to describe them. Figure 2 depicts the type/identifier hierarchy and shows as well how actual data use the identifier code to state what kind of information is stored in data records.

The identification schema just depicted is not complete yet from the data (not metadata) standpoint, since it does not take into account that multiple, identical instruments might be used simultaneously when collecting data. That is, data originating from two or more identical instruments—sharing the same type and identifier codes—may be present in a dataset. To solve this problem, the third element in ASTROLABE's data identification schema is used. It is the instance identifier.

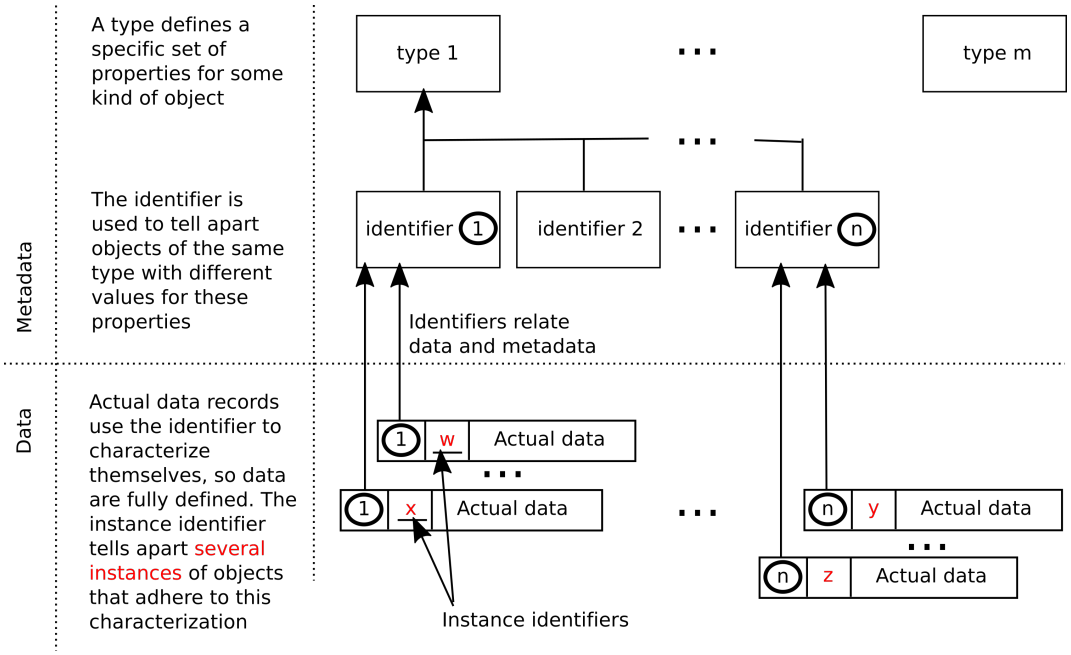

Figure 2. Types, identifier and instance identifier. Cross-referencing from actual data.

The instance identifier is a code used to distinguish between several instances of objects whose types and codes are identical. It purpose is to allow multiplicity.

Thus, actual data records use a combination of an identifier plus an instance identifier to fully characterize themselves. The identifier points to metadata (and therefore, to the exhaustive description of the information the data record contains) while the instance identifier is used to discriminate between several instances of identical data sources. Figure 2 depicts graphically this situation. Figure 3 shows several actual barometer data records in ASTROLABE XML format including the same metadata identifier (baro1, see Figure A2) but two different instance identifiers (1 and 2). That is, data from two identical barometers have been collected at times 124.88 to 124.92 (two readings from barometer 1, only one from barometer 2).

Note that the values of the instance identifiers may be chosen arbitrarily, providing these are different for every object instance they represent.

```
<I id="baro1" n="1"> 124.88 23.44 1023.442 0.3 </I>
<I id="baro1" n="2"> 124.90 23.45 1023.450 0.3 </I>
<I id="baro1" n="1"> 124.92 23.45 1023.443 0.3 </I>
```

Figure 3. Data: identifier and instance identifiers in barometer data.

3.2. Data: Observations

Observations are one of the main pivots the ASTROLABE data model turns around; not in vain, observations are the primary source of data used to derive results, either in TDS or in many other kind of software tools. In fact, many of these tools consider observations as the unique source of information—often forgetting metadata completely. Therefore, finding a powerful abstraction for observations is of capital importance for a data model that means to be generic and extensible.

The type of information provided by sensors is—apparently—very different. It is evident that a barometer will not deliver the same kind of data as with a spectrometer, a gravimeter or a densitometer, to mention some. Moreover, and now moving to the software realm, the algorithms used to deliver useful results out of sensor readings are different too. Taking into account, however, how information

is processed is not the task of a data model; it must deal only with those aspects that are exclusively intrinsic to data.

In spite of the aforementioned dissimilarities in observations, a common structure, their essence, may be identified in all of them. The observations that may be modelled by ASTROLABE consists of a set of expectations (The word expectation in the context of this paper stands for *the best estimate of the expectation of the random variable associated to the measurement*.) (the measurements reported by the device) and an assessment of its quality (the covariance matrix (This is so because ASTROLABE assumes observations with error distributions fully defined by the first and second moments; therefore, an observation always consist of an array of expectations plus a covariance matrix.)). No matter what the observation is, these elements will always be present (Some sensors may not provide actual values for the covariance matrices that should accompany every measurement. In such situations the nominal quality values reported by the manufacturer may be used. Such default values may be specified by means of metadata. For instance, the example in Figure A2 (line 18) provides a default value (1 hPa) for the standard deviation component of the covariance matrix by means of the <c> tag.).

Of course, the number of values present in each kind of observations will vary; a 3-axial accelerometer will report three readings (acceleration around the three axes) while a thermometer will provide just one (the temperature). Even when considering sensors of the same type providing the same number of readings, the units used to deliver such values may vary: m/s^2 or cm/s^2 in the case of accelerometers, for example.

These differences are, however, accidental and may easily modeled through the proper use of metadata (see Section 3.5). What remains is the fact that every observation will be composed of a set of expectations and the related covariance matrix.

Therefore, the first approach to define the common structure for observations would state that it is composed of two elements:

observation: (expectations, covariance matrix)

It is necessary to be able to tell apart different kinds of observations; otherwise, it would be impossible to interpret the semantics of these measurements: observations from a distancemeter would be indistinguishable from those coming from an odometer. To do so, the identifier and instance identifiers defined in Section 3.1 are introduced:

observation: (**identifier**, **instance identifier**, expectations, covariance matrix) (Note that the **boldfaced** words in definitions are used to denote changes with respect to previous ones.)

The unique identifier will point to the appropriate metadata, where aspects as the dimension of the vector of expectations or the units used—as well as other accidental properties—will be specified. The instance identifier, as stated in Section 3.1, will serve to tell apart observations originating from different instances of equivalent data sources (as for instance, two identical barometers).

Note that the need to identify different kinds of observations responds to the (geodetic) principle of heterogeneity (Heterogeneity refers to the use of different types of observables for the determination of the same set or subset of parameters or states. Better systematic error correction is also the goal of heterogeneity.) on which the ASTROLABE data model relies. The need to identify several instances of the same kind of observation responds to the (geodetic) principle of redundancy (Redundancy refers to the repetition of measurements, apparently superfluous and unneeded, with the objective of mitigating the random dispersion of the observables (random variables) being measured (samples of the observable). Better outlier detection is also the goal of redundancy.).

Time is a **requirement** in TDSs because of the nature of their purpose. Observations must therefore be time-tagged to register the moment when the sensor delivered its readings.

observation: (identifier, instance identifier, **time**, expectations, covariance matrix)

Sometimes, it is interesting to keep track of extra data that, although not an intrinsic component of an observation, helps to complement it. This auxiliary data may be of any kind. For instance, an accelerometer delivering acceleration measurements may be seriously affected by temperature. This observable (temperature) is not, from the conceptual standpoint, part of the acceleration observation; however, ignoring it may lead to incorrect results when used by a TDS. On the contrary, extending the observation with this information may help the TDS to, for instance, apply a calibration method to correct the data coming from the accelerometer.

The former is just an example to illustrate why auxiliary information should be a companion of pure observation data. Auxiliary values are called *tags* in the context of ASTROLABE. The number of tags is arbitrary; other accelerometers, for instance, might be affected by other kinds of observables. Other sensors may need not auxiliary values at all.

Therefore, the ASTROLABE data model makes tags an integral part of its observation data entity; the number and properties of tags—that may be zero—for each kind of observation is defined by means of metadata (see <t_spec>, Figure A2, lines 20–28).

observation: (identifier, instance identifier, time, **tags**, expectations, covariance matrix)

Leaving aside metadata, and from the conceptual—ASTROLABE's, at least—standpoint, there is nothing else that an observation should include. There are, however, two more items that the current observation model has adopted: an extra event tag and an activation flag.

The typical dataset used in trajectory determination (either a file or a network stream, for instance) will include *at least* two kinds of data entities: observations and observation equations. Since both data entities will be merged in datasets, a mechanism to tell apart these entities is necessary. A simple marker identifying the kind of entity it precedes will suffice for that purpose. ASTROLABE names this marker as event tag.

The final element composing the observation model is provided just for (processing) convenience purposes: it is the activation flag. When processing data, some observations may be detected as wrong ones (for instance, because a magnetometer starts producing invalid readings when approaching a powerful source of electromagnetic interferences, as power lines). If these observations are not removed from the computation of the output trajectory the results will be distorted. Of course, it is possible to eliminate such observations from the input dataset thus removing their harmful effects; however, this distorts how data was captured, since the original observation will no longer be available in the input dataset.

The solution to this—apparent—problem is to provide with the aforementioned activation flag. An observation set to "inactive" by the human operator must be ignored by any TDS just as it would have never existed.

This leads to the final step in the definition of the observation entity data model as seen by ASTROLABE:

observation: (**event tag**, **activation flag**, identifier, instance identifier, time, tags, expectations, covariance matrix)

Table 1 summarizes the discussion above.

Table 1. The observation entity data model.

Item	Description
Event tag	Identifies the kind of data entity (observation).
Activation flag	Enables or disables the observation for processing purposes.
Identifier	Defines the kind of observation (points to metadata).
Instance identifier	Tells apart multiple occurrences of identical data sources.
Time	Time stamp; time when the measurements were made.
Tags	Data that is not part of the observation but complements it.
Expectations	The observed measurements.
Covariance matrix	Estimated error of the expectations.

Figure 4 shows two examples of observations using the ASTROLABE XML syntax. The metadata describing these of observations may be found in Figure A2. Section 3.5 describes metadata in detail.

It is possible to know that these are observations because of the <l> tag opening (and closing) data records. This XML tag corresponds to the event tag described above. The active flag is represented as the "s" (status) attribute, which may take two values: "a" for active or "r" meaning removed or inactive. Both observations include their identifiers (pointing to the metadata that characterize them) in the "id" attribute. The first observation states that its identifier is "baro1" while the second one identifies itself as "imu1". To link these observations to specific instances of sensors the instance identifier, represented by the "n" attribute, is used, whose values, for the "baro1" and "imu1" observations are, respectively, 32 and 41.

The next field is the time stamp (124.88 in both observations). In the case of the "baro1/32" observation, one tag (auxiliary value) has been included (23.44); this is so because the sensor related to this observation is a temperature-compensated barometer; the tag corresponds to the temperature reading at the moment when atmospheric pressure was measured. There are no tags for the "imu1/41" observation.

Then, the measurements themselves are included. In the case of "obs1/32" a single value (pressure, 1023.44) is provided. The "imu1/41" observation, on the contrary, provides the readings for three angular velocities and accelerations (0.01 0.02 0.015 0.32 0.43 9.95).

Lines 2 and 5 provide the covariance matrices (a single standard deviation in the case of "obs1/32") for these observations respectively.

See Figure A1 for a complete ASTROLABE XML example showing both observations and observation equations.

```
1 <l s="a" id="baro1" n="32"> 124.88 23.44 1023.44
2                                        0.3                              </l>
3
4 <l s="r" id="imu1"  n="41"> 124.88       0.01 0.02 0.015 0.32 0.43 9.95
5                                        1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
```

Figure 4. Data: examples of observations in XML syntax.

3.3. Data: States, Instruments and Mathematical Models

States and instrument data may be modelled using exactly the same structure just discussed in Section 3.2, at least in the context of ASTROLABE and its goals (In a trajectory determination and, in general, in a parameter or state estimation "ecosystem" the role of measurements, states and instruments uses to commute. In other words and more specifically, an instrument—i.e., its calibration parameters, time varying or not—could be the result of a trajectory determination exercise. However, in a next software run, an instrument calibration set of states can be seen as an instrument constant. Or for example, a GNSS-derived position can be a state of a GNSS trajectory determination based

on GNSS measurements. However, in a next INS/GNSS loosely-coupled trajectory determination run, those position states become the position observations. In ASTROLABE a trade off between the engineering model and the mathematical model is made. We could refer to the measurements, states and instruments as known or unknown stochastic processes. However, by telling apart the measurements, from the states, from the instruments we facilitate the use of the standard and the modelling process. This is why, mathematically and structurally, observations, states and instruments are alike.). The following are the definitions for state and instrument data (which have been provided for the sake of completeness, since these are exactly the same as the definition of an observation:)

state: (event tag, activation flag, identifier, instance identifier, time, tags, expectations, covariance matrix)

instrument: (event tag, activation flag, identifier, instance identifier, time, tags, expectations, covariance matrix)

In the case of observations and states, the coincidence of their definitions should not surprise the reader; in fact, when working in post-processing environments, TDSs use to compute trajectories in three steps, the so-called forward, backwards and smoothing ones. The output (trajectories, made of states) produced by the two first steps become the input of the third one. *The role of states is thus changed to observations at that point.*

Observations and states may easily be told apart by the context in which these both entities appear. Observations will always be included in input files/network streams. States are always the output of TDSs, so these will always be found in output files/network streams. For an example of states written in the ASTROLABE XML syntax, the reader may refer to Figure 4, which, in fact, depicts observations.

Instruments, on the other side, are not random variables. That is, time does not affect instrument data, which is considered constant information. From the structural standpoint, instrument data is, once more, a set of values and their quality information (that is, the expectations and covariance matrices found in observations and states). Note that, in the case of instrument (constant) data, the covariance is considered by ASTROLABE as a mere indication of the quality of the list of values.

Since instruments must also be identified, the three level hierarchy used in observations and states is adopted here. Tags are provided for the same reason as with observation data; the difference, in the case of instrument constants, is that these tags are purely informative, and contain their values at the moment the aforementioned constants were computed—that is, when the instrument was calibrated.

The use of time stamps and activation flags as an integral part of the instrument data needs some justification. Although purely informative, time stamps are used to keep track of the time when the instrument constants were calculated. This is very important in the case of unstable instruments that must be recalibrated often; an explicit calibration time helps to avoid problems due to the use of outdated information. Note, finally, that the time stamp must use the same reference frame and coordinate system that those used in observation records.

The presence of the activation flag might seem controversial: if observations from some kind of instrument are included in a dataset, the constants defining the instrument itself should be included as well. On the contrary, if the observations contains no data from some kind of instrument, the inclusion of the instrument data themselves would be superfluous. Therefore, the possibility of activating or deactivating instruments seems useless. This is true; however, this flag is included for convenience, purely practical reasons: it allows storing the constants of several instruments in the same single file. Depending on the sensors (and therefore, observations) used to compute a particular trajectory, the instruments available in such file may be activated or deactivated correspondingly, and the unique instruments file reused easily.

Typical examples of instruments constants would be the focal length of a camera or the position of the center of a GNSS antenna.

The following are two instrument data records that adhere to the definition above but include no tags. The instrument is described in the metadata file fragment shown in Figure A3, more specifically in lines 138–171.

```
<l id="p0_h0" n="51"> 124.88 1024.01 0.0 </l>
<l id="p0_h0" n="52"> 124.88 1023.99 0.0 </l>
```

These records show the expected pressure readings (1024.01 and 1023.99 mBar, see the <units> tag in Figure A3, line 155) for two (instance identifiers 51 and 52 respectively) identical (id = "p0_h0" in both cases) barometers, while their heights are 0.0 meters (the units are defined in line 166 of Figure A3). The value of the time tag (informational only) is 124.88 in both records. No covariance matrices have been included; should these be necessary (just for informational purposes only) then default values for such matrices may be retrieved from the corresponding metadata (<c> tags in lines 156 and 167 in Figure A3).

Concerning mathematical models, there are no data for these; in fact, ASTROLABE models are mathematical equations implemented by TDSs. The only information related to mathematical models that is included in the data model is their metadata (see Section 3.5 and Figure A4).

3.4. Data: Observation Equations

As already discussed in Section 2 and shown in Figure 1, the observation equation relates all the intervening data entities needed to derive a new output state (models, observations, states and instruments).

The first attempt to define the data model for observation equations directly mirrors this definition:

observation equation: (model identifier, list of observation instance identifiers, list of state instance identifiers, list of instrument instance identifiers)

Note that the identifiers used to refer to states, observations and instruments are instance identifiers (see Section 3.1). This means that if data coming from different instances of identical sensors are available, it is possible to refer individually to each of these sensors in the observation equation.

For instance, assuming an observation equation relating one model to two observations and one state whose respective identifiers were compute_position (the model), GPS and IMU (the observations) and position (the state), the observation equation would roughly look like this:

(compute_position, GPS + IMU, position, -) (The dash at the end of the equation stands for "no instrument identifiers". This example equation includes no instruments.)

If two identical GPS receivers were used to collect data, using directly the metadata identifiers to characterize the observation equation records would not leave room for multiplicity. On the contrary, the use of instance identifiers solves this problem. Assuming that the instance identifiers are 20 and 21 (GPS receivers 1 and 2) and 30 (IMU) and 10 (state), the following observation equations would involve, respectively, the first and second GPS receiver:

(compute_position, 20 + 30, 10, -)
(compute_position, 21 + 30, 10, -)

As seen in the example above, the cardinality of the different lists of instance identifiers may be zero in some cases. The model identifier must, however, be always present.

Extra items must be added, as it happens to observation, states and instruments themselves. An event tag, an activation flag and a time stamp must be added to complete the data model for this entity. Their purpose is described in Section 3.2—although that section talks about observations, the explanations given there may be applied here so these are not repeated again. In this context, however, the activation flag may be interpreted as a quick and practical way to avoid the (erroneous) computation of states when a significant number of the observations involved are clearly wrong.

observation equation: (**event tag**, **activation flag**, **time**, model identifier, list of observation instance identifiers, list of state instance identifiers, list of instrument instance identifiers)

Table 2 briefly describes all the elements integrating an observation equation.

Table 2. The observation equation entity data model.

Item	Description
Event tag	Identifies the kind of data entity (observation equation).
Activation flag	Enables or disables the observation equation for processing purposes.
Time	Time stamp for the equation.
Model identifier	Points to the model to use to deliver new states.
Observation instance ids.	List of observations involved in the computation of new states.
State instance ids.	List of states involved in the computation of new states.
Instrument instance ids.	List of instruments involved in the computation of new states.

The explicit inclusion of the model identifier opens the way to the use of different kinds of models in structurally identical observation equations. Revisiting the example above, the model used to derive the new state may be changed while involving exactly the same lists of observations and states:

(compute_position_**method_1**, 20 + 30, 10, -)
(compute_position_**method_2**, 20 + 30, 10, -)
...
(compute_position_**method_x**, 20 + 30, 10, -)

One positive consequence of this feature is that researchers may tests new algorithms simply changing the model identifier used to derive new states in observation equations.

Figure 5 includes three examples of observation equations in ASTROLABE XML syntax. All these may be easily identified by the opening tag <o> which represents the observation equation event tag. The activation flag is materialized by means of the "s" attribute (whose values, "a" and "r" stand for "active" and "inactive" or "removed" respectively). Note that the equation at line 3 has no explicit activation tag: it is assumed active by default.

The observation equation identifiers (attribute "id") point to the respective models involved in the equation, namely "pva1_d", "imu1_bias_d" and "height_update". After these, the time tag (124.88 in all cases) and the lists of observation, state and instrument instance identifiers are included. The metadata for these equations may be found in the example in Figure A4 using the model identifiers (id) shown in Figure 5. (ASTROLABE metadata are discussed in Section 3.5). Such metadata state that, for example, model "height_update" points to *one* observation whose identifier is "baro1", *two* states ("baro1" and "pva1") and just *one* instrument, ("p0_h0"). This implies that a total of *four* instance identifiers must be present in the observation equation, as shown in line 3 of Figure 5: instance identifier 27 for the observation, 34 and 32 for the states and 51 for the instrument.

See Figure A1 for a complete ASTROLABE XML example showing both observations and observation equations.

```
1 <o s="a" id="pva1_d"       > 124.88 27 11 41    </o>
2 <o s="r" id="imu1_bias_d"  > 124.88 11    17    </o>
3 <o       id="height_update"> 124.88 27 34 32 51 </o>
```

Figure 5. Data: examples of observation equations in XML syntax.

3.5. Metadata

Sections 3.2–3.4 describe the structure of the different data entities involved in ASTROLABE's data model. For this definition to be rigorous, some additional aspects related to these entities must be specified. These aspects constitute the metadata.

Observations and states share the same metadata structure, that is, the same kind of facets are specified. For this reason, these will be described below within a common frame. Instruments and models, on the contrary, are a case apart, so their metadata will be specified separately.

The main items constituting the metadata for **observations and states** are the following:

- **Type and identifier.** See Section 3.1 for a detailed explanation on type and identifier codes.
- **Toolbox.** Name of the software module—typically, a Dynamic Link Library (DLL, Windows environment) or Shared Library (Linux environment)—including an implementation of the logic related to the observation or state. The way this name is used by the underlying operating system to find the library is not defined by ASTROLABE.
- **Dimension.** Number of elements in the observation/states expectations vector.
- **Referencing.** The reference frame plus coordinate system to which data refer to is specified here. Alternatively, it is possible to define a coordinate reference frame instead.
- **Units.** Specifies the units of the individual elements of the expectations vector for observations or states.
- **Covariance matrix.** Default covariance matrix for the expectations vector for observations and states. The units of the covariance matrix are the same as those provided for the expectations vector, even in the default case. ASTROLABE accepts "reduced" or "full" covariance matrices. A reduced covariance matrix consist of just standard deviations—assuming zeros for correlation values. This is an optional field.
- **Scale factors.** This optional field contains a list of positive scale factors for the standard deviations included in the covariance matrix. The number of elements in this list equals the dimension of the expectations vector. Scale factors values of 1 are assumed when this field is not present.
- **Tags (auxiliary values) characterization.** Since observations or states may be optionally accompanied by tags, (Section 3.2), it is necessary to characterize these. The way to do it is to define:
  - The number or tags related to the observation or state (**dimension.**)
  - For each of these tags, provide the **referencing** data (again, reference frame plus coordinate system or coordinate reference frame), as well as the **units** in use.

Figure A2 shows how three different observations are defined (lines 1–29, 30–53 and 54–77), using the ASTROLABE XML syntax. Note the XML tags used to describe the metadata elements: <l_spec> starts the definition of an observation; <type> is used to input the observation's type code. The identifier is specified by means of the <id> tag—which is embedded in a higher level structure named <lineage>. Toolbox codes are specified by means of the <toolbox> tag, while reference frames and coordinate systems are described by means of the (<ref>, <ref_frame_VC>, <coor_system_VC>) triplet. For units, the <units> tag is used, <c> is for covariance matrices and <s> is for their scale factors. The tags are characterized by means of the <t_spec> XML tag, which in turn use other ones already defined.

Figure A3 shows the specification of three states (lines 78–94, 95–119 and 120–137). (This figure also depicts the specification of an instrument, and it will be discussed below.) Note that, as stated before, observation and states are equivalent from the structural standpoint, so the XML tags used to describe these are almost the same. The only difference is the XML tag <p_spec> (opening the definition of a state).

Instruments are characterized as follows:

- **Type, identifier, toolbox.** Same meanings as for observation/states metadata. See Section 3.1 for details on type and identifier codes.
- **List of instrument constants.** Includes the number of constants defined (the **dimension**) and the description of each of these, including:

- **Type.** Either scalar or matrix.
- **Referencing.** Again, either the combination of a reference frame plus a coordinate system or a single reference coordinate frame may be used.
- **Units.** Specifies the units of the instrument constant.
- **Covariance.** Covariance matrix for the instrument constant. Note that covariances in instrument constants are merely indication of the quality of these. This field is optional.
- **Scale factor.** Optional. Scale factor for the standard deviations of the covariance matrix. The value of the scale factor is assumed to be 1 when this field is not present.

Figure A3 includes the definition of an instrument (see lines 138–171). Most of the XML tags used in this example should now be obvious (as <id>, <toolbox>, <units>, <c> or <s>, already commented when describing observation and state metadata). Instruments are described by means of the <i_spec> tag. Once more, their types are input by means of the <type> tag. <c_list> is used to describe the list of constants; each constant is specified by means of the <item> tag, where units, covariances, scale factor and referencing information may be detailed.

The following are the most important fields constituting the metadata characterizing a **model**.

- **Type, identifier and toolbox.** Same meanings as for observation/state/instrument metadata. See Section 3.1 for details on type and identifier codes.
- **List of observations.** The set of identifiers of the observations involved in the model.
- **List of states.** The set of identifiers of the states involved in the model. The role played by each state is also specified. These roles may be either free (the TDS will be responsible for estimating its value) or constant (an input, immutable value will be provided for the state).
- **List of instruments.** The set of identifiers of the instruments involved in the model. This list is optional, since some models may work without the need of instrument constants.
- **List of sub-models.** A model may rely in other models to perform its task. This list includes their identifiers, which is optional.

In Figure A4 three models are defined (each one starting with an <m_spec> tag). The lists of observations and states are input by means of the <l_list> and <p_list> tags respectively. When present—these are optional—instrument and sub-model lists are specified by means of the <i_list> and <sub-m_list> tags. All lists are made of items (XML tag <item> in all cases) providing the identifiers (tag: <id>) of the involved observations, states, instruments or sub-models. State items also describe their role by means of the <role> tag.

It is worth mentioning that the identifiers of observations, states and parameters referenced in these lists correspond to those found in the examples included in Figures A2 and A3. These, together with Figure A4 make a complete example of ASTROLABE metadata.

Note that metadata for models describe the model structure and *not* the logic needed to derive new states out of the involved inputs—this is the task of TDSs themselves. The model type and identifier are the pointers which such software must use to ascertain both the specific mathematical model and the kind of data intervening in the computation of new states.

4. The ASTROLABE Specification

The data model described in Section 3 has been materialized in two different ways: the ASTROLABE file and network interface specifications. Sections 4.1 and 4.2 describe, respectively, the fundamental characteristics of these. The full ASTROLABE specification may be found in [38].

4.1. The File Interface

The ASTROLABE file specification completely implements the data/metadata model described in Sections 3.2–3.5 by means of disk files. These include text and binary formats.

Text files (either data or metadata ones) are written in XML. This language was selected by several reasons: it can be used to describe information accurately and unambiguously; it is based on a proven

standard and uses simple text files that make it durable; it is free of legal constraints or threats; XML may be manipulated programmatically and validated against grammar definition files (schema files). The early availability (as far as 1999) of an open source library for Java/C++ able to parse and validate XML files—Apache Xerces/Xerces-C, see [39]—as well as the involvement of some members of the team in other projects that already used XML-based languages [14], also helped to select XML for ASTROLABE text files.

Table 3 shows the most important files making the ASTROLABE file interface. Among these, the most representative are (1) the observation-events—for observation, state and instrument data—and (2) navigation metadata files, which are described in detail in Appendix A.

Table 3. Summary of the different types of files included in the file interface.

File Type	Description
Observation-events *	Observations (input) or states (output).
Instrument data *	Instruments constant data.
Correlation matrices *	Correlation matrices for (1) trajectory observations; (2) trajectory states and (3) trajectory residuals. One file per type of correlation matrix.
Navigation metadata	Metadata information for observations, states, instruments and models.
Directory	File listing all other files involved in the computation of the output trajectory, as the observation-events or metadata files. May be seen as a "project definition" file.
Process options	Optional, free format. This file contains the list of options controlling the behaviour of the TDS producing the output trajectory. Its goal is to keep a record about how the trajectory was generated.
Process log	Optional, free format. List of messages issued by the TDS when computing the trajectory. For tracking purposes.

* Contain only data, not headers. Use external ASTROLABE header files. See Appendix A.

4.2. The Network Interface

The second materialization of the ASTROLABE data model is the network interface, which relies on TCP/IP (Transmission Control Protocol/Internet Protocol) sockets [40]. Typically, it will be used in real-time environments to communicate two processes, one of them sending data (the producer) and the other one receiving them (the consumer).

The ASTROLABE network interface is far more reduced than its file counterpart. In fact, only the observation (state) and observation equations data entities have been materialized up to the moment. Therefore, two processes communicating via this interface must previously agree on all the facets characterizing data (that is, on their metadata), since there are no mechanisms to transmit (receive) these.

Only three messages exist to exchange observation-events data:

- Data:
  - Observation.
  - Observation equation.
- Commands:
  - End of transmission (end of data).

The structure of the two data messages (observation, observation equation) faithfully mirrors the data model defined in Sections 3.2 and 3.4, summarized also in Tables 1 and 2. Minor changes have been applied to make possible the transmission and reception of data, as the addition of the dimensions of some of the components of the observation or observation equation messages (for instance, to indicate how many elements are included in a list of instrument identifiers) so the right amount of data is transferred. Nonetheless, these changes are implementation-related and in no way change the rationale behind the data model, so these will not be described here for the sake of simplicity.

The unique command message, *End of transmission*, is used by a producer process to tell its consumer counterpart that the transmission of a dataset has finished. It is defined as follows:

end_of_transmission: (message tag)

The unique component of this command is a tag to distinguish this message from others being transmitted between the two endpoints. This tag is the equivalent of the event tag described in Sections 3.2–3.4.

Since the current version of the network interface only allows sending an unique dataset (observation-events data), this command would not be necessary; closing the socket connection would have been enough to signal the end of the transmission process. However, it is foreseen that additional kinds of datasets (as instruments or output parameters) will be transmitted by future versions of the network interface. The availability of an *End of transmission* command will then allow the transmission of several datasets using the same *open* socket connection; there will be no need to close and open connections repeatdly to send different datasets. Sending *End of transmission* messages between these will be enough.

5. The ASTROLABE C++ Library

The CTTC has implemented a portable object oriented C++ library including reader and writer (file interface) and sender/receiver (network interface) classes. It offers all the necessary tools to process ASTROLABE data and metadata.

Specific classes have been provided to manage each available data format (as for instance, text or binary data files, see Section 4.1), so a client software module may use the specific implementation for each of these formats if desired. However, the use of external ASTROLABE header files describing how and where data are to be found, opened the way to a couple of generic classes (a reader or receiver, a writer or sender) able to manage all the available formats in a completely transparent way.

For instance, the generic reader class uses the information found in the ASTROLABE header file to ascertain how and where actual data are stored. Once this is determined, it instantiates (transparently) the appropriate specific reader class. The advantage of using such generic reader lies in the fact that client modules do not need to be aware of the details concerning how data are stored; thus, code is simpler and valid for any available format.

Another feature only available in the file interface of the ASTROLABE C++ library is the ability to read files either in forward or backward directions. This feature has been included since TDSs working in post-processing mode usually generate trajectories in three steps: (1) computing the output in forward direction (that is, from its beginning to its end); then (2) in backwards direction and; finally (3) filtering the two previous results to obtain the final output. Processing backwards implies the need to read data in backward direction, so adding such kind of feature to the library facilitates the development of client software modules.

Performance is also an issue that has been addressed in this library. All readers and writers use a technique known as "buffered reading (writing)" to speed up the process. It is worth to mention that NAVEGA doubled its performance when the ASTROLABE library was used to substitute the former readers and writers that did not implement this technique—no other changes were made to the tool. This is a noticeable improvement, especially when post-processing long datasets covering several hours of data that might take up to an hour or more of elapsed time to process.

The network interface always transmits binary data for efficiency reasons. However, the representation (and interpretation) of binary information may differ depending on the architecture of the processor managing it, more precisely when data are transferred from one computer to another via a network connection. To avoid such problems, the ASTROLABE library encodes and decodes all data using the XDR (eXternal Data Representation) standard [41] thus overcoming this issue. This encoding/decoding processing is performed transparently.

6. Use Cases. Sensors Supported. Sensors Not Supported

The ASTROLABE data model has been put to the test in real use cases along the years, facing different situations involving several projects as well as a variety of sensors. Such exposition to real-life problems made the original data model evolve to what it is now. Obviously, not all the concepts described in this paper were implemented from the very beginning, as for instance, the proper handling of metadata.

Up to the moment, the tandem NAVEGA/ASTROLABE has been able to deal with several kinds of sensors and their related observations. Table 4 summarizes a subset of the projects where NAVEGA has been used, including the sensors and related observation types that were involved in these projects. The interested reader will find more details about one of these projects, GAL, in Appendix B.

In general, ASTROLABE will be able to incorporate any kind of sensor providing observations whose error distributions are fully defined by its first and second moments (see Section 2). This is so because the structure (model) of an ASTROLABE observation (see Section 3.2) includes an expectations array plus a covariance matrix, which is all that an observation with such kind of error distribution needs to be properly modeled.

Table 4. Sensors already tested in ASTROLABE.

Project	Purpose	Sensors (Observations)
ENCORE (see [42,43])	Precise positioning for surveying applications using multi-frequency GNSS data.	GNSS receiver (GNSS raw data).
IADIRA (see [44])	Validation of the close + tight coupling concept.	IMU (accelerations, angular velocities). GNSS receiver (GNSS raw data).
CLOSE-SEARCH (see [45])	Robust navigation for UAVs (real time position and attitude computation using close-coupling).	IMU (accelerations, angular velocities). GNSS receiver (GNSS raw data). EGNOS signals (EGNOS corrections). Magnetometer (Magnetic field). Barometer (pressure).
GAL (see [46])	Airborne gravimetry.	Multiple IMUs (accelerations, angular velocities). GNSS receiver (x, y, z).
ATENEA (see [47])	Mobile mapping for urban cartography.	IMU (accelerations, angular velocities.) GNSS receiver (x, y, z). Odometer (time tagged distances, angular velocities). Optical camera (coordinates of tie points). LiDAR (ranges, angles).

This is true at least from the data standpoint, that is, ASTROLABE's. Obviously, and assuming that it is possible to model some kind of observation using ASTROLABE's approach, it will also be necessary to devise the proper mathematical model(s) for such observation to be useful from the TDS software's point of view. In short, it is necessary to provide with the appropriate mathematical machinery to transform the input observation into output states. This said, it must be clarified that this is not a problem of the ASTROLABE data model at all, since it takes care of data, not algorithms.

The other side of the coin is that observations with error distributions that cannot be fully defined by its first two moments may not be modeled using ASTROLABE. For example, map-matching contexts are usually affected by this kind of problems. See for instance [48] for a description of a situation where the constraints set by the environment (walls) make observations exhibit other types of error distributions.

7. Conclusions and Outlook

The ASTROLABE data model, file and network specifications and successive implementations of the C++ library have been put to the test for several years now. This includes a variety of European projects where different kinds of sensors had to be incorporated (see Table 4 for details). Being conceived as a generic and extensible system, the addition of such new sensors posed no unsurmountable problems for the devised abstraction. Needless to say, this continuous exposition to real-life conditions served to improve the data model progressively. It is possible to state, that ASTROLABE has been, at least up to the moment, able to manage change, evolution and innovation in the field it is specifically targeted at: data representation in the context of TDSs dealing with observations whose error distributions are fully charaterized by the first and second moments. This is of special importance, since the decision of developing a specific data model instead of using existing, mature ones as those described in [32,33] supposed a non-negligible risk.

The ASTROLABE data model and its two specifications (file and network) opened the path to the development of a C++ library exposing a very terse API and a high level of abstraction. Third party software modules accessing ASTROLABE data by means of this library may use very high level code that is independent on details like how and where data are stored (or transmitted).

The development of the ASTROLABE data model, specifications and library went hand in hand with the implementation of CTTC's TDS, NAVEGA. NAVEGA was designed using the same principles on which ASTROLABE rely, that is, it is a generic and extensible software tool mirroring internally the concepts present in the data model. This correspondence notably eased its development and, as expected, reduced to a minimum the maintenance tasks required to cope with change. Other TDSs may also benefit from the characteristics of the ASTROLABE specification by incorporating the library and its features related to data input/output and metadata management.

ASTROLABE is not yet complete. Although both the specification and the C++ library are very close to their final versions, some work remains to be done; for instance, no proper metadata to define the coordinate reference frame for time stamps exist yet. The foreseen work for the coming months will serve to address these minor issues.

The CTTC is considering putting the ASTROLABE specification as well as the portable C++ library in the public domain as soon as the work on these is finished. For more details, contact the authors.

Acknowledgments: The research reported in this paper started around eight years ago at the former Institute of Geomatics and has been funded by means of several European projects. We would like to highlight IADIRA, ATENEA, ENCORE, CLOSE-SEARCH and GAL. The authors would also like to thank Deimos for their support. Last but not least, the authors would like to thank Eduard Angelats and Pere Molina for their time, patience and helpful comments.

Author Contributions: The data model described in this work was devised mainly by M. Eulàlia Parés and Ismael Colomina. José A. Navarro contributed to some parts of the data model, implemented the ASTROLABE C++ library and wrote the majority of this paper. Finally, Marta Bláquez contributed to set the foundations of the data model abstraction.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The File Interface: The Observation-Events and Navigation Metadata Files

This appendix relies on a full example to describe the observation-events and navigation metadata files, the pillars of the ASTROLABE file interface.

The **observation-events** (*obs-e*) file is used to store observations (when used as input) or trajectories (states, on output) since these two data entities are structurally equivalent.

When used as an input file, it contain a series of time-sorted observations and observation equations records—as defined in Sections 3.2 and 3.3. When it represents an output trajectory, only states—also time-sorted—are included.

Figure A1 depicts a complete (although short) observations-event file in text (XML) format used to store observations and observation equations.

```
01     <l id="baro1"        n="32"> 124.88 23.44 1023.44
02                                              0.3                                  </l>
03     <l id="imu1"         n="41"> 124.88       0.01 0.02 0.015 0.32 0.43 9.95
04                                              1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
05     <l id="imu1_bias_pn" n="17"> 124.88       0.0  0.0  0.0   0.0  0.0  0.0  </l>
06     <o id="pva1_d"             > 124.88       27   11   41                   </o>
07     <o id="imu1_bias_d"        > 124.88       11        17                   </o>
08     <o id="height_update"      > 124.88       27   34   32    51             </o>
09
10     <l id="baro1"        n="33"> 124.90 23.45 1023.45
11                                              0.3                                  </l>
12     <l id="imu1"         n="41"> 124.90       0.01 0.01 0.014 0.33 0.44 9.95
13                                              1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
14     <l id="imu1_bias_pn" n="17"> 124.90       0.0  0.0  0.0   0.0  0.0  0.0  </l>
15     <o id="pva1_d"             > 124.90       27   11   41                   </o>
16     <o id="imu1_bias_d"        > 124.90       11        17                   </o>
17     <o id="height_update"      > 124.90       27   35   33    52             </o>
18
19     <l id="baro1"        n="32"> 124.92 23.45 1023.45
20                                              0.3                                  </l>
21     <l id="imu1"         n="41"> 124.92       0.01 0.01 0.013 0.30 0.42 9.95
22                                              1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
23     <l id="imu1_bias_pn" n="17"> 124.92       0.0  0.0  0.0   0.0  0.0  0.0  </l>
24     <o id="pva1_d"             > 124.92       27   11   41                   </o>
25     <o id="imu1_bias_d"        > 124.92       11        17                   </o>
26     <o id="height_update"      > 124.92       27   34   32    51             </o>
```

Figure A1. An observations-event file in text (XML) format.

In this example all the elements described in Sections 3.2 and 3.3 are shown. In every data record, the opening XML tag (either of the lowercase letters "l" or "o") represents the record event tag used to tell apart observations (tag "<l>") from observation equations (tag: "<o>"). Identifiers are input by means of the attribute "id"; the activation flags appear as the attribute "s". The instance identifier (in the case of observations only) is written using the "n" attribute.

The remaining values, located between <l> or <o> opening and closing tags, correspond to the rest of items in the record. To ascertain their meaning, the related metadata are required. These are the time stamp, the tags (auxiliary values, if any), as well as the expectation vector and its related covariance matrix—in the case of observations—or the list of instance identifiers needed by observation equations.

The metadata defining the observation and observation equations included in this example may be found in Figures A2–A4. Line number 03 of Figure A1 contains an observation (<l> tag) data record whose identifier (<id>) is "imu1". The example in Figure A2 defines the metadata for this observation in lines 030–053. More specifically, line 033 includes the identifier sought, "imu1". The dimension of the observation array is six (line 037 in the metadata) so it is correct to have exactly six values (0.01 0.02 0.015 0.32 0.43 9.95) in the observation data record after the first one (124.88) which is the time stamp. Although metadata provide a default covariance matrix in lines 050–051 (in fact, only standard deviations in this case), the data record includes its own values for this matrix in line 04. Lines 047–048 in the metadata file also state that the units for the measurements are rad/s, rad/s, rad/s, m/s^2, m/s^2 and m/s^2.

```
<l_spec s="a">
  <type> baro </type>
  <lineage>
    <id>   baro1 </id>
    <name> Baro measurements </name>
  </lineage>
  <toolbox> BAROMETER </toolbox>
  <dimension> 1 </dimension>
  <ref>
    <ref_frame_VC>
      QNH
    </ref_frame_VC>
    <coor_system_VC>
      cartesian
    </coor_system_VC>
  </ref>
  <units> hPa </units>
  <c> 1 </c>
  <s> 1 </s>
  <t_spec>
    <dimension> 1 </dimension>
    <ref>
      <coor_ref_frame_VC>
        Celsius
      </coor_ref_frame_VC>
    </ref>
    <units> °C </units>
  </t_spec>
</l_spec>
```

```
<l_spec s="a">
  <type> imu </type>
  <lineage>
    <id>   imu1 </id>
    <name> IMU measurements </name>
  </lineage>
  <toolbox> INS </toolbox>
  <dimension> 6 </dimension>
  <ref>
    <ref_frame_VC>
      Bfrd
    </ref_frame_VC>
    <coor_system_VC>
      cartesian
    </coor_system_VC>
  </ref>
  <units>
    rad/s, rad/s, rad/s,
    m/s^2, m/s^2, m/s^2
  </units>
  <c> 5e-2 5e-2 5e-2
      1e-1 1e-1 1e-1 </c>
  <s> 1 </s>
</l_spec>
```

```
<l_spec s="a">
  <type> RW_6_PN </type>
  <lineage>
    <id>   imu1_bias_pn </id>
    <name> IMU measurements </name>
  </lineage>
  <toolbox> INS </toolbox>
  <dimension> 6 </dimension>
  <ref>
    <ref_frame_VC>
      Bfrd
    </ref_frame_VC>
    <coor_system_VC>
      cartesian
    </coor_system_VC>
  </ref>
  <units>
    rad/s, rad/s, rad/s,
    m/s^2, m/s^2, m/s^2
  </units>
  <c> 5e-2 5e-2 5e-2
      1e-1 1e-1 1e-1 </c>
  <s> 1 </s>
</l_spec>
```

Figure A2. Metadata: defining observations.

```
<p_spec s="a">
 <type> IMU_bias </type>
 <lineage>
   <id> imu1_bias</id>
   <name> Calibration biases (IMU) </name>
 </lineage>
 <toolbox> INS </toolbox>
 <dimension> 6 </dimension>
 <ref>
   <ref_frame_VC> Bfrd </ref_frame_VC>
   <coor_system_VC>
     cartesian
   </coor_system_VC>
 </ref>
 <units> rad/s, rad/s, rad/s,
         m/s^2, m/s^2, m/s^2 </units>
</p_spec>
```

```
<p_spec s="a">
  <type> baro_cal </type>
  <lineage>
    <id> baro1_cal </id>
    <name>
      Calibration values (barometer)
    </name>
  </lineage>
  <toolbox> BAROMETER </toolbox>
  <dimension> 1 </dimension>
  <ref>
    <ref_frame_VC> QNH </ref_frame_VC>
    <coor_system_VC>
      cartesian
    </coor_system_VC>
  </ref>
 <units> hPa </units>
</p_spec>
```

```
<p_spec s="a">
  <type> pva </type>
  <lineage>
    <id> pva1 </id>
    <name>
      Position, velocity and
      attitude in WGS84
    </name>
  </lineage>
  <toolbox> INS </toolbox>
  <dimension> 9 </dimension>
  <ref>
    <ref_frame_VC> WGS84 </ref_frame_VC>
    <coor_system_VC>
      geodetic, geodetic, geodetic,
      Lned, Lned, Lned,
      Bfrd-Lned, Bfrd-Lned, Bfrd-Lned
    </coor_system_VC>
  </ref>
  <units>
    rad, rad, m,
    m/s, m/s, m/s,
    rad, rad, rad
  </units>
</p_spec>
```

```
<i_spec s="a">
  <type> baro_p0_h0 </type>
  <lineage>
    <id> p0_h0 </id>
    <name>Initial pressure+altitude</name>
  </lineage>
  <toolbox> BAROMETER </toolbox>
  <c_list>
    <dimension> 2 </dimension>
    <item n="1">
      <type> scalar </type>
      <ref>
        <ref_frame_VC> QNH </ref_frame_VC>
        <coor_system_VC>
          pressure
        </coor_system_VC>
      </ref>
      <units> mBar </units>
      <c> 1.2 </c>
      <s> 1 </s>
    </item>
    <item n="2">
      <type> scalar </type>
      <ref>
        <coor_ref_frame_VC>
          WGS84-ellipsoidal-height
        </coor_ref_frame_VC>
      </ref>
      <units> m </units>
      <c> 0.15 </c>
      <s> 1 </s>
    </item>
  </c_list>
</i_spec>
```

Figure A3. Metadata: defining states and instruments.

```
<m_spec s="a">
  <type> local_mec_eq_bias_d </type>
  <lineage>
    <id>   pva1_d  </id>
    <name> Mechanization equations </name>
  </lineage>
  <toolbox> INS </toolbox>
  <l_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> imu1 </id>
    </item>
  </l_list>
  <p_list>
    <dimension> 2 </dimension>
    <item n="1">
      <id> pva1 </id>
      <role> free </role>
    </item>
    <item n="2">
      <id> imu1_bias </id>
      <role> free </role>
    </item>
  </p_list>
  <sub-m_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> WGS84 </id>
    </item>
  </sub-m_list>
</m_spec>
```

```
<m_spec s="a">
  <type> RW_6_d </type>
  <lineage>
    <id> imu1_bias_d </id>
    <name>IMU biases as random walk</name>
  </lineage>
  <toolbox> INS </toolbox>
  <l_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> imu1_bias_pn </id>
    </item>
  </l_list>
  <p_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> imu1_bias </id>
      <role> free </role>
    </item>
  </p_list>
</m_spec>
```

```
<m_spec s="a">
  <type> PVA_baro_U </type>
  <lineage>
    <id> height_update  </id>
    <name> Update of height </name>
  </lineage>
  <toolbox> BAROMETER </toolbox>
  <l_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> baro1 </id>
    </item>
  </l_list>
  <p_list>
    <dimension> 2 </dimension>
    <item n="1">
      <id> pva1 </id>
      <role> free </role>
    </item>
    <item n="2">
      <id> baro1_cal </id>
      <role> free </role>
    </item>
  </p_list>
  <i_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> p0_h0 </id>
    </item>
  </i_list>
</m_spec>
```

Figure A4. Metadata: defining models.

The observation equation (tag <o>) in line 06 of this example includes the model whose identifier (<id>) is "pva1_d". This model is described in the metadata shown in Figure A4, lines 172–202. The descriptive text in the <name> field states that this model implements the so-called "mechanization equations". Looking at the <dimension> tags in lines 180 and 186 it should be clear that this equation relates 1 observation (whose identifier is "imu1", line 182) and 2 states ("pva1" and "imu1_bias", lines 188 and 192). The list of sub-models (lines 196–201 in this metadata file) states than an additional sub-model will be required ("WGS84", line 199). Matching the list of instance identifiers included in the observation equation data record in Figure A1, line 06 and the aforementioned metadata leads to the following correspondences: 27 is the instance identifier of an observation whose identifier is "imu1"; 11 and 41 are instance identifiers of states whose identifiers are "pva1" and "imu1_bias" respectively.

Note that data coming from two identical barometers has been acquired (Figure A1, lines 01, 10 and 19). This is made evident by the different instance identifiers used in these three observations (attribute "n" is 32 in lines 01 and 19 but it is 33 in line 10).

As shown above, all the information in the data file is fully characterized by the metadata files. Cross-referencing data and metadata files is made by means of the identifiers (see Section 3.1).

An observation-events file used to represent **output trajectories** (series of states) adheres to the description above. The only difference with actual observation data files is that *no* observation equations are included—there is no need to compute new states since all these have already been obtained.

Finally, observation-events are pseudo-XML files, since no opening or closing XML headers or footers are present. Only <l> (and eventually <o>) records are included. Headers are stored in *external ASTROLABE header files*.

Figure A5 shows two examples of ASTROLABE header files; the first one points to the text observations-events data file just described, while the second one shows how an ASTROLABE data stream is characterized.

No matter what the case is, an ASTROLABE header file will always include the following items:

- The kind of data being stored or transmitted (as an observation-events file or an instruments file). This is specified by means of the tag "type"—see lines 08 and 20 in Figure A5.
- Where actual (observations or states) data are stored or transmitted. When using the file interface, data may be stored either in text or binary files. This is specified by means of the attribute "format" included in the header's <data> tag (line 08 in Figure A5). In the case of the network interface, data are sent or received through TCP/IP sockets (see Section 4.2 for details). Line 20 shows how the "format" tag changes to state that a "socket" is used.
- The parameters used to characterize the data channel. In the case of files, a file name is used (line 09 in Figure A5) while an optional server plus a port number is required in the case of network streams (line 21 in the same example).

There are two reasons to separate headers from actual data:

- Uniformity. All kinds of data materializations (binary or text files, network streams) are characterized in the same way through the use of a separate header. Data (in whatever form) always contain only <l> and <o> records, and no more.
- Data segmentation. Separating headers from actual data is useful if (data) files are to be split into chunks for security reasons; a unique header may be therefore attached to a *series* of successive data files representing a single data acquisition campaign. Should a failure in the acquisition system occur, only the last file would be lost, thus minimizing data loss. In fact, the ASTROLABE data model allows for file segmentation for this specific reason. Data segmentation implies, of course, a naming scheme for the different data chunks thus generated. ASTROLABE defines such naming convention, although it is not described here.

The selection of the different data formats available in ASTROLABE will obviously depend on the user's criteria and constraints. Text, for instance, is very useful in post-processing tasks, where humans

may edit easily readable files. Binary versions of these, on the contrary, reduce significantly the amount of storage needed to keep big amounts of data but are difficult to handle by human personnel. A well known example of the text/binary implementations tandem are RINEX [34] and BINEX [49] (BINary EXchange format) respectively. Sockets will be a must in the case of distributed processing. This is the reason why the ASTROLABE specifications provide with different specifications, trying to cover a wide range of situations.

The **navigation metadata files** are the mechanism to store all the metadata (see Section 3.5) related to the several data entities included in the ASTROLABE data model. These are always text, XML-based files.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <astrolabe-header_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03   version="1.0" xsi:noNamespaceSchemaLocation="astrolabe-header_file.xsd">
04   <lineage>
05     <id> july_campaign_01 </id>
06   </lineage>
07   <data>
08     <device type = "obs-e_file" format="text_file">
09       july_campaign_data.obs-e
10     </device>
11   </data>
12 </astrolabe-header_file>

13 <?xml version="1.0" encoding="UTF-8"?>
14 <astrolabe-header_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15   version="1.0" xsi:noNamespaceSchemaLocation="astrolabe-header_file.xsd">
16   <lineage>
17     <id> july_campaign_01 </id>
18   </lineage>
19   <data>
20     <device type = "obs-e_file" format="socket">
21       localhost:2000
22     </device>
23   </data>
24 </astrolabe-header_file>
```

Figure A5. Examples of ASTROLABE XML header files.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <nav_metadata_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03                   xsi:noNamespaceSchemaLocation="nav_metadata.xsd">

                ...

** </nav_metadata_file>
```

Figure A6. Example of a navigation metadata file's header and footer.

Figure A6 depicts the header and footer in a navigation metadata file. It has been included for the sake of completeness. This figure is incomplete. To obtain a fully fledged navigation metadata file, the contents of Figures A2–A4 should be inserted (no matter the order) in the place of the ellipsis.

Navigation metadata files always include header and footer XML tags, as opposite to data files, that use the separate ASTROLABE header file.

Note that:

- In lines 02 and ** (the last one) the tag to denote navigation metadata files may be found: <nav_metadata_file>.

- In line 03 the name of the validating schema used to check the correctness of the file is specified: nav_metadata.xsd.

To finish, the whole set of metadata characterizing a dataset may be included either in a single navigation metadata file or to be spread across several ones.

Appendix B. ASTROLABE in the GAL Project

This appendix describes how the ASTROLABE data model was used in the GAL project, one of the real-life use cases mentioned in Section 6.

The goal of GAL, or *GAlileo for Gravity*, is the study and development of a state-of-the-art methodology for the determination of precise and high-resolution gravity field models. The key issue is the precise Kinematic Airborne Gravimetry (KAG), with the joint use of most recent techniques and technologies, such as GPS/Galileo and strapdown Inertial Measurement Units (IMUs), and its further integration with the global models from the gravimetric mission GOCE.

The main sensors used in the GAL data collection campaigns were one geodetic-grade GNSS receiver (Javad TR-G3T), one navigation-grade IMU (iMAR iNAV FJI) and one high tactical-grade IMU (IGI IIe). Table A1 depicts the main characteristics of the IMUs. The Javad TR-G3T is able to receive GPS (L1/L2/L2C/L5), GLONASS(L1/L2) and Galileo (E1/E5A) signals.

Table A1. Characteristics of the IMUs used in the GAL project.

Characteristic	iMAR iNAV FJI	IGI IIe
Gyro Drift/Offset	0.01°/hr	0.03°/hr
Gyro Random Walk/Q	<0.001°/$\sqrt{h}$	<0.005°/$\sqrt{h}$
Acc. Drift/Offset	<100 µg	<300 µg
Acc. Random Walk/Q	<8 µg/$\sqrt{Hz}$	n/a
Gyro Range	±450°/s	±610°/s
Acc. Range	±5 g (option 2/g)	±20 g
Gyro Drift/Offset	0.01°/hr	0.03°/hr
Gyro Random Walk/Q	<0.001°/$\sqrt{h}$	<0.005°/$\sqrt{h}$
Gyro Scale factor	<30 ppm/<10 ppm	n/a
Gyro Axis Misalignment	<100 µrad	n/a
Acc. Drift/Offset	<100 µg	<300 µg
Acc. Random Walk/Q	<8 µg/$\sqrt{Hz}$	n/a
Acc. Scale factor	<100 ppm/<20 µg/g^2	n/a
Acc. Axis Misalignment	<100 µrad	n/a
Data output rate	1...1000 Hz, BW 400 Hz	128, 256, 400 Hz
Size	370 × 213 × 179 mm	126 × 146 × 98 mm
Weight	10.2 kg, approx.	2.2 kg, approx.

Two aerial campaigns took place, the first in Lleida and the second over the catalan Pyrenees (Spain). 5.5 and 1.7 h of data were collected respectively. Figure A7 shows the flight plans for both campaigns. The selected areas had to fulfill a series of requirements, the most important being that quality gravimetric ground control points had to exist for verification purposes.

Both IMUs deliver the values of three linear accelerations (a_x, a_y, a_z) in m/s^2 and three angular velocities $(\omega_x, \omega_y, \omega_z)$ in rad/s, at 400 Hz. The coordinate reference frame is "body inertial", that is, all measurements are referred to the three IMU's internal x, y and z axes. The set of properties characterizing these IMUs is the same. Therefore, and since there are no differences from this standpoint, the two IMUs may be modeled using the same type (See Section 3.1 for information about types, identifiers and instance identifiers.) of observation. Figure A8 shows the metadata for the FJI and IIe IMUs. Note that the same type is used, but the identifiers differ.

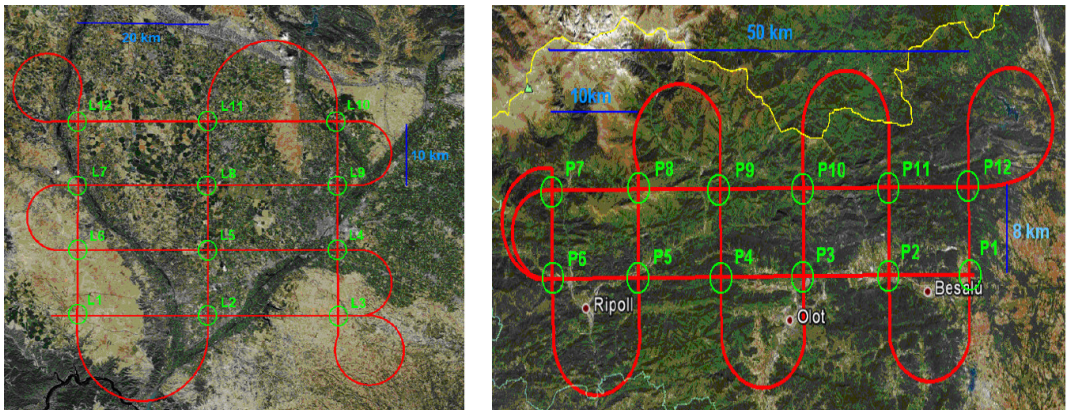

Figure A7. Fligh plans for the Lleida (**left**) and Pyrenees (**righ**) campaigns.

```
01 <l_spec>                                  18 <l_spec>
02   <type> IMU_a_w </type>                  19   <type> IMU_a_w </type>
03   <lineage>                               20   <lineage>
04     <id>   fji </id>                      21     <id>   IIe </id>
05     <name> IMU (a + w) -                  22     <name> IMU (a + w) -
06            iMAR iNAV FJI </name>          23            IGI IIe </name>
07   </lineage>                              24   </lineage>
08   <toolbox> INS </toolbox>                25   <toolbox> INS </toolbox>
09   <dimension> 6 </dimension>              26   <dimension> 6 </dimension>
10   <ref>                                   27   <ref>
11     <coor_ref_frame_VC>                   28     <coor_ref_frame_VC>
12       inertial                            39       inertial
13     </coor_ref_frame_VC>                  30     </coor_ref_frame_VC>
14   </ref>                                  31   </ref>
15   <units> m*s^-2, m*s^-2, m*s^-2,         32   <units> m*s^-2, m*s^-2, m*s^-2,
16           rad/s, rad/s, rad/s </units>    33           rad/s, rad/s, rad/s </units>
17 </l_spec>                                 34 </l_spec>
```

Figure A8. Metadata defining the IMU observations. **Left**: FJI; **right**: IIe.

The Javad GNSS receiver provides positions (x, y, z). Units are meters. The coordinate reference frame is ECEF (Earth-Centered-Earth-Fixed). Figure A9 shows the metadata defining the Javad observation.

The output trajectory is defined by a series of "position, velocity and attitude" (PVA) states, which consist of a position (longitude, latitude, height), a velocity (v_x, v_y, v_z) and three attitude angles (heading, pitch and roll). Such kind of state, with identifier `pva1`, has already been described in a former example (see Figure A3, lines 95–119).

Models are needed to transform observations into states. The most important ones used in the GAL project were (1) PVA prediction, hybridizing the observations originating from the two available IMUs and generating an output PVA state and (2) PVA update, using the GNSS receiver only and producing, again, an output PVA state.

The metadata defining the observation equations related to these two models may be found in Figure A10. Note how the references to the identifiers of the different observations and states involved in these equations clearly define what kind of data will be used or generated by the models. The first one, whose identifier is `pva_2IMU_1`, points to two IMU observations, namely `fji` (line 11) and `IIe` (line 14). The metadata for these two observations may be found in Figure A8. The second model, with identifier `pva_GNSS_1`, requires a single `javad` observation (line 35, Figure A10), which was defined

in Figure A9. Finally, both models point to the same output state, `pva1` (lines 20 and 41, Figure A10), defined in Figure A3, lines 95–119.

```
01 <l_spec>
02   <type> GNSS_xyz </type>
03   <lineage>
04     <id>   javad </id>
05     <name> GNSS receiver delivering
06            x, y and z </name>
07   </lineage>
08   <toolbox> INS </toolbox>
09   <dimension> 3 </dimension>
10   <ref>
11     <coor_ref_frame_VC>
12       ECEF
14     </coor_ref_frame_VC>
15   </ref>
16   <units> m, m, m </units>
17 </l_spec>
```

Figure A9. Metadata defining the Javad observations.

```
01 <m_spec>
02   <type> PVA_from_double_IMU </type>
03   <lineage>
04     <id> pva_2IMU_1 </id>
05     <name> Two IMUs to PVA </name>
06   </lineage>
07   <toolbox> INS </toolbox>
08   <l_list>
09     <dimension> 2 </dimension>
10     <item n="1">
11       <id> fji </id>
12     </item>
13     <item n="2">
14       <id> IIe </id>
15     </item>
16   </l_list>
17   <p_list>
18     <dimension> 1 </dimension>
19     <item n="1">
20       <id> pva1 </id>
21       <role> free </role>
22     </item>
23   </p_list>
24 </m_spec>
25 <m_spec>
26   <type> PVA_from_GNSS </type>
27   <lineage>
28     <id> pva_GNSS_1  </id>
29     <name> GNSS xyz to PVA </name>
30   </lineage>
31   <toolbox> INS </toolbox>
32   <l_list>
33     <dimension> 1 </dimension>
34     <item n="1">
35       <id> javad </id>
36     </item>
37   </l_list>
38   <p_list>
39     <dimension> 1 </dimension>
40     <item n="1">
41       <id> pva1 </id>
42       <role> free </role>
43     </item>
44   </p_list>
45 </m_spec>
```

Figure A10. Two models from the GAL project.

```
1 <l id="fji" n="1">   474631.000  0.048352247977806 -0.102871215632823   8.98256623212034
2                                 -0.024930613136148  0.050247523604562   0.006996777810447 </l>
3 <l id="lle" n="1">   474631.000 -0.23193359375     -0.18310546875     -10.6811523437
4                                  0.006484985351562  0.011825561523437 -0.010299682617187  </l>
5 <l id="javad" n="1"> 474631.000  4749387.498        228989.265         4237067.382        </l>
6 <o id=pva_2IMU_1>    474631.000 1 1 1 </o>
7 <o id=pva_GNSS_1>    474631.000 1 1   </o>
```

Figure A11. Actual data from the GAL project (a single epoch).

Having described the relevant metadata, it is possible to see—and understand—actual data from the GAL project. Figure A11 shows a single epoch (data sharing the same time tag) including an instance of every observation and observation equation discussed above.

Note the event tags "l" or "o" telling apart observations and equations, the different identifiers characterizing these (`fji`, `IIe`, `javad`, `pva_2IMU_1` and `pva_GNSS_1`), and the instance identifiers, which have the same value (1) in all cases. This is not a problem, though.

Since instance identifiers are used to tell apart observations *whose identifier is the same*, they do not need to be different *when identifiers already differ*. In fact, the full identification of observations in the data realm is achieved by means of the union of identifiers plus instance identifiers: `fji` + 1, `IIe` + 1 and `javad` + 1 for the three observations in the example. These unions are unique, so there is no ambiguity.

In the case of observation equations (Figure A11, lines 6–7), it is the position of the instance identifiers that eliminates the ambiguity. For instance, in the equation in line 6 (identifier `pva_2IMU_1`) the first instance identifier points to `fji` observations, the second to `IIe` ones and the third to `pva1` states (see the corresponding metadata for the observation equation in Figure A10). So, in spite of the numerical coincidence, there is no doubt about what these instance identifiers are pointing to (`fji` + 1, `IIe` + 1 and `pva1`+1).

The value of the time tag is 474631.000 in all cases. Lines 1–2 and 3–4 in Figure A11 depict observations from the FJI and IIe IMUs respectively. These are structurally equivalent (both include three linear accelerations and angular velocities). Even though the units used are the same, the corresponding acceleration or angular velocity values do not match, not even approximately. This is because the IMUs were not mounted in the same position and therefore the axes pointed to different directions. Line 5 in Figure A11 depicts a Javad (GNSS) observation, including its three position components (x, y, z).

The covariance matrices have been removed in all cases to simplify the example—these may be omitted in ASTROLABE data and specified in the metadata defining the observations, if desired.

Finally, lines 6 and 7 in Figure A11 show an example of the two observation equations corresponding to the two models. As already stated, the first one (`pva_2IMU_1`) is used to predict a new PVA state using one observation from each IMU, while the second one (`pva_GNSS_1`), updates the PVA state using data originating from the Javad GNSS receiver only.

References

1. Titterton, D.H.; Weston, J.L. *Strapdown Inertial Navigation Technology*, 2nd ed.; The American Institute of Aeronautics and Astronautics: Reston, VA, USA; The Institution of Electrical Engineers: Herts, UK, 2004.
2. Colomina, I.; Molina, P. Unmanned Aerial Systems for Photogrammetry and Remote Sensing: A review. *ISPRS J. Photogramm. Remote Sens.* **2014**, *92*, 79–97.
3. Karpowicz, J. *Above the Field with UAVs in Precision Agriculture*; Commercial UAV Expo: Las Vegas, NV, USA, 2016.
4. European GNSS Agency (GSA). *GNSS Market Report*; European GNSS Agency: Prague, Czech Republic, 2015.

5. Zhu, L.; Yang, A.; Wu, D.; Liu, L. Survey of Indoor Positioning Technologies and Systems. In *Life System Modeling and Simulation*; Springer: Heidelberg, Germany, 2014; pp. 400–409.
6. Minh, V.T. Trajectory Generation for autonomous vehicles. In *Mechatronics 2013: Recent Technological and Scientific Advances*; Springer: Berlin, Germany, 2014; pp. 615–626.
7. Mongrédien, C.; Hide, C.; Fairhurst, P.; Ammann, D. Centimeter positioning for UAVs and mass market applications: UAVs, precision agriculture and robotic guidance require high accuracy at low cost. *GPS World* **2016**, *25*, 24–33.
8. Groves, P.D.; Wang, L.; Walter, D.; Martin, H.; Voutsis, K. Toward a Unified PNT—Part 1, Complexity and Context: Key Challenges of Multisensor Positioning. *GPS World* **2014**, *10*, 18–49.
9. Groves, P.D.; Wang, L.; Walter, D.; Martin, H.; Voutsis, K. Toward a Unified PNT — Part 2, Ambiguity and Environmental Data: Two Further Key Challenges of Multisensor Positioning. *GPS World* **2014**, *25*, 18–35.
10. Beck, K.; Andres, C. *Extreme Programming Explained: Embrace Change*, 2nd ed.; Addison-Wesley Professional: Boston, MA, USA, 2004.
11. Martin, R.C. *Agile Software Development: Principles, Patterns, and Practices*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2003.
12. Meyer, B. *Object-Oriented Software Construction*, 2nd ed.; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1997.
13. Colomina, I.; Navarro, J.A.; Térmens, A. GeoTeX: A general point determination system. In Proceedings of the XVIIth International Congress of the ISPRS, Washington, DC, USA, 2–14 August 1992.
14. Colomina, I.; Blázquez, M.; Navarro, J.A.; Sastre, J. The need and keys for a new generation network adjustment software. In Proceedings of the ISPRS Congress, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Melbourne, Australia, 25 August–1 September 2012.
15. Parés, M.E.; Colomina, I. On software Architecture Concepts for a Unified, Generic and Extensible Trajectory Determination System. In Proceedings of the ION GNSS, Tampa, FL, USA, 8–12 September 2015.
16. Soloviev, A.; Yang, C. Reconfigurable Integration Filter Engine (RIFE) for Plug-and-Play Navigation. In Proceedings of the ION GNSS+, Nashville, TN, USA, 16–20 September 2013.
17. Soloviev, A.; Veth, M.; Yang, C.; Miller, M. Plug and play sensor fusion for navigation in GNSS-challenged environments. In Proceedings of the ION GNSS+ 2016, Portland, OR, USA, 12–13 September 2016.
18. Förstner, W.; Wrobel, B.; Paderes, F.; Craig, R.; Fraser, C.; Dolloff, J. Analytical Photogrammetric Operations. In *Manual of Photogrammetry*, 5th ed.; McGlone, C., Ed.; American Society for Photogrammetry and Remote Sensing: Bethesta, MA, USA, 2004; pp. 887–943.
19. Tscherning, C.C. Defining the basic entities in a geodetic data base. *Bull. Géod.* **1978**, *52*, 85–92.
20. Elassal, A.A. Generalized adjustment by least squares (GALS). *Photogramm. Eng. Remote Sens.* **1983**, *49*, 201–206.
21. Sarjakoski, T. Object-oriented approaches in the design of more capable (adjustment) systems. In Proceedings of the XVIth International Congress of the ISPRS, Kyoto, Japan, 1–10 July 1988.
22. Crippa, B.; de Haan, A.A.; Mussio, L. The formal structure of geodetic and photogrammetric observations. In Proceedings of the Tutorial on Mathematical Aspects of Data Analysis, ISPRS, Intercomission WG III/VI, Pisa, Italy, 1–2 June 1989.
23. Colomina, I. Discrete mathematical techniques in the analysis and adjustment of hybrid networks. In Proceedings of the XVIIth International Congress of the ISPRS, Washington, DC, USA, 2–14 August 1992.
24. Kresse, W.; Fadaie, K. *ISO Standards for Geographic Information*; Springer: Berlin/Heidelberg, Germany, 2004.
25. Hasan, A.M.; Samsudin, K.; Ramli, A.R.; Azmir, R.S.; Ismaeel, S.A. A Review of Navigation Systems (Integration and Algorithms). *Aust. J. Basic Appl. Sci.* **2009**, *3*, 943–959.
26. Gade, K. NAVLAB, a Generic Simulation and Post-processing Tool for Navigation. *Model. Identif. Control* **2005**, *26*, 135–150.
27. Martin, M.K. New low cost avionics with INS/GPS for a variety of vehicles. *IEEE Aerosp. Electron. Syst. Mag.* **1998**, *13*, 41–46.
28. Hagen, O.K. TerrLab—A generic simulation and post-processing tool for terrain referenced navigation. In Proceedings of the OCEANS 2006, Shanghai, China, 18–21 September 2006.
29. O'Leary, P.; Eliser, J.; Turbe, M.; Betts, K. Implementation of an Open Architecture for Plug-and-Play Navigation Software. In Proceedings of the Joint Navigation Conference, Dayton, OH, USA, 7–9 June 2016.
30. Elsner, D. Universal Plug-N-Play Sensor Integration for Advanced Navigation. Ph.D. Dissertation, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, USA, 2012.

31. Ashkenazi, V. Coordinate Systems: How to Get Your Position Very Precise and Completely Wrong. *J. Navig.* **1986**, *39*, 269–278.
32. OGC®. 07-000—OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Available online: http://portal.opengeospatial.org/files/?artifact_id=21273 (accessed on 28 October 2015).
33. OGC®. 10-004r3—Geographic Information—Observations and Measurements. Available online: http://portal.opengeospatial.org/files/?artifact_id=41579 (accessed on 28 October 2015).
34. RINEX. The Receiver Independent Exchange Format. Version 3.02. Available online: ftp://igs.org/pub/data/format/rinex302.pdf (accessed on 28 October 2015).
35. LAS Specification. Version 1.3 R11. Available online: http://www.asprs.org/a/society/committees/standards/LAS_1_3_r11.pdf (accessed on 28 October 2015).
36. Grewal, M.; Andrews, P. *Kalman Filtering. Theory and Practice*; Prentice Hall: Englewood Cliffs, NJ, USA, 1993.
37. Groves, P.D. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed.; Artech House: Norwood, MA, USA, 2013.
38. Parés, M.E.; Navarro, J.A.; Colomina, I. *ASTROLABE. Interface Control Document*; Internal Unpublished Report, CTTC / GeoNumerics S.L.; Unpublished work, 2016.
39. Apache Software Foundation. Apache Xerces. Available online: http://xerces.apache.org/ (accessed on 30 May 2016).
40. Stevens, W.R.; Fenner, B.; Rudoff, A.M. *UNIX Network Programming*, 2nd ed.; Addison-Wesley Professional: Boston, MA, USA, 2004.
41. RFC 4506—XDR: External Data Representation Standard. Available online: https://tools.ietf.org/pdf/rfc4506.pdf (accessed on 28 October 2015).
42. Silva, P.; Silva, J.; Peres, T.; Colomina, I.; Miranda, C.; Parés, M.E.; Andreotti, M.; Hill, C.; Galera, J.; Camargo, P.; et al. ENCORE: Enhanced Galileo code receiver for surveying applications. In Proceedings of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, ME, USA, 2011; pp. 3679–3689.
43. Colomina, I.; Miranda, C.; Parés, M.E.; Andreotti, M.; Hill, C.; da Silva, P.F.; Silva, J.S.; Parés, T.; Galera, M.J.F.; Camargo, P.O.; et al. Galileo's surveying potential. *GPS World* **2012**, *3*, 18–33.
44. Silva, P.; Silva, J.; Caramagno, A.; Wis, M.; Parés, M.E.; Colomina, I.; Fernández, J.; Diez, J.; Gabaglio, V. IADIRA: Inertial aided deeply integrated receiver architecture. In Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006), Fort Worth, TX, USA, 26–29 September 2006; pp. 2686–2694.
45. Molina, P.; Colomina, I.; Vitoria, T.; Silva, P.F.; Skaloud, J.; Kornus, W.; Prades, R.; Aguilera, C. Searching lost people with UAVs: The system and results of the CLOSE-SEARCH project. In Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Melbourne, Australia, 25 August–1 September 2012; Volume XXXIX-B1, pp. 299–306.
46. Skaloud, J.; Colomina, I.; Parés, M.E.; Blázquez, M.; Silva, J.; Chersich, M. Progress in airborne gravimetry by combining strapdown inertial and new satellite observations via dynamic networks. In Proceedings of the IUGG 2015 Conference, Prague, Czech Republic, 22 June–2 July 2015.
47. Fernández, A.; Diez, J.; de Castro, D.; Dovis, F.; Silva, P.; Friess, P.; Wis, M.; Colomina, I.; Lindenberger, J.; Fernández, I. ATENEA: Advanced techniques for deeply integrated GNSS/INS/LiDAR navigation. In Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS), Portland, OR, USA, 19–23 September 2011; pp. 2395–2405.
48. Isaacs, J.T.; Irish, A.T.; Quitin, F.; Madhow, U.; Hespanham, J.P. Bayesian localization and mapping using GNSS SNR measurements. In Proceedings of the IEEE/ION PLANS 2014, Monterey, CA, USA, 5–8 May 2014.
49. BINEX. Binary Exchange Format. Available online: http://binex.unavco.org/binex.html (accessed on 28 October 2015).

Bibliography

[1] Aggarwal, P., El-Sheimy, N., 2009. Hybrid Extended Particle Filter (HEPF) for INS/GPS Integrated System. In Proceedings of the 22nd International Meeting of the Satellite Division of The Institute of Navigation, Savannah (USA), 2009 September 22-25.

[2] Alinsavath, K. N., Nugroho, L. E., Hamamoto, K., 2019. The Seamlessness of Outdoor and Indoor Localization Approaches based on a Ubiquitous Computing Environment: A Survey. In Proceedings of the 2019 2nd International Conference on Information Science and Systems. pp. 316-324. ACM.

[3] Angelats, E., Parés, M.E., Colomina, I., 2011. Methods, algorithms and tools for precise terrestrial navigation. In Proceedings of 9th International Geomatic Week, Barcelona (Spain), 2011 March 15-17.

[4] Angelats, E., Molina, P., Parés, M.E., Colomina,I., 2014. A parallax-based robust image matching for improving multisensor navigation in GNSS-denied environments. In Proceedings of the ION GNSS+ 2014, Tampa (USA),2014 September 08-12.

[5] Angelats, E., Parés, M. E., Colomina, I., 2016. The potential of non-semantic features for UAV remote sensing data fusion. In Proceedings of 2nd Virtual Geoscience Conference, Bergen (Norway), 2016 September 21–23.

[6] Angelats, E., Navarro, J.A., 2017. Towards a fast, low-cost indoor mapping and positioning system for civil protection and emergency teams. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42.

[7] Ascending Technologies, 2017. AscTec Falcon 8 + AscTec Trinity – Safety data sheet. http://www.asctec.de/downloads/public/F8_AscTec-Falcon-8-AscTec-Trinity_safety-data-sheet.pdf Accessed: 6 September 2019

[8] Baarda, W., 1967. Statistical concepts in geodesy (Vol. 2). Rijkscommissie voor Geodesie.

[9] Becker, D., Nielsen, J. E., Ayres-Sampaio, D., Forsberg, R., Becker, M., Bastos, L., 2015. Drift reduction in strapdown airborne gravimetry using a simple thermal correction. Journal of Geodesy, 89(11), 1133-1144.

[10] Bruton,A.,2000. Improving the accuracy and resolution of SINS/DGPS airborne gravimetry. PhD Thesis, UCGE Reports No. 20145, Department of Geomatics Engineering, The University of Calgary, Calgary, Alberta, Canada.

[11] Clausen, P., Skaloud, J., Orso, S., Guerrier, S., 2018. Construction of dynamically-dependent stochastic error models. In Proceedings of the 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS) (pp. 1336-1341). IEEE.

[12] Colomina, I., Navarro, J.A., Térmens, A., 1992. GeoTeX: a general point determination system. In Proceedings of the XVIIth International Congress of the ISPRS (International Society for Photogrammetry and Remote Sensing), Washington DC (USA), 1992 August 2-14.

[13] Colomina, I., Giménez, M., Rosales, J.J., Wis, M., 2004. On the use of redundant inertial data for geodetic appliactions. In Proceedings of the IONS GNSS 2004,Long Beach (USA), 2004 September 21-24.

[14] Colomina, I., Parés, M.E., 2011. Sensor Orientation: precise trajectory & attitude determination with INS. Lecture Notes of the Geomatic Updates international course 2011.

[15] Colomina I., Blázquez, M., Navarro J.A., Sastre J., 2012. The need and keys for a new generation network adjustment software. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 303-308.

[16] Colomina, I., Miranda,C., Parés,M.E., Andreotti, M., Hill,C., da Silva,P. F., Silva,J. S., Perez,T. Galera, M. J. F. P., Camargo,O. ,Fernández, A. , Palomo,J. M., Moreira, J., Streiff, G., Granemann, E. Z., Aguilera , C., 2012. Galileo's Surveying Potential. GPS World, Vol 23, No 3, 2012.

[17] Colomina, I., Molina, P., 2014. Unmanned aerial systems for photogrammetry and remote sensing: A review. ISPRS Journal of photogrammetry and remote sensing, 92, 79-97.

[18] Curran, J.T., Fernández-Prades, C., Morrison, A., Bavaro, M., 2018. Innovation: The continued evolution of the GNSS software-defined radio. GPS World, Vol 29, No 1, 2018.

[19] Dowson, M., 1997. The Ariane 5 software failure. ACM SIGSOFT Software Engineering Notes, 22(2), 84.

[20] Elassal, A.A., 1983. Generalized adjustment by least squares (GALS). Photogrammetric Engineering and Remote Sensing, 49(2), 201-206.

[21] Fang, J., Xiaolin Gong, X., 2010. Predictive Iterated Kalman Filter for INS/GPS Integration and Its Application to SAR Motion Compensation. IEEE Transactions on Instrumentation and Measurement, 59(4), 909-915.

[22] Fernández, A., Diez, J., de Castro, D., Silva, P.F., Colomina, I., Parés, M.E., Dovis, F., Friess, P., Wis, M., Lindenberger, J., Fernandez. I., 2011. ATENEA: Advanced Techniques for Deeply Integrated GNSS/INS/LiDAR Navigation. In Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, OR, September 2011.

[23] Fernández, A., Wis, M., Colomina, I., Angelats, E., Parés,M.E., 2014 . Real-time Navigation and Mapping with Mobile Mapping Systems using LiDAR/Camera/INS/GNSS Advanced Hybridization Algorithms: Description and Test Results , In Proceedings of the ION GNSS+ 2014, Tampa (USA),2014 September 08-12.

[24] Fernández, E., Calero,D., Parés,M.E., 2017. CSAC Characterization and Its Impact on GNSS Clock Augmentation Performance. Sensors, 17(2), 370.

[25] García, P., 2016. IMU calibration platform. TFG Thesis, Universitat Politècnica de Catalunya, Castelldefels, Barcelona, Spain.

[26] Geonumerics,2019. REALITY. RPAS EGNOS adoption and liaison with navigation integrity, http://www.geonumerics.es/index.php/projects/88-reality-rpas-egnos-adoption-and-liaison-with-navigation-integrity Accessed: 7 October 2019

[27] Geospatial Solutions Staff, 2015. Geodetics Teams with Velodyne for Real-Time Mobile Mapping Systems. Geospatial solutions, May 7, 2015.

[28] Grewal,M.S., Andrews,A.P.,1993. Kalman filtering: theory and practice. Prentice-Hall, Englewood Cliffs.

[29] Grewal, M.S., Kain, J., 2010. Kalman filter Implementation with Improved Numerical Properties IEEE transactions on automatic control, 55(9), 2058-2068.

[30] Groves P. D, Wang L., Walter D., Martin H., Voutsis K., 2014. Toward a Unified PNT — Part 1, Complexity and Context: Key Challenges of Multisensor Positioning. GPS World, Vol 25, No 10, 2014.

[31] Groves P. D, Wang L., Walter D., Martin H., Voutsis K., 2014. Toward a Unified PNT — Part 2, Ambiguity and Environmental Data: Two Further Key Challenges of Multisensor Positioning. GPS World, Vol 25, No 10, 2014.

[32] Guerrier, S., 2010. Robust FDI in redundant MEMS-IMUS Systems. In Proceedings of the Calibration and Orientation Workshop EuroCOW 2010, EuroSDR and ISPRS, Castelldefels, Spain, 2010 February 10–12

[33] Guivant, J.E. and Nebot, E.M., 2001. Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation. IEEE transactions on robotics and automation, 17(3), 242-257.

[34] Hernández-Pajares,M., 2000. Application of ionospheric tomography to real-time GPS carrier-phase ambiguities resolution, at scales of 400-1000 km and with high geomagnetic activity. Geophysical Research Letters, 27(13), 2009-2012.

[35] Hou, H.,2004. Modeling Inertial Sensors Errors Using Allan Variance. PhD Thesis, UCGE Reports No. 20201, Department of Geomatics Engineering, The University of Calgary, Calgary, Alberta, Canada.

[36] Humpherys, J., West, J. 2010. Kalman filtering with Newton's method [lecture notes]. IEEE Control Systems Magazine, 30(6), 101-106.

[37] ICAO, 2007 ICAO Standard and Recommended Procedures (SARPS) Annex 10

[38] IEEE,2005. IEEE Recommended Practice for Inertial Sensor Test Equipment, Instrumentation, Data Acquisition and Analysis. IEEE std 1554-2005.

[39] IFEN,2015. NavX-NCS Professional GNSS Simulator, http://www.ifen.com/products/navx-ncs-professional-gnss-simulator.html Accessed: 6 October 2015

[40] Jekeli,C.,2001. Inertial Navigation Systems with Geodetic Applications. Walter de Gruyter, Berlin.

[41] Khaghani, M., Skaloud, J., 2016. Autonomous Vehicle Dynamic Model Based Navigation for Small UAVs. Navigation: Journal of The Institute of Navigation, 63(3), 345-358.

[42] Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Journal of basic Engineering, 82(1), 35-45

[43] Kalman, R.E., Bucy, R.S., 1961. New results in linear filtering and prediction theory. Journal of basic engineering, 83(1), 95-108.

[44] Kennedy, S., Rossi, J., 2008. Performance of a Deeply Coupled Commercial Grade GPS/INS System from KVH and NovAtel Inc. In Proceedings of IEEE/ION PLANS 2008. Monterey, California. 2008 May 6 - 8.

[45] Keymeulen, D., et al. 2005. Effect of temperature on MEMS vibratory rate gyroscope. Conference Proceedings of IEEE Aerospace Conference 2005, 2005.

[46] ,Konolige, K and Agrawal, M. 2009. FrameSLAM: From bundle adjustment to real-time visual mapping. IEEE Transactions on Robotics. Vol 24, 5. pp 1066–1077.

[47] Krishnaswamy, K., 2008. Sensor Fusion for GNSS Denied Navigation. In Proceedings of IEEE/ION PLANS 2008. Monterey, California. 2008 May 6 - 8.

[48] Lapachelle, G., Petovello, M., Beard, R., Raym, J.K., 2016. GNSS Solutions: Atomic Clock on Satellites and Mitigating Multipath. INSIDE GNSS, 22–27, 2016.

[49] Lennartsson, A., Skoogh,D.,2003. Sensor redundancy for inertial navigation. FOI-R-1085-SE, december 2003, 1650-1942, FOI(Swedish Defence Research Agency), pp.31, Stockholm, SE.

[50] Lee-Ryeok,H. 2004. A Fuzzy-Kalman filtering strategy for state estimation PhD Thesis, College of graduate studies and research of the university of Seskatchewan, Seskatchewan, Canada.

[51] Maybeck,P.S.,1979. Stochastic models, estimation and control. Academic Press Inc.

[52] Mathworks team. Trajectory Generation (3rd & 5th orders). http://www.mathworks.com/matlabcentral/fileexchange/402 trajectory-generation–3rd—5th-orders- Accessed: 6 September 2019

[53] Mathworks team. Three-Axis Inertial Measurement Unit. http://es.mathworks.com/help/aeroblks/threeaxisinertialmeasur Accessed: 6 September 2019

[54] Mircosemi. Microsemi Quantum SA.45s Datasheet, ref: 900 00331 000H. Microsemi Corporate Headquarters: Aliso Viejo, CA, USA, December 2014 (121014).

[55] Molina, P., Colomina, I., Troger, M., Hofmann-Wellenhof, B., Aguilera, C., 2011. Qualitative Motion Analysis: INS/GNSS in Care-Giving Applications. GPS World, Vol 22, No 1, 2011.

[56] Molina, P., Colomina, I., Victoria, P., Skaloud, J., Kornus, W., Prades, R., Aguilera, C., 2012. Drones to the rescue. InsideGNSS, pp. 36-47, July 2012.

[57] Montaño, J., Wis, M., Pulido,J.A., Latorre,A., Molina, P., Fernández, E., Angelats, E., Colomina, I., 2015. Validation of inertial and imaging navigation techniques for space applications with UAVS. In Proceedings of the DASIA 2015 conference. Barcelona.

[58] Nassar, S., Schwarz, K.-P., Naser, E.-S., 2004. Modeling Inertial Sensor Errors Using Autoregressive (AR) Models. Navigation, 51(4), 259-268.

[59] Navarro, J., 1998. Object oriented technologies and beyond for software generation and integration in Geomatics. PhD Thesis, Universitat de les Illes Balears, Illes Balears, Spain.

[60] Navarro, J. A. and Parés, M. E. and Colomina, I. and Bianchi, G. and Pluchino, S. and Baddour, R. and Consoli, A. and Ayadi, J and Gameiro, A. and Sekkas, O. and Tsetsos, V. and Gatsos, T. and Navoni, R.,2015. A redundant GNSS-INS low-cost UAV navigation solution for professional applications. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 40.

[61] Navarro, J.A., Parés, M.E., Colomina, I. Blázquez,M.,2017. ASTROLABE: A rigorous, geodetic-oriented data model for trajectory determination systems. ISPRS International Journal of Geo-Information, 6(4), 98.

[62] Oberg, J., 2001. Year of the rocket [Chinese space research]. IEEE Spectrum, vol. 38(5), 62-68.

[63] Parés, M.E.,Colomina, I. 2015. On software Architecture Concepts for a Unified, Generic and Extensible Trajectory Determination System . In Proceedings of ION GNSS+ 2015, Tampa (USA), 2015 September 14-18.

[64] Parés, M.E.,Navarro, J.A., Colomina, I. 2015. On the generation of realistic simulated inertial measurements. In Proceedings of International Gyro Symposium, Karslruhe, September 2015.

[65] Park, M., 2004. Error Analysis and Stochastic Modeling of MEMS based Inertial Sensors for Land Vehicle Navigation Applications. PhD Thesis, UCGE Reports No. 20194, Department of Geomatics Engineering, The University of Calgary, Calgary, Alberta, Canada.

[66] Preston, S.. Bevly, D., 2014. CSAC-Aided GPS Multipath Mitigation. In Proceedings of the 46th Annual Precise Time and Time Interval Systems and Applications Meeting, Boston, MA, USA, 1–4 December 2014; pp. 228–234.

[67] Preston, S., 2015. GPS Multipath Detection and Mitigation Timing Bias Techniques. Master's Thesis, Auburn University, Auburn, AL, USA, 2015.

[68] Rajamani, R., 2011. Vehicle dynamics and control. Springer Science & Business Media

[69] Ramlall, R., Streeter, J., Schnecker, J.F., 2011. Three satellite navigation in an urban canyon using a chip-scale atomic clock. In Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, OR, September 2011.

[70] Ristic,B., Arulampelam,S., Gordon,N., 2003. Beyond Kalman filter: Particle filters for tracking applications. Artech house.

[71] Rosales, J.J., Colomina, I., 2005. A flexible approach for the numerical solution of the INS mechanization equations. In Proceedings of the 6th International Geomatic Week, Barcelona.

[72] Rose, C., 2009. Camera Vision and Inertial Measurement Unit Sensor Fusion for Lane Detection and Tracking using Polynomial Bounding Curves. In Proceedings of the 22nd International Meeting of the Satellite Division of The Institute of Navigation, Savannah (USA), 2009 September 22-25.

[73] Roskam, J., 1995. Airplane flight dynamics and automatic flight controls. DARcorporation.

[74] Saeedia, S., 2009. Vision-Aided Inertial Navigation for Pose Estimation of Aerial Vehicles. In Proceedings of the 22nd International Meeting of the Satellite Division of The Institute of Navigation, Savannah (USA), 2009 September 22-25.

[75] Sahawneh, L., and M. A. Jarrah., 2008. Development and calibration of low cost MEMS IMU for UAV applications. In Proceedings of the Mechatronics and Its Applications, 2008. ISMA 2008. 5th International Symposium on. IEEE, 2008.

[76] Sales, M. Impact of the characterization and pre-treatment of the inertial signal in the INS-GNSS solution. TFG Thesis, Universitat Politècnica de Catalunya, Castelldefels, Barcelona, Spain.

[77] Sarkka, S. 2009. Recursive Noise Adaptive Kalman filtering by Variational Bayesian Approximations. IEEE Transactions on Automatic control, 54(3), 596-600.

[78] Savage, P.G., 1998. Strapdown inertial navigation integration algorithm design - part 1: attitude algorithms. Journal of guidance, control, and dynamics, 21(1), 19-28.

[79] Savage, P.G., 1998. Strapdown inertial navigation integration algorithm design - part 2: velocity and position algorithms. Journal of Guidance, Control, and Dynamics, 21(2), 208-221.

[80] SESAR, 2016. European Drones Outlook Study.

[81] Schön, T.B., Lindsten, F., Dahlin, J., Wågberg, J., Naesseth, C.A. Svensson, A., Dai, L., 2015. Sequential Monte Carlo Methods for System Identification. arXiv preprint arXiv:1503.06058. 2015.

[82] Silva, P., 2007. Inertial Aiding: Performance Analysis using Tight Integrated Architecture. In Proceedings of the TimeNAV 2007.

[83] Silva P., Colomina,I. Miranda,C. Parés,M.E., 2011. ENCORE: Enhanced Galileo Code Receiver for Surveying Applications. In Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, OR, September 2011.

[84] Skaloud, J., Parés, M.E., Blazquez, M., Colomina, I. 2016. A method of airborne gravimetry by combining strapdown inertial and new satellite observations via dynamic networks. In International Symposium on Earth and Environmental Sciences for Future Generations (pp. 111-122). Springer, Cham.

[85] Soloviev, A., 2008. Tight Coupling of GPS, Laser Scanner, and Inertial Measurements for Navigation in Urban Environments. In Proceedings of IEEE/ION PLANS 2008. May 6 - 8, 2008. Monterey, California.

[86] Sturza, M. 1983 GPS Navigation Using Three Satellites and a Precise Clock. Navigation, 30(2), 146-156.

[87] Swaminathan, B., 2014. Vibration Mitigation on Engine Test Stands Using Conventional Analysis Techniques. Topics in Modal Analysis I, Volume 7. Conference Proceedings of the Society for Experimental Mechanics Series, 2014.

[88] Teunissen, P.J.G., 2008. On a stronger-then-best property for best prediction. Journal of Geodesy, 82(3), 167-175.

[89] Ting, J.A., Theodorou,E., Schaal, S., 2007. A Kalman Filter for robust Outlier Detection. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. San Diego, CA, USA

[90] Titterton, D.H., Weston, J.L., 1996. Strapdown Inertial Navigation Technology. IEE (Institution of Electrical Engineers), pp. 464, UK.

[91] Vadlamani, A K., 2010. Airborne Laser Scanner Aided Inertial for Terrain Referenced Navigation in Unknown Environments. PhD Thesis, Electrical Engineering Department, Ohio University, Ohio, USA.

[92] Waegli, A., Guerrier, S., Skaloud, J.,2008. Redundant MEMS-IMU integrated with GPS for Performance Assessment in Sports. In Proceedings of IEEE/ION PLANS 2008, May 6 - 8, 2008. Monterey, California.

[93] Waegli, A., Skaloud, J., Guerrier, S., Parés, M.E., Colomina, I., 2010. Noise Reduction and Estimation in Multiple Micro-Electro-Mechanical Inertial Systems. Measurement Science and Technology, 21(6), 065201.

[94] Walter, M.R., Eustice, R.M., Leonard, J.J., 2007. Exactly sparse extended information filters for feature based SLAM. The International Journal of Robotics Research, 26(4), 335-359.

[95] Wis, M., Colomina, I., 2015. Dynamic Stochastic Modeling of Inertial Sensors for INS/GNSS Navigation. In Proceedings of ION GNSS+ 2015, Tampa (USA), 2015 September 14-18.

[96] Yuksel, Y., 2009. A New Autoregressive Error Modeling Method, Based on Wavelet Decomposition for MEMS Inertial Sensors. In Proceedings of the 22nd International Meeting of the Satellite Division of The Institute of Navigation, Savannah (USA), 2009 September 22-25.

[97] Zhu, N., Marais, J., Bétaille, D., Berbineau, M., 2018. GNSS position integrity in urban environments: A review of literature. IEEE Transactions on Intelligent Transportation Systems, 19(9), 2762-2778.

[98] Zumberge, J.F., 1997 Precise point positioning for the efficient and robust analysis of GPS data from large networks. Journal of geophysical research: solid earth, 102(B3), 5005-5017.