

Optimization and Real-Time Aspects of Aircraft Model-Based Navigation

Présentée le 15 décembre 2022

Faculté de l'environnement naturel, architectural et construit
Laboratoire de topométrie
Programme doctoral en robotique, contrôle et systèmes intelligents

pour l'obtention du grade de Docteur ès Sciences

par

Gabriel François LAUPRÉ

Acceptée sur proposition du jury

Dr R. Boulic, président du jury
Dr J. Skaloud, directeur de thèse
Prof. T. Moore, rapporteur
Dr A. Beyeler, rapporteur
Prof. A. Geiger, rapporteur

Keep your eyes on the stars,
and your feet on the ground.
— Theodore Roosevelt

To my future self...

Acknowledgements

A thesis is far from been done alone and I would like to express my sincere appreciation to everyone who helped me. In particular,

- To my supervisor, Prof. Dr. Jan Skaloud, for his extraordinary supervision and guidance throughout all the stages of my studies and his always-positive attitude to look at the good side of every aspect. To Prof. Bertrand Merminod for accepting me to pursue my Thesis with his laboratory.
- To my friends and colleagues, Dr. Mehran Khaghani, Dr. Philipp Clausen and Dr. Emmanuel Clédat who helped me start my research on the topic of autonomous navigation, which was new for me, and provided great assistance whenever I needed it.
- To Jesse Lahaye and Aman Sharma for revisiting some parts of this thesis.
- To my fellow colleagues and staff of the Geodetic Engineering Laboratory at Ecole Polytechnique Federale de Lausanne, for their assistance and remarkable friendship.
- To my family for their support and encouragement.

This research was supported by the Swiss Commission for Technology and Innovation (CTI) under the name *VDM2NAV*, and the innovation project number *25800.1 PFIW-IW* (07.2017 - 04.2021). The Department of Defense, Protection and Sport (DDPS) represented by armasuisse, Science and Technology founded part of this research (03.2017 - 11.2019) three years (contract numbers: 8003514953 (2017), 8003518612 (2018), 8003522302 (2019)). In 2020, the project was extended by one year (contract number: 8003526614) after the accepted proposal of the candidate. The project ended on November 2020. Numerous elements presented in the thesis are parts of both projects which I am grateful and most obliged.

Lausanne, August 2, 2022

G. F. L.

Abstract

The Vehicle Dynamic Model (VDM) based navigation of fixed-wing drones determines the airborne trajectory in conjunction with Inertial Measurement Unit (IMU) sensors. Without Global Navigation Satellite Systems (GNSS) signals, this method estimates navigation quantities with substantially better quality than the conventional kinematic inertial-based approach. The research focuses on the following key aspects: i) determination/calibration of aerodynamic model parameters using flight data only, ii) real-time implementation of the estimation/navigation scheme during different flight phases, and iii) experimental evaluation of two types of drones.

After the introduction related to aerodynamic modeling and estimation aspects, a novel calibration of aerodynamic coefficients from flight data is proposed for their coarse and fine estimation. The first methodology is proposed as an original consecutive application of three linear estimators for the wind, aerodynamic moments, and force parameters, respectively. The information conveyed with each measurement is used within a Schmidt-Kalman filter according to a heuristic approach to observability (Grammian). The most observable parameters are thus updated, mostly in relation to dynamic maneuvers, to obtain a sufficiently good estimate, which is further improved in the second stage. There, the parameters are refined either via VDM-based filtering or with an optimal smoother. An off-line estimation is preferred for a not-yet-calibrated drone, as it can benefit from precise GNSS observations and attitude updates obtained either by a conventional combination of Inertial Navigation System (INS) and GNSS and/or via photogrammetry.

The real-time model-based navigation is then implemented within a companion computer embedded in the payload of a prototype drone. The information flow between sensors, autopilot, and computer is handled via nodes of the Robotic Operation System (ROS). Following a sensitivity analysis on the quality of time-tagging between the sensors and the control commands, the open-source autopilot is modified so that the required information on the actuators is expressed in GNSS time to within less than 1 ms. The Schmidt-Kalman implementation is proposed to manage the different phases of flight, from initialization, to nominal, and interrupted reception of GNSS signals.

The implementation and its navigation performance are validated in several flights using MEMS-inertial sensors of different quality for the prototype drone with a conventional shape,

Acknowledgements

and for a commercial delta-wing profiled drone. After a few minutes of navigation with GNSS accounting for the possible re-estimation of parameters, autonomous VDM/IMU-based navigation outperforms that based solely on IMU and pressure sensors by up to five times in terms of absolute accuracy. These investigations and findings raise new directions for further development and the use of model-based navigation for downstream applications, such as wind estimation and operations in unstable GNSS-signal environments.

Keywords: Vehicle Dynamic Model, Autonomous Navigation, Real-Time, Schmidt-Kalman Filter, Aerodynamic Coefficient Calibration

Résumé

La navigation basée sur le modèle dynamique d'un véhicule (VDM) de drones à voilure fixe est utilisée pour déterminer leur trajectoire aérienne en conjonction avec des centrales inertielles (IMU). En l'absence de signaux GNSS (Global Navigation Satellite Systems), cette méthode estime les solutions de navigation avec une qualité sensiblement meilleure que l'approche conventionnelle basée sur la cinématique inertielle. La recherche se concentre sur les aspects clés suivants : i) détermination/calibrage des paramètres du modèle aérodynamique en utilisant uniquement les données de vol, ii) mise en œuvre en temps réel d'un schéma d'estimation/navigation au cours des différentes phases de vol, et iii) évaluation expérimentale sur deux types de drones.

Après une introduction liée aux aspects de modélisation et d'estimation aérodynamiques, une calibration des coefficients aérodynamiques à partir des données de vol est proposée pour leur estimation grossière et raffinée. La première méthodologie est proposée comme une application consécutive originale de trois estimateurs linéaires pour les paramètres de vent, de moments aérodynamiques et de force. L'information véhiculée avec chaque mesure est utilisée au sein d'un filtre de Schmidt-Kalman selon une approche heuristique de l'observabilité (Grammian). Les paramètres les plus observables sont ainsi mis à jour, le plus souvent en relation avec des manœuvres dynamiques, pour obtenir une estimation suffisamment bonne, qui est encore améliorée dans la seconde étape. Ensuite, les paramètres sont affinés soit via un filtre basé sur VDM, soit avec un smoother optimal. Une estimation en post-traitement est préférée pour un drone encore non calibré, car elle peut bénéficier d'observations GNSS précises et de mises à jour d'attitude obtenues soit par combinaison conventionnelle du système de navigation inertielle (INS) avec GNSS et / ou via photogrammétrie.

La navigation basée sur VDM est ensuite mise en œuvre en temps réel au sein d'un ordinateur intégré dans la charge utile d'un drone prototype. Le flux d'informations entre les capteurs, le pilote automatique et l'ordinateur est géré via des nœuds du Robotic Operation System (ROS). Suite à une analyse de sensibilité sur la qualité du marquage du temps entre les capteurs et les commandes de contrôle, le pilote automatique open source est modifié pour que les informations requises sur les surfaces de contrôle du drone soient exprimées en temps GNSS à moins de 1 ms près. L'implémentation de Schmidt-Kalman est proposée pour gérer les différentes phases de vol, depuis l'initialisation, la réception nominale et interrompue des signaux GNSS.

Acknowledgements

La mise en œuvre et les performances de navigation sont validées sur plusieurs vols à l'aide de capteurs MEMS-inertiel de qualité différente pour le drone prototype qui a une forme conventionnelle, et pour un drone commercial profilé aile delta. Après quelques minutes de navigation avec les signaux GNSS permettant d'éventuelles réestimations des paramètres, la navigation autonome basée sur VDM/IMU surpasse celle basée uniquement sur les IMUs et les capteurs de pression jusqu'à cinq fois en termes de précision absolue. Ces investigations et découvertes ouvrent des directions pour de futures développements et l'utilisation de la navigation basée sur des modèles dynamiques pour de nouvelles applications, telles que l'estimation du vent et les opérations dans des environnements de signaux GNSS instables.

Mots clefs : Modèle Dynamique d'un Vehicle, Navigation Autonome, Temps Réel, Filtre de Schmidt-Kalman, Calibration des Coefficients Aérodynamiques

Zusammenfassung

Eine auf dem Vehicle Dynamic Model (VDM) basierende Navigation von Starrflügler-Drohnen wird zur Bestimmung der Flugbahn in Verbindung mit Inertialsensoren eingesetzt. In Abwesenheit von Globales Navigationssatellitensystem (GNSS)-Signalen schätzt diese Methode die Navigationsgrößen mit einer wesentlich besseren Qualität als der herkömmliche kinematische, auf Trägheitssensoren (IMU) basierende Ansatz. Diese Forschungsarbeit konzentriert sich auf die folgenden Hauptaspekte: i) Bestimmung / Kalibrierung der aerodynamischen Modellparameter ausschließlich anhand von Flugdaten, ii) Echtzeit-Implementierung des Schätz-/Navigationsschemas während verschiedener Flugphasen und iii) experimentelle Auswertung an zwei Drohnentypen.

Nach der einer Einführung in die aerodynamischen Modellierungs- und Schätzungsaspekte wird eine neuartige Kalibrierung der aerodynamischen Koeffizienten aus Flugdaten für deren Grob- und Feinschätzung vorgeschlagen. Die erste Methode wird als originelle, aufeinanderfolgende Anwendung von drei linearen Schätzern für die Parameter Wind, aerodynamisches Moment bzw. Kraft vorgeschlagen. Die Informationen, die mit jeder Messung einhergehen, werden in einem Schmidt-Kalman-Filter gemäß einer Heuristik zur Beobachtbarkeit (Grammian) verwendet. Die am besten beobachtbaren Parameter werden auf diese Weise aktualisiert, meist in Bezug auf dynamische Manöver, um eine ausreichend gute Schätzung zu erhalten, die in einer zweiten Stufe weiter verbessert wird. Dort werden die Parameter entweder durch eine VDMbasierte Filterung oder eine optimale Glättung verfeinert. Für eine noch nicht kalibrierte Drohne wird eine Offline-Schätzung bevorzugt, da sie von den präzisen GNSS-Beobachtungen und Lageaktualisierungen profitieren kann, die entweder durch herkömmliches INS/GNSS und/oder durch Photogrammetrie gewonnen werden.

Die modellbasierte Echtzeit-Navigation wird dann in einem Begleitcomputer implementiert, der in die Nutzlast einer Prototyp-Drohne eingebettet ist. Der Informationsfluss zwischen Sensoren, Autopilot und Computer wird über Knotenpunkte des Robotic Operation System (ROS) abgewickelt. Im Anschluss an eine Sensitivitätsanalyse zur Qualität des Time-Tagging zwischen den Sensoren und den Steuerbefehlen wird der Open-Source-Autopilot so modifiziert, dass die erforderlichen Informationen über die Aktoren in GNSS-Zeit besser als 1 ms umgesetzt werden. Die Schmidt-Kalman-Implementierung wird vorgeschlagen, um die verschiedenen Flugphasen zu kontrollieren, von der Initialisierung über den Nominalwert bis hin zur Empfangsunterbrechung der GNSS-Signale.

Acknowledgements

Die Implementierung und ihre Navigationsleistung werden in mehreren Flügen mit MEMS-Inertialsensoren unterschiedlicher Qualität für den Prototyp-Drohne, der eine konventionelle Form hat, und für eine kommerzielle Drohne mit Deltaflügel validiert. Nach einigen Minuten der Navigation mit GNSS unter Berücksichtigung der möglichen Neuschätzung von Parametern ist die autonome VDM/IMU-basierte Navigation bis zu fünfmal besser als die allein auf IMU und Drucksensoren basierende. Diese Untersuchungen und Erkenntnisse eröffnen Wege für die weitere Entwicklung und den Einsatz der modellbasierten Navigation für neue Anwendungen, wie z.B. die Windabschätzung und den Betrieb in einer GNSS-reduzierten Umgebung.

Stichwörter: Dynamisches Modell eines Fahrzeugs, Autonome Navigation, Echtzeit, Schmidt-Kalman Filter, Kalibrierung der Aerodynamischen Koeffizienten

Contents

Acknowledgements	i
Abstract (English/Français/Deutsch)	iii
List of Figures	xv
List of Tables	xxi
List of Listings	xxiii
List of Algorithms	xxv
Conventions and Notation	xxvii
1 Introduction	1
Introduction	1
1.1 Context	1
1.2 Research Objectives	3
1.3 Software Methodology	3
1.4 External Contributions	5
1.5 Thesis Outline	5
I Preliminaries	7
2 Estimation Methods	9
2.1 Linear Least Squares	9
2.1.1 Ordinary and Weighted Least Squares	9
2.1.2 Recursive Least Square	10
2.2 Kalman Filter	11
2.2.1 State Space Dynamic Systems	11
2.3 Extended Kalman Filter	13
2.3.1 Error State Extended Kalman Filter	13
2.4 Optimal Smoothing	15

3	Background of Integrated Navigation with Vehicle Dynamic Model	17
3.1	Rotation Operations	18
3.1.1	Rotation Matrix	18
3.1.2	Differential Rotations	18
3.1.3	Quaternion	20
3.1.4	Main Quaternion Properties	21
3.1.5	Differential Quaternion	23
3.1.6	Rotation Uncertainty	24
3.2	Quaternion-Euler Error State Extended Kalman Filter	24
3.3	Frames Definitions	26
3.3.1	Inertial Frame	26
3.3.2	The Earth Frame	27
3.3.3	Local Level Frame	28
3.3.4	Body Frame	29
3.3.5	Wind Frame	30
3.3.6	Camera and IMU Frames	30
3.4	Navigation Equations	32
3.5	Specificities via Vehicle Dynamic Model	34
3.5.1	Estimation Scheme	36
3.5.2	State Space	37
3.5.3	Observation Models	40
3.5.4	Linearization	42
II	VDM Enhancement	45
4	Estimation Enhancements	47
4.1	Numerical Stability	47
4.1.1	Indicators	48
4.1.2	Scaling	49
4.2	Factorization and Schmidt-Kalman	50
4.2.1	UDU Implementation	50
4.2.2	Partial Updates	51
4.2.3	Initialization	53
4.2.4	GNSS Outage	53
4.3	State Space Reduction	54
5	Aerodynamic Coefficient Estimation	57
5.1	General Calibration Methodology	58
5.2	Approximate Coefficients Determination	58
5.2.1	Estimation Separation	59
5.2.2	Generic Model Structure	60
5.2.3	Estimator I -Wind	61

5.2.4	Estimator II - Moments	62
5.2.5	Estimator III - Forces	63
5.2.6	Observability and Uniqueness	63
5.3	Coefficients Refinement	66
5.3.1	Calibration with External 'pose'	66
5.3.2	Attitude References from Photogrammetry	67
 III Real-time framework and setup		 71
 6 Platforms and Experiments		 73
6.1	Platforms	74
6.2	Payloads	75
6.2.1	"IGN-GECKO"	75
6.2.2	"SODA-GECKO"	77
6.2.3	"SODA-STIM"	77
6.3	Sensors and Subsystems	78
6.3.1	Overview	78
6.3.2	In the UAV	80
6.3.3	Devices on the Ground	84
6.3.4	Portable Weather Station	85
6.4	Flights	87
6.4.1	VDM Concept Validation with Experimental Data	87
6.4.2	IGN-GECKO	89
6.4.3	SODA-GECKO	89
6.4.4	SODA-STIM	90
6.4.5	eBeeX Campaign	92
 7 Real-Time Implementation		 93
7.1	Data in GPS-Time	94
7.2	ROS Architecture in the Embedded Computer	94
7.3	VDMc - Software	97
7.3.1	Operation Modes	97
7.3.2	Architecture	98
7.3.3	Automatic Linearization in C++	99
7.3.4	Asynchronous Observations Handler	99
7.3.5	Dynamic State Reduction	102
7.4	Flights Phases	103
 IV Results and Analysis		 109
 8 Determination of Coefficients		 111
8.1	Effect of Flight Dynamics and Initialization	112

Contents

8.1.1	Trajectory Definition	112
8.1.2	Flight Simulation	114
8.1.3	Sensor and VDM Parameters Errors	114
8.1.4	Parameters Estimation per Segment	115
8.1.5	Sequence of Maneuvers	117
8.1.6	Effect of Initial VDM Parameter Values	119
8.2	Identification of Coefficients with Observability Criteria	121
8.2.1	Observability Grammian	122
8.2.2	Aerodynamic Model-Parameters	125
8.2.3	Parameter Initialization and Uncertainty	126
8.3	Refinement Methods	129
8.3.1	Attitude Update via Photogrammetry	129
8.3.2	Optimal Smoother	132
8.3.3	Parameter Reduction	132
8.4	Numerical Comparison of Coefficients	133
9	Real-Time Aspects	135
9.1	Sensors Time Delay and Influence of Time-Tagging errors	136
9.1.1	Requirements	136
9.1.2	PX4 Control Command Time-Tagging	142
9.2	VDM Online Demonstrator	142
9.2.1	Components	142
9.2.2	Computational Load	144
9.3	Initialization	145
9.3.1	Incorrect Model-Parameter Initialization	145
9.3.2	Initialization Improvement with Partial-Schmidt Update	145
9.4	Wind	147
9.4.1	Wind Estimation in Simulation	147
9.4.2	Wind estimation in real-time	150
10	Autonomous Navigation	153
10.1	Coarse Coefficient Estimation	154
10.1.1	TP2	154
10.1.2	eBeeX	154
10.2	Partial Correction of Navigation States	156
10.3	Comparison to Inertial Coasting	158
10.3.1	Online Demonstrator Campaign	159
10.4	Reduced Model	163
V	Conclusion and Recommendations	167
11	Conclusion	169
11.1	Contributions	169

11.1.1 Conceptual	169
11.1.2 Engineering	170
11.2 Limitations	171
11.3 Further work	171
A Additional Analysis and results	173
A.1 Numerical Conditioning	173
A.2 Numerical Integration	175
A.2.1 Attitude with Quaternions	175
A.2.2 Small Integration Time	176
A.3 VDM vs Inertial Coasting for the Different Payloads	177
A.3.1 eBeeX with "GECKO" Payload	177
A.3.2 TP2 with "IGN-GECKO" Payload	178
A.3.3 TP2 with "SODA-STIM" Payload for the WMF Campaign	179
A.4 Coefficients with Photogrammetry Aiding	180
A.5 Fixed VDM Parameters	183
A.6 Aerodynamic Coefficient Refinement Thanks to 'pose' Sensor	184
A.6.1 VDM-based Autonomous Positioning Performance	185
A.7 Simulation VDM Parameters Correlations	186
A.8 Experimental Validation of Wind Estimation	187
A.9 WMF TP2 and eBeeX Forces and Moment Residual with Flight STIM6	188
A.9.1 TP2 - Moment Residuals	188
A.9.2 TP2 - Force Residuals	189
A.9.3 eBeeX - Moment Residuals	190
A.9.4 eBeeX - Forces Residuals	191
B Theoretical supplements	193
B.1 Static and Dynamic Pressure Conversion	193
B.1.1 From Airspeed to Ground Velocity	193
B.1.2 From Static Pressure to Altitude	194
B.2 Extended Kalman Filter	194
B.2.1 From Continuous to Discrete KF - Matrix Derivation	194
B.2.2 Transition Matrix and Covariance Noise Update Approximation	195
B.3 Additional Material on Angles	196
B.3.1 ODE for Attitude in Euler	196
B.3.2 Euler Angle and Quaternion Conversion	197
B.3.3 Euler Angle to Quaternion	197
B.3.4 Quaternion to Euler angle	197
C Hardware	199
C.1 Inter-Devices Communication	199
C.1.1 GNSS Receiver to Autopilot (GNSS to AP)	199
C.1.2 GNSS Receiver to Gecko4NAV / SentiBoard	199

Contents

C.1.3	Gecko4NAV / SentiBoard to Embedded Computer (IMU to PC)	199
C.1.4	Image Acquisition (AP to CAM to PC to GNSS)	200
C.2	Sensor	200
C.2.1	IMU Error Statistic	200
C.2.2	Stochastic Models for GNSS Noise	201
C.3	TP2	201
C.3.1	Geometric Parameters	201
C.3.2	Wind Surface Determination	203
C.4	eBeeX	205
C.4.1	"eBee-GECKO" Payload	205
C.4.2	eBeeX Aerodynamic Coefficients	206
C.5	Weather Station	207
C.5.1	Weather Station Schematic	207
C.5.2	Weather Station Layout	208
C.6	SODA-STIM Payload Schematic	209
D	Software	211
D.1	Software Communication	211
D.2	ROS Nodes	214
D.2.1	Start all Nodes	214
D.2.2	Node Subscribers and Publishers	215
D.3	VDMc	218
D.3.1	VDM C++ class architecture	218
D.3.2	VDMc Main Time Loop	220
D.3.3	Flight Phases	220
D.3.4	Configuration	223
D.3.5	Sensors Classes	225
D.3.6	The FilterTimer Class	225
D.3.7	The Main Loop	226
D.3.8	Implementation of Timers	227
D.3.9	Specialized Timers	229
D.3.10	Thorton Bierman Implementation for UDU Factorization	231
	Bibliography	243
	Abbreviations	247
	Curriculum Vitae	249

List of Figures

1.1	Methodology for the development of the real-time VDM-based navigation system, its calibration and optimizations	4
2.1	Different estimation phases depending on the availability of observations	15
3.1	Inertial ECI frame in black, rotating ECEF frame in dashed-blue	26
3.2	Earth <i>e</i> -frame and local (ENU) <i>l</i> -frame	28
3.3	Rotation around the three body axes	30
3.4	Local level, body, and wind frames with airspeed \mathbf{V} , wind velocity \mathbf{w} , and UAV velocity \mathbf{v} , adapted from [47]	31
3.5	Body, camera and IMU frames	31
3.6	<i>TP2</i> control surfaces, positive angular rotations (black) around body frame axis, forces (blue)	36
3.7	eBeeX control surfaces and positive angular rotations around body frame (black)	37
3.8	VDM-based navigation filter architecture ($\tilde{\mathbf{x}}_k \equiv \hat{\mathbf{x}}_{k k-1}$), adapted from [47]	38
5.1	Aerodynamic coefficient calibration flowchart	58
5.2	Calibration procedure as a sequence of three decoupled estimators	61
5.3	Coefficient estimation algorithm flowchart with three estimators for the wind (plus Pitot scale factor), moments, and forces	65
5.4	Steps required to obtain attitude updates in the body-frame using the precise pre-processing trajectory treatment and photogrammetry	68
6.1	Fixed-wing platform with 5 control surfaces (left): the two versions of the TOPOplane (a) version-1 (top) and version-2 (bottom), and the <i>eBeeX</i> from senseFly (b)	74
6.2	(a) "IGN-Gecko" with a high-quality camera (digiCAM), board with 2 redundant Inertial Measurement Unit (IMU) and embedded 32-bit micro-computer, (b) "SODA-Gecko" with SODA camera, 4 redundant IMUs, and embedded 64-bit computer UpBoard, (c) "SODA-STIM" with an UpBoard, SODA camera, <i>STIM-318</i> IMU, SentiBoard and Dahu board	76
6.3	A list of hardware (HW) and software (SW) modules	79
6.4	General connection between hardware components	80
6.5	Radio Control (RC) used by the <i>TP</i> operator	84

List of Figures

6.6	Ground Control Station set up with the operator's computer running QGround-Control (QGC) and the telemetry transceiver	84
6.7	(a) Portable Weather Station main module and (b) acquisition set-up with tripod	85
6.8	Flight performed in August 2017 with the <i>TP1</i> under the names (a) <i>20170821_nx5id1</i> , and (b) <i>20170822_nx5id1</i>	87
6.9	Experimental flight references (blue): a) <i>IGN8</i> , b) <i>IGN7</i> , c) <i>IGN6X</i> and d) <i>IGN6u</i> , beginning of the trajectory (red triangle)	89
6.10	Experimental flights references (blue): a) <i>STIM5</i> , b) <i>STIM6</i> , c) <i>STIM7</i> , d) <i>STIM_12</i> , and e) <i>STIM_13</i> , beginning of the trajectory (red circle)	91
6.11	<i>eBeeX</i> flight trajectories with <i>CF_eBee_756</i> (a) used as calibration and <i>AF_eBee_652</i> (b) as application flights	92
7.1	ROS architecture of the embedded PC	95
7.2	<i>VDMc</i> logo	97
7.3	Modes of the Vehicle Dynamic Model (VDM) C-based operation	97
7.4	Simplified design of <i>VDMc</i> I/O	98
7.5	Definition of the velocity and angular rate differential equations (Eq. 3.57 and Eq. 3.49) using <i>Mathematica</i> environment	99
7.6	Linearization flowchart	100
7.7	Include of automatically-generated functions for the Global Navigation Satellite System (GNSS) position model <i>h_func</i> and Jacobian <i>linH_func</i> methods . . .	101
7.8	The EKF processing the GNSS and CC data as they arrive computing new states (a) returning in the past EKF states corresponding to the first new observation (b) and back-propagating the states with the new corrections (c)	102
7.9	Flight phases and corresponding software configuration	104
7.10	(a) Launch of the <i>TP2</i> against the wind, (b) first second of the flight, and (c) <i>GiiNav</i> (d) <i>VDMc</i> initialization	105
7.11	2D (a) and 3D (b) dynamic maneuvers during the first 6 minutes of the flight <i>ebeeX_756</i> for aerodynamic coefficient refinement	106
7.12	Successful manual landing on a wheat field in Echendans with the <i>TP2</i>	107
8.1	Simulation workflow in self-calibration	112
8.2	Correlation matrix P between state-vector elements at the end of a maneuver: a) - Level flight, b) - Controlled orbit c) - Infinity loop, d) - Combination of segments. The wind is not estimated and set to zero (no correlation)	117
8.3	Combination of maneuvers. The resulting trajectory (blue) is designed with waypoints (pink crosses) and starts at the red circle	118
8.4	Error estimation evolution versus 1σ during a specific trajectory: a) - Level flight, b) - Combination of maneuvers	118
8.5	Estimation error of the 21 VDM parameters at the end of 3 calibration maneuvers	119
8.6	Methodology to test the influence of incorrect initial VDM parameters	119
8.7	Average VDM parameter errors at the end of different trajectories after their varying initial corruptions (a) from 10 to 100% (b) zoomed from 10 to 50% . . .	120

8.8	Experimental pipeline	122
8.9	Tracking Eigen vector E _v of observability grammian	123
8.10	Maximum closeness in wind observability	123
8.11	Evolution of Eigen Values associated to the basis	124
8.12	(top) Juxtaposition of force-y computed using IMU and calibrated model and their residuals (bottom) for <i>TP2</i> (a), and force-x for <i>eBeeX</i> (b)	127
8.13	Initial chosen points \mathbf{x}_0 for the moment estimator	128
8.14	Camera orientation (triangles) and terrain. After [105]	130
8.15	Percentage of change for each VDM parameter with and without the use of attitude reference measurements	131
8.16	State correlation matrix (<i>CF_i8</i> with highlighted \mathbf{x}_p after a) 30 s of filtering; b) at the end of forward-filtering; and c) optimal smoothing	132
9.1	Time-tagging of the emitted control commands with delays	137
9.2	Flowchart of simulations to assess the effect of time-tagging errors	137
9.3	Reference trajectory (dashed-blue) and VDM-based navigation solution (green) with correct control command time-tagging during 3 minutes of GNSS outage	138
9.4	Maximum position error for all runs with motor time-tagging errors	139
9.5	Maximum position error for all runs with servo time-tagging errors	140
9.6	Reference trajectory (dashed-blue) and VDM navigation solution (green) over 3 minutes of GNSS outage with 10 [ms] of fixed and random delay error for the servo time-tagging	141
9.7	(a) Screen capture of QGC during a mission, displaying two tracks on map: navigation solution computed by PX4 autopilot (in red), navigation solution computed by Inertial Navigation System (INS)/GNSS <i>GiNav</i> (in blue). VDMc trajectory not displayed. (b) Widget to send custom command to the Unmanned Aerial Vehicle (UAV) embedded computer and display received telemetry messages	143
9.8	Max. position error during the first 200 s after initialization without and with partial-Schmidt ($T_{ini}=100$ s) for the three application flights	146
9.9	Evolution of horizontal position error (magnitude) after initialization for $T_{ini} = [0, 100]$ s in flights <i>AF_i7</i> (left) and <i>AF_i6x</i> (right)	147
9.10	Simulated 15-minutes trajectory	148
9.11	Wind estimation during the (a) first 20 seconds and (b) the whole trajectory of a simulated trajectory for two different initialization scenarios: (i) approximated initial values in red, and (ii) with zeros in green with respect to the reference wind (blue)	149
9.12	Wind estimation for the flight <i>STIM_12</i> using WMF (red), the VDM filter (blue) and PX4 (yellow)	150
10.1	<i>TP2</i> flight <i>AF_STIM5</i> (a) with a 2-minute outage after 15 minute, and flight <i>AF_STIM7</i> (b) with outage after 4 minutes and (c) invariance of calibrated VDM to different IMU. GNSS outage starts at the black cross	155

List of Figures

10.2	eBeeX application flight <i>AF_eBeeX_652</i> with 2 minutes outage after (a) 23 and (b) 28 minutes for VDM (black) and INS (red) solution	156
10.3	Max. and median horizontal errors without (dark grey) and with (light grey) partial updates during 2-min GNSS outages	156
10.4	2 minute GNSS outage without (red dashes), with (solid green) partial updates, on (a) <i>AF_i7</i> , (b) <i>AF_i6x</i> , (c) and (d) <i>AF_i6u</i> with the reference trajectory (dotted blue)	157
10.5	Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for INS only with <i>STIM-318</i> versus <i>ADIS-16475</i> for flight <i>AF_STIM7</i>	158
10.6	Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for flights <i>STIM_12</i> and <i>STIM_13</i> with 47 and 26 states, and INS	161
10.7	<i>STIM_14</i> : VDM (yellow) vs INS-based navigation solution (blue) compared with the reference (red) during GNSS outages of about 5 minutes after (a) 9 minutes and (b) 18 minutes	162
10.8	<i>STIM_15</i> : VDM (yellow) vs INS-based navigation solution (blue) compared with the reference (red) during GNSS outages of about 8 minutes after 6 minutes, zoomed version (b)	164
10.9	Max. and median horizontal errors during 2 minutes of GNSS outages with the full (dark grey) and reduced (light grey) models	165
A.1	Order of magnitude (right-axis) of the initial covariance matrix elements \mathbf{P}_0 of a state-vector components (left-axis): a) before, b) after scaling	173
A.2	Condition number of the covariance matrix with and without scaling of lat/long and propeller speed n	174
A.3	Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for flights <i>eBeeX_756</i> <i>eBeeX_652</i> using VDM and INS	177
A.4	Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for VDM and INS	178
A.5	INS/GNSS based navigation performance under 2-min GNSS outage for the flights (a) <i>AF_i7</i> , and (b) <i>AF_i6x</i>	179
A.6	For flight <i>AF_i6u</i> , (a) Last 400 seconds of VDM (green), INS (red), reference (blue) during a GNSS outage of 6 minutes: (a) 2D, (b) 2D-zoomed, (c) error(t)	180
A.7	Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for VDM and INS for flights <i>AF_STIM5</i> and <i>CF_STIM6</i>	181
A.8	Comparison during 2 minute GNSS outage of VDM-based navigation performance using (a) the <i>CF_i8</i> trained aerodynamic coefficients with camera attitude references (green) or without (red), and the application flights (b) <i>AF_i7</i> and (c) <i>AF_i6X</i>	182
A.9	VDM-based navigation with 27 states	183
A.10	Last 130 s of VDM-based navigation with 2 min of GNSS outage after 10 min (a) and 20 (b) min of flight with the aerodynamic coefficients estimated with the 'pose' method (Sec. 5.3.1)	186

A.11 Anemometer heading resolution	187
A.12 Moment residual for each axis	188
A.13 Force residual for each axis	189
A.14 Moment residual for each axis	190
A.15 Force residual for each axis	191
C.1 Image acquisition events	200
C.2 Flight control surfaces, wing span b and surface S of a fixed-wing UAV: <i>TOPOPlane version 2</i> . The dense mesh of the platform was created with the help of the Laboratory of Intelligent System scanner system.	202
C.3 The cord \bar{c} and the angle of attack α	202
C.4 Propeller diameter D for (left [116]) a full blade, and (right [117]) an half-folding propeller	203
C.5 A 3D point cloud reconstruction (a) and orthophoto with surface measurements (b) of the wing surface.	204
C.6 3D printed support placed in the eBeeX payload bay with a special board (Gecko4Nav) containing 2 MEMs-IMUs as seen from front (left) and bottom (right).	205
D.1 Information flow between SW/HW components	212
D.2 (a) QGround Control software with a flight trajectory in manual mode and (b) Pair of transceiver module for the UAV (left), and Ground station	213
D.3 Messaging protocols between the embedded computer, autopilot and the Ground Control Station (GCS)	214
D.4 VDMc C++ version 1 (2020) class architecture	218
D.5 VDMc version 2 (2022) C++ class architecture	219
D.6 Main EKF Loop of VDMc	220
D.7 CPU load before (47 states) and after (26 states) the dynamic state reduction	224

List of Tables

2.1	Discrete Kalman Filter steps	12
2.2	Discrete Error-State Extended Kalman Filter steps	15
2.3	Backward Discrete Error-State Extended Kalman Filter steps	16
3.1	Aerodynamic coefficients for the forces and moments for the two platforms: <i>TP2</i> and <i>eBeeX</i>	35
4.1	Adapted scaled matrix for the discrete Extended Kalman filter steps	49
4.2	ESEKF with <i>UDU</i> factorization	51
6.1	General characteristics of the UAVs	74
6.2	Payload characteristics	76
6.3	GNSS receivers used in chronological order	82
6.4	Weather station specifications	86
6.5	Flight summaries	88
8.1	List of results for coefficient calibration	112
8.2	Categorization of simulated trajectories into segments (7 types)	113
8.3	Initial navigation state uncertainties	115
8.4	Group of VDM parameter error per trajectory	116
8.5	Number of successful navigation runs per initial error in VDM parameters per trajectories	121
8.6	<i>CF_i8</i> flight details.	130
8.7	Initial and estimated (filtered) VDM parameters without (GNSS) and with pho- togrammetry (GNSS + Att)	131
8.8	Proposed correlated pairs for model reduction and their linear relations	133
8.9	Coefficients obtained from WMF (Sec. 8.2), attitude observation (Sec. 8.3.1) and smoother (Sec. 8.3.2) methods for the flight <i>CF_i8</i>	133
9.1	List of real-time-related results	136
9.2	Fixed and random delays for motor data	139
9.3	Fixed and random delays for servos data	140
9.4	Three different systems with various sub-models tested with the VDMc software on the embedded computer	144

List of Tables

9.5	Execution time of the prediction and update steps for the different filter size . . .	144
9.6	Comparison of estimated horizontal wind for <i>STIM_12</i> and <i>STIM_13</i> using the WMF method and VDM-EKF	150
10.1	List of autonomous navigation results	153
10.2	Initial state uncertainty	159
10.3	Start times (in minutes) of simulated GNSS outage after takeoff for <i>STIM_12</i> and <i>STIM_13</i> flights (online demonstrator campaign 1) and GNSS outages performed in real-time for the flights <i>STIM_14</i> and <i>STIM_15</i> (online demonstrator campaign 2)	159
10.4	Initial coefficient and after 10 minutes of refinement for flights <i>STIM_12</i> and <i>STIM_13</i>	160
A.1	Start times (in minutes) of 2 minute-long GNSS outage after takeoff for both calibration (<i>eBeeX_756</i>) and application (<i>eBeeX_652</i>) flights	177
A.2	Start times of the 2 minutes GNSS outage within the application flights (in minutes after take-off).	178
A.3	Start times (in minutes) of 2 minute-long GNSS outage after takeoff for <i>STIM_5</i> and <i>STIM_6</i> flights (WMF campaign)	179
A.4	Position error at the end of the 2 minute GNSS outage, 2D and 3D position RMSE and % of time with hor. error under 150m	183
A.5	Percentage of time within 2 min long autonomous navigation during which the horizontal positioning error stays within the specs represented by 1σ and 3σ . This test is invoked after 10 min and 20 min of flight	185
A.6	Highly correlated VDM parameter pairs for particular trajectories	186
A.7	2D static wind estimation comparison - mean values over flight duration	187
B.1	Higher order approximation of Φ and \mathbf{Q}_k	196
C.1	STIM318 noise characteristics	201
C.2	ADIS-16475 noise characteristics	201
C.3	NavChip noise characteristics	201
C.4	Typical uncertainty values for Single Point Positioning (SPP) and Post-Processed Kinematic (PPK) GNSS solution	202
C.5	Coefficients obtained with the WMF and 'pose' sensor method	206

Listings

7.1	Delayed IMU observation	101
7.2	Rejection of incorrect time-tagged GNSS observations	101
D.1	surrey topic	221
D.2	airData topic	221
D.3	Navigation states initialization after 2 minutes	221
D.4	Actuator initialization	223
D.5	Dynamic state reduction log	223
D.6	VDMc initialization with loading of the configuration files	224
D.7	VDMc loading default initial values (covariance matrix)	224



List of Algorithms

1	UDU Decomposition	231
2	Thornton propagation	232
3	Bierman observation update	233

Conventions and Notation

Estimation Notations and Symbols

F	dynamic matrix
Φ	transition matrix
G	process noise shaping matrix
$h()$	observation model equation
H	measurement design matrix
K	Kalman Gain
L	input coupling matrix
P	covariance matrix of x
Q	process noise covariance matrix of x
R	measurement noise covariance
x	refers to a parameter (state) vector
$\delta\mathbf{x}$	error-state vector
$\hat{\mathbf{x}}$	updated quantity of x
$\bar{\mathbf{x}}$	predicted quantity of x
$\mathbf{v}(t)$	measurement noise at time t
$\mathbf{w}(t)$	process noise at time t
z	a observation vector
$E[\mathbf{x}]$	expected value of x
$var[\mathbf{x}]$	variance of x
$cov[\mathbf{x}]$	covariance of x

Integrated Navigation and Geodesy Notations and Symbols

a	semi-major axis of an ellipse
e	eccentricity
h	ellipsoidal height
q	quaternion
\mathbf{r}_{c-b}^b	lever-arm from c to b expressed in b -frame
\mathbf{R}_a^b	rotation matrix from frame a to b
R_m	Earth radius prime vertical of curvature

Conventions and Notation

R_p	Earth radius prime vertical of curvature
\mathbf{x}^b	vector \mathbf{x} expressed in a body-frame
\mathbf{x}^i	vector \mathbf{x} expressed in an inertial-frame
\mathbf{x}^e	vector \mathbf{x} expressed in an Earth-frame
\mathbf{x}^l	vector \mathbf{x} expressed in a local-frame
ϕ	latitude
λ	longitude
$[\Psi]_x$	skew-matrix of Ψ
Ω_{ab}^b	skew-matrix of angular-rate in a -frame w.r.t. the b frame, expressed in- b frame

VDM specific Notations and Symbols

α	angle of attack
β	side-slip angle
b	wing span
\bar{c}	mean aerodynamic chord
$C_{...}$	model coefficient
δ_a	aileron deflection
δ_e	aileron elevator
δ_r	aileron rudder
D	propeller diameter
\mathbf{I}^b	matrix of inertia
\bar{q}	dynamic pressure
m	mass
n	propeller speed
p	static pressure
q	dynamic pressure
ρ	air density
S	wing surface
\mathbf{f}^b	specific forces
F_T^b	thrust force
F_x^w	drag force
F_y^w	lateral force
F_z^w	lift forces
\mathbf{M}^b	specific moments
M_x^b	aerodynamic moment around x-axis
\mathbf{x}_e^l	position vector of ellipsoidal coordinate expressed in the l -frame
\mathbf{v}_e^l	velocity vector with ellipsoidal coordinate expressed in the l -frame
\mathbf{q}_b^l	attitude vector of the body with respect to the local-frame
$\boldsymbol{\omega}_{ib}^b$	angular velocity of body with respect to inertial frame expressed in the body-frame
\mathbf{x}_n	navigation states
\mathbf{x}_p	VDM parameter states

\mathbf{x}_a	actuator states
\mathbf{x}_w	wind states
\mathbf{x}_e	IMU error states
\mathbf{V}	air speed vector
\mathbf{w}	wind velocity
w_{ie}	mean Earth angular velocity of 7292115.0×10^{-5} rad/s

Mathematical Conventions and Symbols

$f()$	refers to a function
$f(k)$	value of the function $f()$ at step k
\mathbf{x}	refers to a vector
x_k	k th value of a sequence
$x(t)$	realization of \mathbf{x} at time t
\dot{x}	time derivative of x
\ddot{x}	second time derivative of x
\mathbf{X}	refers to a matrix
\mathbb{H}	quaternion space
\mathbf{I}	identity matrix
\mathbf{J}	Jacobian
\mathbf{x}^T	refers to the transpose vector of \mathbf{x}
$(R)^{-1}$	inverse of the matrix \mathbf{R}
$\frac{\partial f(x)}{\partial x}$	derivative of $f(x)$ with respect to x
$\left. \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right _{\mathbf{x}=\hat{\mathbf{x}}}$	derivative of $f(x)$ with respect to x , evaluated at \hat{x}
\approx	approximately equal to
\neq	is different than
\triangleq	defines
\otimes	quaternion product
$[\boldsymbol{\omega}]_{\times}$	quaternion equivalent of $\boldsymbol{\omega}$
$ \mathbf{x} $	cardinality of x
$\ \cdot\ $	norm
$\mathbb{R}^{l \times n}$	number set dimension $l \times n$

1 Introduction

1.1 Context

The expansion of applications utilizing small Unmanned Aerial Vehicle (UAV)s has been unprecedented in the last few years. No aviation sector has experienced such rapid technological evolution in both hardware and software. The drone industry is expected to reach 10% of the aerial market in the next ten years, representing several tens of billion dollars annually [1].

Construction [2], building inspections [3], agriculture field monitoring [4], mapping operations [5], or emergency equipment delivery [6] are a few of the applications among many that employ unmanned aerial vehicles. An increasing number of interdisciplinary research fields are beginning to investigate the utility of drone technology, and the number of publications related to drones or their application has increased by a factor of 10 between 2013 and 2018 and is still growing [7]. With an increase in UAV utilization comes the need to perform autonomous missions during which navigation accuracy and safety are of great importance. Most of the aircraft operated outdoors rely on Global Navigation Satellite System (GNSS) signals to guarantee positioning quality and their combination with Inertial Navigation System (INS) to determine attitude. Critical situations arise when one of the two systems becomes unavailable, compromising the safety and reliability of missions.

In addition, the increasing number of UAVs operated in public airspace has triggered the adoption of new regulations to manage them effectively. These regulations consider many factors, including safety features such as return-to-home options, which ensure secure landing of UAV in compromised circumstances, for example, low battery, loss of communication, or degradation of weather or navigation performance. Taking into account the latter, when a UAV loses reception of a satellite signal due to interference, spoofing or jamming, or simply physical obstructions, such security procedures are challenged in the absence of additional sensors (such as cameras and lidars) or their reduced perception due to fog, darkness, or absorption.

Due to the aforementioned reasons, the quality of autonomous navigation remains a dominant

research topic, especially for methods aiming for its improvement without additional sensors. A relatively new approach to autonomous navigation that uses a Vehicle Dynamic Model (VDM) in a particular architecture has shown the potential to improve the positioning accuracy of small fixed-wing UAV within GNSS denied or perturbed environments compared to inertial-based navigation [8]. In mapping flights with precise (Real time Kinematic (RTK)/PPK) GNSS positioning, this approach also aims to improve direct orientation (attitude estimation) [9]. This patented architecture [10] was defended in 2018 by Dr. Khaghani in his Ph.D. thesis [11].

A major challenge related to VDM-based navigation is its dependence on a priori knowledge of the aerodynamic characteristics of the platform. The prerequisite for the system is a precise knowledge of the coefficients of the aerodynamic model for each specific aircraft. Only with “reasonably” correct values, will the dynamic process model represent the realistic flight behavior of the platform for navigation (filter) to work correctly; otherwise, it may become unstable and diverge [12].

The determination of some model parameters is straightforward, such as the weight of the aircraft or the surface area of the wing. Wind tunnel tests or Computational Fluid Dynamic (CFD) simulations can be conducted to determine other VDM parameters as presented in [13] for a small UAV. However, these methods are expensive, time-consuming, do not provide all parameters, and, for the latter, require an excellent physical model. Moreover, modification of the platform or its payload requires adaptation of the coefficients. Therefore, a need to obtain such parameters by other means arises. When the initial values for these parameters are adapted from a similar type of aircraft, these can be further calibrated based on sensor observations in flight. This method requires sensor data (IMU, GNSS, images, dynamic and static air pressure) of sufficient quality, flight control commands from the autopilot within a shared (GNSS) time frame, and some (e.g., state space) estimation procedures.

However, it is challenging to estimate the parameters individually due to their implicit correlations [11]. The limitations of the in-flight estimation of VDM parameters during nominal dynamics motivate the investigation of a dedicated calibration procedure.

In addition, a critical emerging challenge with an online calibration procedure is the need to estimate many parameters. The apparent consequence of these characteristics is the growth of computational resource requirements and an increased risk of potential numerical instability. Due to the restricted processing power of an embedded system, a valuable needed improvement is the reduction of the navigation filter’s computational overhead and the implementation of safeguards to improve its numerical stability.

Previous research on VDM-based navigation using a MATLAB-based offline framework motivates the design of an embedded real-time solution that can be utilized to improve dead-reckoning capacity within a GNSS-denied environment. However, such an implementation raises several engineering questions related to real-time constraints e.g., global time synchronization of sensor and flight control data, their real-time access, and system navigation initialization, adaptation of the estimation scheme during GNSS outage to name a few.

1.2 Research Objectives

After the analysis of the current state of the VDM-based navigation framework, the objectives of this research are established as threefold:

Obj. 1: Numerical stability

Optimize the current navigation estimation scheme to a more stable version from a real-time application perspective. The goal is mainly to improve the numerical stability in the filter by (i) adapting factorization algorithms to the VDM-based navigation system, (ii) conditioning the different elements of the filter to reduce the probability of incorrect convergence and build-up of numerical errors, and (iii) reduce the filter size to improve computational efficiency.

Obj. 2: Initialization and calibration

Elaborate on a multi-phase calibration procedure to estimate platform-specific aerodynamic coefficients. The procedure explores the potential of obtaining such estimations by using sensor measurements available in most UAV both in post-processing and online estimation and without employing additional equipment or computer-aided modeling tools (e.g., wind tunnel experiments or CFD). The first phase aims to provide the VDM-based navigation software with the (finest possible) knowledge of initial parameters that are further refined via state space in subsequent calibration phases.

Obj. 3: Real-time Implementation

Develop the concepts and realize their implementation of the first fully embedded real-time prototype of a VDM-based navigation solution on a fixed-wing drone. This involves setting up an environment to gather, synchronize, and time-tag all the required data, and develop support software to process them in the navigation solution correctly. Such an estimation environment must be designed from a real-time perspective to cope with the asynchronous nature of the sensor measurements and flight control commands.

1.3 Software Methodology

Fig. 1.1 illustrates the different frameworks (rectangles) representing a research environment with the work packages (ellipses) describing the research objectives, the adapted and developed methods, the software implementations and the links between them.

Dr. Khaghani developed as part of his thesis [11] a MATLAB-based simulation and post-processing environment. The first part of the software called *Post-Processing (PP)* VDM in Fig. 1.1 has been adapted to include the development of the first two research objectives. The module possesses a simulation environment and a VDM-based navigation estimator based on Extended Kalman Filter (EKF). The simulator can emulate the flight trajectory of a platform defined by its dynamic model. Furthermore, the simulator can produce and save

Chapter 1. Introduction

ideal sensor measurements (IMU, GNSS, airspeed, barometer), as well as the corresponding control commands, which can be corrupted by noise and used later in the estimator to assess the quality of the VDM-based navigation system [14]. Modifications to the model and filter optimization are first tested within this framework before being adapted to a C++ version.

The real-time aspect of the system (objective 3) requires the design of a complete soft-

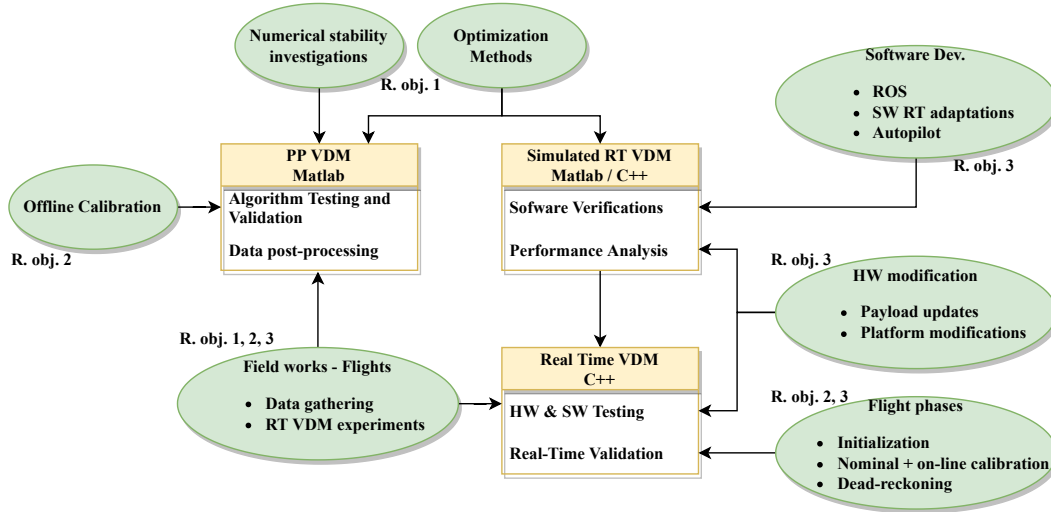


Figure 1.1: Methodology for the development of the real-time VDM-based navigation system, its calibration and optimizations

ware/hardware environment to acquire, label with the same global GNSS time reference, and distribute the necessary sensors and platform flight control commands to the processing software. To this end, Robot Operating System (ROS) [15] is chosen. The payload, the aircraft autopilot and different software must be fully compatible and adapted to the ROS structure. The software is tested in a *Simulated RT VDM* framework together with the MATLAB environment, as mentioned above.

When the feasibility of the real-time framework is verified, the *Real-Time VDM* framework can be investigated online through flight campaigns. The algorithms developed can be tested and validated in real-time at this stage. The utility of field experiments is two-fold. First, it permits the collection of the necessary flight data (sensors and autopilot data), which are used in postprocessing to i) calibrate the dynamic models used in the navigation software both in simulation and in real time, and ii) test the various modified algorithms with non-simulated data. Second, it gives the possibility to assess the performance of VDM-based navigation in real-time.

Transitions between flight campaigns, algorithm modifications, testing, and real-time adaptations were made iteratively with the advancement of the findings.

1.4 External Contributions

Realizing the research objectives requires numerous scientific and engineering aspects of various complexity. To achieve these goals, the candidate has benefited from the sporadic assistance of the *EPFL-TOPO* laboratory, which is greatly appreciated. Although the author is the main contributor to most of the material presented in this thesis, several tasks were accomplished through collaborative efforts and presented in joint publications. These topics will be clearly stated at the beginning of the corresponding part, including the following:

- Parts of the first phase (off-line) calibration procedure to estimate the initial aerodynamic coefficients from a given platform based on post-processed sensor data (Sec. 5.2)
- Hardware modifications of the platform, including sensor mounting and replacements (GNSS receiver and antenna, Pitot tube), autopilot configuration (Sec. 6.3) and the flight operation of the UAV (Sec. 6.4)
- The design and modifications of several versions of the payload, sensor mounting and wiring, and the design of the dedicated sensor boards (Sec. 6.2).

1.5 Thesis Outline

The thesis is presented in five parts, covering ten chapters in total, the content of which is summarized hereafter:

PART I: Preliminaries

Reviews the theoretical basis used in the thesis and recalls the VDM derivation for the two fixed-wing drones used in this thesis.

- **Chapter 2: Estimation Methods** presents different estimation techniques such as least squares and Kalman filtering with some of their extensions. This chapter aims to establish the semantics used in this document.
- **Chapter 3: The background of Integrated Navigation with Vehicle Dynamic Model** recalls the fundamentals of integrated navigation with VDM, the definition of frames, and the basics of geodesy.

PART II: VDM Navigation Enhancements

The second part of the thesis addresses the first two research objectives.

- **Chapter 4: Numerical aspects** develops means to improve the general state system estimation scheme, reduces potential numerical errors, and simplifies aerodynamic models based on highly correlated states.

Chapter 1. Introduction

- **Chapter 5: Calibration** proposes a multiphase calibration procedure. The first phase executed offline takes advantage of data collected during a high dynamic calibration flight to identify the platform-dependent aerodynamic coefficients characterizing the forces and moments, the so-called VDM parameters. The following online phases exploit the capabilities to calibrate and fine-tune some states and weather-dependent parameters using sensors present in the platform and, if not present, using a custom-made weather station.

PART III: Real-Time Implementation

The third part aims to materialize the autonomous navigation system based on VDM in real-time using a custom-made platform and covers the third objective of the investigation.

- **Chapter 6: Experimental Components** describes in detail the platforms used during this thesis, together with the payloads carried during the flight campaigns, as well as the characteristics of the devices and sensors that were integrated into each payload.
- **Chapter 7: Systems and Software Architecture** aims to provide a general understanding of the embedded real-time system with its subelements and constraints. It describes the numerous applications responsible for the correct data transfer and time-tagging of sensor and flight control commands and the C++ adaptation of the (MATLAB) VDM navigation estimator. The different flight phases of the VDM-based navigation system identified in this thesis are also presented.

PART IV: Results

The fourth part of the thesis gathers and presents the most important results and analysis of the different research objectives in three chapters.

- **Chapter 8: Coefficient determination** presents the two-phase calibration procedure with the navigation improvements achieved in research objective two.
- **Chapter 9: Real-time aspects** covers the numerical elements of the filter formulated in objective one and the details of the real-time implementation (objective three) and its requirements.
- **Chapter 10: Autonomous navigation** details the performance of VDM-based autonomous navigation using the improvements and methodologies proposed in this thesis.

PART V: Conclusion Remarks

The last part concludes the thesis by summarizing its main contributions and limitations and providing recommendations for future work.

Preliminaries **Part I**

2 Estimation Methods

Overview

Estimation principles are used in most parts of this research, and this chapter is dedicated to briefly reviewing relevant estimation methods, the adopted mathematical notations, and conventions. It should be noted that the chapter does not cover exhaustive derivations and comprehensive proofs but should, however, give enough material to understand the application of such tools in this thesis. Estimation theory aims to obtain values of specific parameters of a model that are a priori unknown and need to be inferred from measured/observed data. The model describes a functional relationship between these parameters and observed empirical data. In the framework of this thesis, the aerodynamic model of the fixed-wing drone is described by a set of coefficients, whose values are a priori unknown. Similarly, the drone trajectory is represented by a number of parameters that vary in time the value of which is conditioned by direct or indirect observations. In this context, the estimators covered in this chapter are (i) Least Squares (LS), where the fundamentals are introduced for the weighted and recursive versions; (ii) recursive estimators of quantities governed by non-linear differential equations in time (e.g., EKF); (iii) its post-processing variants, particularly the optimal recursive smoother. While filtering is suitable for real-time applications, optimal smoothing is particularly useful for calibration as it maximizes all the available information for the estimation of parameters over the whole duration of the experiment (e.g., a flight). Most derivations are adapted from [16–18].

2.1 Linear Least Squares

2.1.1 Ordinary and Weighted Least Squares

The most well-known linear estimation method is the LS. This method uses a redundancy in observations $\mathbf{z} \in \mathbb{R}^{l \times 1}$ with respect to the n number of parameters $\mathbf{x} \in \mathbb{R}^{n \times 1}$ to be identified,

$|z| = l > n = |\mathbf{x}|$. This defines the linear equation

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \boldsymbol{\epsilon} \quad (2.1)$$

where $\mathbf{H} \in \mathbb{R}^{l \times n}$ is a non random matrix called the *measurement design matrix* relating the parameters \mathbf{x} to the observations \mathbf{z} , and $\boldsymbol{\epsilon} \in \mathbb{R}^{l \times 1}$ is a noise vector of random normally distributed variables such that its realization has a mean of zero $E[\epsilon_i] = 0$, a constant variance $\text{var}[\epsilon_i] = \sigma^2 \forall i = [1, l]$, $\text{cov}[\epsilon_i, \epsilon_j] = 0 \forall i, j = [1, l]$ for $i \neq j$.

With a set of observations \mathbf{z} , the goal is to fit the data to the linear model specified by \mathbf{H} and obtain the best estimate $\tilde{\mathbf{x}}$ of the parameters \mathbf{x} by minimizing the sum of squared residuals of Eq. 2.1, $\mathbf{z} - \mathbf{H}\tilde{\mathbf{x}} = \mathbf{r}$. This leads to minimizing the objective function r

$$\text{argmin}_r = (r^T r) = (\mathbf{z} - \mathbf{H}\tilde{\mathbf{x}})^T (\mathbf{z} - \mathbf{H}\tilde{\mathbf{x}}) \quad (2.2)$$

However, as it is the case in general, the measurements \mathbf{z} have different uncertainties $\text{var}[\epsilon_i] = \sigma_i^2$ and can be correlated $\exists i, j$ with $i \neq j$ and $\text{cov}[\epsilon_i, \epsilon_j] \neq 0$. The Best Linear Unbiased Estimator (BLUE) is achieved by weighting the measurements by their inverse variance [19]. Defining $\mathbf{R}^{-1} = \frac{1}{\text{var}[\boldsymbol{\epsilon}]}$ as the matrix of the inverse of the observation covariance \mathbf{R} , the objective function to be minimized is

$$\text{argmin}_r = (r^T r) = (\mathbf{z} - \mathbf{H}\tilde{\mathbf{x}})^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{H}\tilde{\mathbf{x}}) \quad (2.3)$$

developing Eq. 2.3, taking the Jacobian [20] with respect to $\tilde{\mathbf{x}}$ and setting it to zero gives

$$\tilde{\mathbf{x}} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{z} \quad (2.4)$$

where

$$(\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} = \mathbf{P} \quad (2.5)$$

defines the covariance matrix of the parameters \mathbf{x} .

2.1.2 Recursive Least Square

When the number of observations becomes large $|z| = l \gg n = |\mathbf{x}|$ the design matrix \mathbf{H} grows proportionally and the computation of the estimated $\tilde{\mathbf{x}}$ becomes a heavy operation. However, if the observations \mathbf{z} are uncorrelated and initial parameters can be computed with a subset of observations, the estimation can be reformulated more efficiently. The Recursive Least Squares (RLS) defines recursive steps to add new observations \mathbf{z} to an existing estimate. The essential steps are summarized hereafter.

Assuming that at step k , where k observations are available, the estimate $\tilde{\mathbf{x}}$ with its covariance

\mathbf{P} are given by (from Eq. 2.5 and Eq. 2.4)

$$\tilde{\mathbf{x}}_k = \mathbf{P}_k \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k \quad (2.6)$$

$$\mathbf{P}_k = (\mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k)^{-1} \quad (2.7)$$

At time $k + 1$ a new uncorrelated observation z_{k+1} is available with correlation matrix R_{k+1} and design matrix H_{k+1} . Adapting the LS equation at step $k + 1$ gives

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \mathbf{R}_{k+1}^{-1} \mathbf{z}_{k+1} \quad (2.8)$$

$$\mathbf{P}_{k+1} = (\mathbf{H}_{k+1}^T \mathbf{R}_{k+1}^{-1} \mathbf{H}_{k+1})^{-1} \quad (2.9)$$

It should be noted that the system of equations Eq. 2.8 and Eq. 2.9 are correctly defined only when \mathbf{P} is full rank. In a recursive way, Eq. 2.8 and Eq. 2.9 can be reformulated with respect to the step k and new observations elements, z_{k+1} , R_{k+1} , and H_{k+1} , yielding

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \underbrace{\mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \mathbf{R}_{k+1}^{-1}}_{\text{gain}} \underbrace{(z_{k+1} - H_{k+1} \tilde{\mathbf{x}}_k)}_{\text{innovation}} \quad (2.10)$$

$$= \tilde{\mathbf{x}}_k + \mathbf{K}_{k+1} (z_{k+1} - H_{k+1} \tilde{\mathbf{x}}_k) \quad (2.11)$$

$$\mathbf{K}_{k+1} = \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1})^{-1} \quad (2.12)$$

$$(\mathbf{P}_{k+1})^{-1} = \mathbf{H}_{k+1}^T \mathbf{R}_{k+1}^{-1} \mathbf{H}_{k+1} = (\mathbf{P}_k)^{-1} + \mathbf{H}_{k+1}^T \mathbf{R}_{k+1}^{-1} \mathbf{H}_{k+1} \quad (2.13)$$

where matrix \mathbf{K}_{k+1} represents the weighted (gain) influence of the new observation z_{k+1} with respect to the previous estimate $\tilde{\mathbf{x}}_k$. $z_{k+1} - H_{k+1} \tilde{\mathbf{x}}_k$ tells how the estimate $\tilde{\mathbf{x}}_{k+1}$ needs to be updated with the gain. Expressing the covariance \mathbf{P}_{k+1} as a function of the gain \mathbf{K}_{k+1} , Eq. 2.13 can be expressed as

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1} \mathbf{H}_{k+1}) \mathbf{P}_k \quad (2.14)$$

with the Eq. 2.11, 2.12, and 2.14 the RLS equations are defined.

2.2 Kalman Filter

state space methods are useful for recursive estimations of parameters $\mathbf{x}(t)$ with time dependency.

2.2.1 State Space Dynamic Systems

A brief introduction on continuous and discrete-time state space systems are given in the sequels. The former defines the continuous elements which are discretized for the later system used in the software implementation both in MATLAB and C++. The derivations are simplified to assume zero input vector $\mathbf{u}(t)$, as normally defined in [21].

Continuous-Time Systems

The causal continuous-time state space system definition at time $t \in \mathbb{R}^+$ is expressed as

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) + \underbrace{\mathbf{L}(t)\mathbf{u}(t)}_{\text{omitted for now}} \quad (2.15)$$

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t) \quad (2.16)$$

where in the first equation, $\mathbf{x}(t)$ is the system state, $\mathbf{F}(t)$ is the *dynamic matrix* changing the state over time, $\mathbf{G}(t)$ is the *process noise shaping matrix*, $\mathbf{w}(t)$ is the process noise and $\mathbf{L}(t)$ is the *input coupling matrix*. The second equation is the same as Eq. 2.1, where $\mathbf{z}(t)$, $\mathbf{H}(t)$ and $\mathbf{v}(t)$ are the measurements, the *measurements design matrix* and the measurements' noise, respectively. The noises of the dynamic system are assumed to be Gaussian, independently and identically distributed (iid) with 0 mean and covariance $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ for the noise vector $\mathbf{w}(t)$ and $\mathbf{v}(t)$, respectively.

Discrete-Time Systems

The corresponding causal discrete-time state space system at step $k \in \mathbb{Z}^+$ is expressed as

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \mathbf{G}_k \mathbf{w}_k \quad (2.17)$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad (2.18)$$

with Φ_k , \mathbf{G}_k , \mathbf{H}_k being the discrete-time correspondences of $\mathbf{F}(t)$, $\mathbf{G}(t)$, and $\mathbf{H}(t)$ respectively. For now, the system is considered linear, therefore Φ_k , \mathbf{Q}_k , \mathbf{G}_k , and \mathbf{H}_k are matrices. Their derivations (approximations) are well known and not presented here (simplified derivations are given in Appendix B.2.1). The estimated states \mathbf{x}_k at time (step) t_k has covariance $P_k = E[\mathbf{x}_k \mathbf{x}_k^T]$. Therefore, the covariance at time t_{k+1} is given by

$$\mathbf{P}_{k+1} = E[\mathbf{x}_{k+1} \mathbf{x}_{k+1}^T] = E[(\Phi_k \mathbf{x}_k + \mathbf{G}_k \mathbf{w}_k)(\mathbf{x}_k^T \Phi_k^T + \mathbf{G}_k^T \mathbf{w}_k^T)] = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k \quad (2.19)$$

Kalman Filter Equations

The Kalman filter steps are regrouped in the Tab. 2.1 with the specific Eq. 2.17 and 2.19 for a discrete time linear system which *predicts* (\sim upper script) the states and the measurement *updates* (\wedge upper script) from the RLS derivation Eq. 2.12, 2.11 and 2.14. Note that the innovation

Table 2.1: Discrete Kalman Filter steps

Prediction steps	Update steps
$\tilde{\mathbf{x}}_{k+1} = \Phi_k \hat{\mathbf{x}}_k$	$\mathbf{S}_k = \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{W}_k$
$\hat{\mathbf{P}}_{k+1} = \Phi_k \hat{\mathbf{P}}_k \Phi_k^T + \mathbf{Q}_k$	$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1}$
	$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{z} - \mathbf{H}_k \tilde{\mathbf{x}}_k)$
	$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \hat{\mathbf{P}}_k$

equation $(\mathbf{z} - \mathbf{H}_k \tilde{\mathbf{x}}_k)$ was introduced in Eq. 2.11 and $(\mathbf{z} - \mathbf{H}_k \hat{\mathbf{x}}_k)$ is called the *residual*.

2.3 Extended Kalman Filter

The RLS and Kalman filter methods are estimation algorithms for linear systems. However, the system dynamics or the observation equations in navigation are generally non-linear and the EKF is a sub-optimal estimation method to handle such equations. By (re)introducing the deterministic input vector \mathbf{u}_k at time (step) t_k , the discrete non-linear system dynamic and observation equations are given by

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{G}_k(w)_k \quad (2.20)$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k \quad (2.21)$$

which give the EKF steps. The transition matrix $\Phi_k = \frac{\partial \mathbf{f}(\tilde{\mathbf{x}}_k, \mathbf{u}_k)}{\partial \mathbf{x}}$ is the Jacobian matrix formed with the partial derivative of \mathbf{f} , and $H_k = \frac{\partial \mathbf{h}(\tilde{\mathbf{x}}_k)}{\partial \mathbf{x}}$ is the Jacobian of the observation model function \mathbf{h} with respect to the system states \mathbf{x} . The computations of \mathbf{Q}_k is given in Appendix B.2.1.

2.3.1 Error State Extended Kalman Filter

The EKF is a nonlinear estimation method and is therefore sensitive to the approximation of $\tilde{\mathbf{x}}$, where otherwise the linearization of \mathbf{f} will be far from the actual states and filter divergence can occur [18]. Additionally, the accumulated errors in the states by numerous prediction steps, without updates to the true states, can become large. Interestingly, state errors appear to have less complex stochastic natures than the states themselves [22]. The error state equations are better modeled with linear approximations than the state equations, and thus the motivation to use the state errors over the states are manifold [23] and can be summarized as follows:

- They are close to the origin and thus avoid issues related to parameter singularities or gimbal lock, ensuring the linearity of the system.
- The error-states are small step quantities and safely approximated as first-order values. The state derivatives are therefore greatly simplified in terms of computational load and complexity.
- The state updates can be applied at a lower rate than the predictions because the signal dynamic in the error-states are smaller than in the states.

Before defining the steps for the Error State Extended Kalman Filter (ESEKF), the linearized equations of the Kalman Filter (KF) are derived as follows.

The reference (true) states $\mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}$ shall be defined as the addition of the estimated states $\hat{\mathbf{x}}$ and the error-states $\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}$. To simplify the notation, the time argument is abandoned and

the new system dynamic equation becomes

$$\dot{\mathbf{x}} = \hat{\dot{\mathbf{x}}} + \delta\dot{\mathbf{x}} = \mathbf{f}(\hat{\mathbf{x}} + \delta\mathbf{x}, \mathbf{u}) + \mathbf{w} \quad (2.22)$$

$$\approx \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u}) + \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \delta\mathbf{x} + \mathbf{w} \quad (2.23)$$

where Eq. 2.23 is the Taylor expansion with the first order term only. Note the difference between the derivative argument $\partial\mathbf{x}$ and the error-state $\delta\mathbf{x}$. The linearized dynamic equation for the error-state is therefore

$$\delta\dot{\mathbf{x}} = \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \delta\mathbf{x} + \mathbf{w} \quad (2.24)$$

$$= \mathbf{F}(\hat{\mathbf{x}}, \mathbf{u})\delta\mathbf{x} + \mathbf{w} \quad (2.25)$$

The measurement Eq. 2.21 is linearized similarly to Eq. 2.22. By dropping the subscript k specifying the step, it gives

$$\mathbf{z} = \mathbf{h}(\hat{\mathbf{x}} + \delta\mathbf{x}) + \mathbf{r}_k \quad (2.26)$$

$$= \mathbf{h}(\hat{\mathbf{x}}) + \left. \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \delta\mathbf{x} + \mathbf{r} \quad (2.27)$$

Note that in the Kalman filter step, the measurement updates are done with respect to the estimated $\hat{\mathbf{x}}$ and not the updated $\hat{\mathbf{x}}$ states. The upper script is kept for notation consistency.

The residual $\delta\mathbf{z}$ is the subtraction of Eq. 2.27 with Eq. 2.21 and yields

$$\delta\mathbf{z} = \left. \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \delta\mathbf{x} + \mathbf{r} \quad (2.28)$$

$$= \mathbf{H}(\hat{\mathbf{x}})\delta\mathbf{x} + \mathbf{r} \quad (2.29)$$

The linearized error Eq. 2.25 and residual Eq. 2.29 form a new linear system

$$\delta\dot{\mathbf{x}} = \mathbf{F}(\hat{\mathbf{x}}, \mathbf{u})\delta\mathbf{x} + \mathbf{w} \quad (2.30)$$

$$\delta\mathbf{z}_k = \mathbf{H}(\hat{\mathbf{x}})\delta\mathbf{x} + \mathbf{r}_k \quad (2.31)$$

and can be used in an optimal non-linear estimator such as the EKF, forming the ESEKF. For the discrete-time equations, the matrix $\bar{\Phi}_k$ and \mathbf{Q}_k , as well as the covariance propagation, are derived in a similar way as for the linear KF presented in Sec. 2.2.1. The filter estimates and updates the error-states and not the states themselves. In the EKF, the prediction of the states $\tilde{\mathbf{x}}_k$ is performed via the nonlinear function $\mathbf{f}()$ and not with the transition matrix $\bar{\Phi}$ as presented in Eq. 2.20. These states are considered as the reference error-less states. Therefore, in the ESEKF, the prediction step of the error-states $\delta\tilde{\mathbf{x}}_{k+1}$ are based on the current residual error $\delta\tilde{\mathbf{x}}_k$ which equal zero. The prediction of the error-states is therefore always null and this operation is omitted in the ESEKF. In addition, the error-states updated $\hat{\mathbf{x}}_{k+1}$ depends on the correction of the predicted null-error-states $\delta\tilde{\mathbf{x}}_k$. The equation is thus simplified to $\delta\hat{\mathbf{x}}_k = \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\tilde{\mathbf{x}}_k)) = \mathbf{K}_k\delta\tilde{\mathbf{z}}_k$.

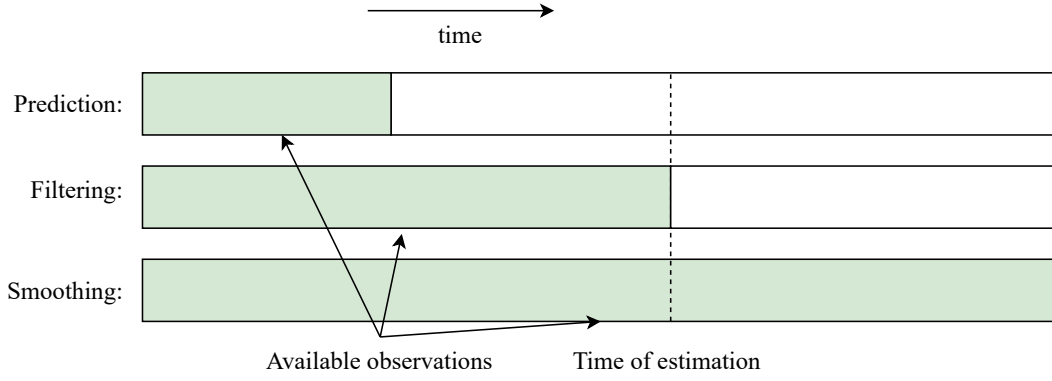


Figure 2.1: Different estimation phases depending on the availability of observations

The filtering steps are summarized in Tab. 2.2.

Table 2.2: Discrete Error-State Extended Kalman Filter steps

Prediction steps	Update steps
$\tilde{\mathbf{x}}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k, \delta t)$	$\mathbf{S}_k = \mathbf{H}_k \tilde{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{W}_k$
$\delta \tilde{\mathbf{x}}_{k+1} = \Phi_k \delta \hat{\mathbf{x}}_k = 0$	$\mathbf{K}_k = \tilde{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1}$
$\tilde{\mathbf{P}}_{k+1} = \Phi_k \tilde{\mathbf{P}}_k \Phi_k^T + \mathbf{Q}_k$	$\delta \hat{\mathbf{x}}_k = \delta \tilde{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{z}_k - \mathbf{h}(\tilde{\mathbf{x}}_k)) = \mathbf{K}_k \delta \tilde{\mathbf{z}}_k$
	$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \delta \hat{\mathbf{x}}_k$
	$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \tilde{\mathbf{P}}_k$

2.4 Optimal Smoothing

Different estimation phases depend on when the observations are available and when the state estimation needs to be performed. The prediction foresees the states in a future time, based on the current states and dynamic models. When few observations are available at the estimation time, the estimate is filtered (updated). When the observations are saved from being treated in a post-processing manner, the estimation of the states at a particular time of interest can benefit from the knowledge of future observations, in a technique known as smoothing. The three phases can be seen in Fig. 2.1 at the time t of estimation interest.

While the prediction and filtering steps have been described above, the optimal smoothing algorithm is presented hereafter. Some optimal smoothing algorithms are described in [24, 25], and these sources inspire the following summary. Smoothing principles are based on the weighted combinations of several (at least 2) forward and backward filter phases. The *fixed-interval smoother* is composed of a forward filtering phase, as described earlier in this section, with a backward phase briefly summarized here. The states estimate and corresponding covariance matrix in the forward filter phase are denoted with \mathbf{x}_f and \mathbf{P}_f , respectively. The backward states estimate \mathbf{x}_b and covariance \mathbf{P}_b are computed by reversing the sequence of predictions and updated events, and the equations are summarized in Tab. 2.3 where the

Table 2.3: Backward Discrete Error-State Extended Kalman Filter steps

Prediction steps	Update steps
$\tilde{\mathbf{x}}_{\mathbf{b}k-1} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k, \delta t)$	$\mathbf{S}_k = \mathbf{H}_k \tilde{\mathbf{P}}_{\mathbf{b}k} \mathbf{H}_k^T + \mathbf{W}_k$
$\tilde{\mathbf{P}}_{k-1} = \Phi_k \hat{\mathbf{P}}_k \Phi_k^T + \mathbf{Q}_k$	$\mathbf{K}_k = \tilde{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1}$
	$\delta \hat{\mathbf{x}}_{\mathbf{b}k} = \mathbf{K}_k \delta \mathbf{z}_k$
	$\hat{\mathbf{x}}_{\mathbf{b}k-1} = \hat{\mathbf{x}}_{\mathbf{b}k} + \delta \hat{\mathbf{x}}_{\mathbf{b}k}$
	$\hat{\mathbf{P}}_{\mathbf{b}k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \tilde{\mathbf{P}}_{\mathbf{b}k}$

integration time δt is now negative and Φ is computed as in Eq. B.7 with the negative dynamic matrix $-\mathbf{F}$ reversing the propagation.

The smoothed estimate \mathbf{x}_s and covariance \mathbf{P}_s are the combination of the two filtered sequences (forward and backward) and are given by the following equation for each step k , with the subscript k removed for readability

$$\mathbf{x}_s = \mathbf{x}_f + \mathbf{P}_f \mathbf{P}_b^{-1} (\mathbf{x}_b - \mathbf{x}_f) \quad (2.32)$$

$$\mathbf{P}_s = \left(\mathbf{P}_f^{-1} + \mathbf{P}_b^{-1} \right)^{-1} \quad (2.33)$$

Other optimal smoothing techniques exist, including the Modified Bryson-Frazier (MBF) smoother [26] that uses two passes of filtering, where the main advantage is the non-requirement of a covariance matrix inversion, and the Rauch-Tung-Striebel (RTS) [27] that requires only one filtering sequence (forward or backward). The smoothing methodology is employed in one proposed method to calibrate the aerodynamic coefficients (Sec. 5.3.1).

Summary

This chapter has covered an overview of the estimation methods that are used in this research. It has summarized least squares and Kalman filter. These methods shall be later used for VDM-based navigation and calibration. Subsequently, optimal smoothing was reviewed as an enhancement for state estimation in a VDM-based framework. However, for understanding model-based navigation, the knowledge of general geodetic principles and inertial navigation aspects is of primary importance. These aspects will be reviewed in the next chapter.

3 Background of Integrated Navigation with Vehicle Dynamic Model

Overview

This chapter introduces the fundamentals covering general inertial navigation and geodesy as needed for the aspects of VDM-based navigation. These theoretical concepts are well established from which several conventions and definitions are recalled as used throughout the manuscript. The basics of rotational operations are introduced for the Euler angles and quaternion representation. The benefits and limitations of the two representations are discussed to understand why both are employed in the attitude of the drone. Then the frames employed in this thesis are defined with the corresponding transformations among them. These are: inertial, Earth, local-level, and body, as well as some sensor frames. Afterward, the rigid body dynamics are expressed with respect to these frames via differential equations for the position, linear and angular velocity, and attitude, respectively. The equation parameters are employed as the navigation states in the estimation scheme presented afterward. Contrary to inertial-based navigation, model-based navigation obtains specific forces and angular velocities from the aerodynamic model of a drone rather than from measurements. Separate aerodynamic models are presented for both platforms used in this thesis: a fixed-wing and a delta-wing UAV, parameters of which are to be determined from in-flight data. The observation models of the different sensors used in this context are described, including an IMU, GNSS receiver, airspeed, and a barometer. All models are linearized with respect to all parameters of interest using automatic tools to generate the mandatory Jacobian matrices for the estimators presented in the previous chapter. The manuscripts and internal teaching materials inspiring the presented mathematical relations and illustrations are [11, 16, 17, 28, 29].

3.1 Rotation Operations

3.1.1 Rotation Matrix

A vector \mathbf{x}^a represented in any Cartesian frame a can be expressed with respect to any other Cartesian frame b with a specific rotation matrix denoted as \mathbf{R}_a^b to obtain the same vector expressed in the second frame as

$$\mathbf{x}^b = \mathbf{R}_a^b \mathbf{x}^a \quad (3.1)$$

where the subscript indicates the initial and the superscript indicates the final frame. The matrix \mathbf{R}_a^b is called a Direct Cosine Matrix (DCM).

Euler Angles

The DCM can be divided into three rotations around each axis of the initial frame a where the rotation quantities are given by Euler angles [30]. The order around which axis the sequence of rotations is performed has to be defined. For the a frame with its axis x_1^a , x_2^a , and x_3^a , a sequence of rotation can be given by first rotating around the x_1^a , then x_2^a and finally x_3^a by the angles, α , β , and γ , respectively. The complete rotation is therefore given by

$$\mathbf{R}_a^b = \mathbf{R}_3(\gamma)\mathbf{R}_2(\beta)\mathbf{R}_1(\alpha) \quad (3.2)$$

$$= \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad (3.3)$$

The creation of the DCM from Euler angles will be referred to by the operation $f_{dcm}(\alpha, \beta, \gamma)$. It should be noted that the DCM is only dependent on the relative rotations between two frames.

3.1.2 Differential Rotations

An attitude representation using Euler angles follows the Special Orthogonal Group ($SO(3)$) manifold, meaning that two attitude vectors cannot be added element-wise together as in a Euclidean space. In other words, an initial orientation \mathbf{x}^a to which the quantity δx^a is added is not equal to

$$x_{final}^a \neq x^a + \delta x^a \quad (3.4)$$

with

$$\delta x^a = \begin{bmatrix} \delta \alpha \\ \delta \beta \\ \delta \gamma \end{bmatrix} \quad (3.5)$$

If δx^a is composed of small angles, this approximation can be used for software implementation.

The vector x^a can be transformed into a corresponding DCM \mathbf{R}^a , following a sequence of rotation as presented in Sec. 3.1.1. Similarly, the quantity δx^a can be represented in another DCM matrix. Then the two matrices can be multiplied to obtain the final orientation \mathbf{R}_{final}^a .

$$\mathbf{R}_{final}^a = \mathbf{R}^a \cdot f_{dcm}(\delta x^a) \quad (3.6)$$

The Euler angles of $\mathbf{x}_{final}^a = f_{dcm}^{-1}(\mathbf{R}_{final}^a)$ can be recovered from the matrix \mathbf{R}_{final}^a with the inverse transformation $f_{dcm}^{-1}()$ given without proof [31] by

$$\alpha = \text{atan}\left(\frac{R_{21}}{R_{11}}\right) \quad (3.7)$$

$$\beta = -\text{asin}(R_{31}) \quad (3.8)$$

$$\gamma = \text{atan}\left(\frac{R_{32}}{R_{33}}\right) \quad (3.9)$$

where $R_{i,j}, i, j \in [1, 2, 3]$ are the column and row indices, respectively, of \mathbf{R}_{final}^a .

Skew Matrix

For small rotation perturbations, the DCM differs from the identity matrix by only small quantities when the approximation $\sin(x) \approx x$ and $\cos(x) \approx 1$ is used. The rotation matrices R_1, R_2 and R_3 can be approximated as

$$R_1 = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha \\ 0 & -\alpha & 1 \end{bmatrix}}_{I_{3 \times 3} - \alpha}, R_2 = \underbrace{\begin{bmatrix} 1 & 0 & -\beta \\ 0 & 1 & 0 \\ \beta & 0 & 1 \end{bmatrix}}_{I_{3 \times 3} - \beta}, R_3 = \underbrace{\begin{bmatrix} 1 & \gamma & 0 \\ -\gamma & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{I_{3 \times 3} - \gamma} \quad (3.10)$$

and the complete rotation matrix $R = R_1 \cdot R_2 \cdot R_3$ can be rewritten as

$$R = (I_{3 \times 3} - \alpha)(I_{3 \times 3} - \beta)(I_{3 \times 3} - \gamma) = I_{3 \times 3} - \alpha - \beta - \gamma + \underbrace{\gamma\alpha + \gamma\beta + \alpha\beta - \gamma\alpha\beta}_{\text{values} \approx 0} \quad (3.11)$$

$$= I_{3 \times 3} - (\gamma, \alpha, \beta) = I_{3 \times 3} - \Psi \quad (3.12)$$

where

$$\Psi(\gamma, \alpha, \beta) = \begin{bmatrix} 0 & \gamma & -\beta \\ -\gamma & 0 & \alpha \\ \beta & -\alpha & 0 \end{bmatrix} \triangleq [\Psi]_{\times} \quad (3.13)$$

Chapter 3. Background of Integrated Navigation with Vehicle Dynamic Model

is the skew-symmetric matrix called an axiator [32] and $[\Psi]_x$ is the reduced representation. The skew matrix $[\Psi]_x$ is useful when the time derivative of a rotation matrix R_a^b has to be computed to describe the angular velocity or the rate of change of the attitude. Moreover, a time dependant rotation matrix $R_a^b(t)$ can be decomposed into an initial rotation matrix $R_a^b(t_0)$ with $t_0 \rightarrow t$ to which the infinitesimal angles are added $I_{3 \times 3} - \Psi(\delta t)$ where $\delta t = t - t_0$ and $\delta t \rightarrow 0$

$$R_a^b(t_0 + \delta t) = (I_{3 \times 3} - \Psi) R_a^b(t_0) \quad (3.14)$$

With the definition to the derivative, the time-dependant rotation matrix R_a^b gives

$$\dot{R}_a^b = \lim_{\delta t \rightarrow 0} \frac{R_a^b(t_0 + \delta t) - R_a^b(t_0)}{\delta t} \quad (3.15)$$

Using Eq. 3.14, the term $R_a^b(t_0 + \delta t)$ can be replaced by $(I_{3 \times 3} - \Psi) R_a^b(t_0)$.

Substituting, the term $R_a^b(t_0)$ cancels out once. The time derivative can be rewritten as

$$\dot{R}_a^b = \lim_{\delta t \rightarrow 0} \frac{-\Psi}{\delta t} R_a^b(t_0) = -\Omega_{ab}^b R_a^b(t_0) \quad (3.16)$$

where term $\frac{\Psi}{\delta t}$ can be written as Ω_{ab}^b and correspond to the skew-symmetric matrix defined in Eq. 3.12. The elements of the matrix Ω_{ab}^b can be interpreted as the angular-rate in the a frame with respect to the b frame (the two subscripts) and expressed in the b frame (superscript). The derivation holds for any two arbitrary frames. The Eq. 3.16 can be further developed to give $\dot{R}_a^b = R_a^b(t_0) \Omega_{ba}^a$ where the rotation sense and the frame in which the rotation is expressed have been inverted.

The skew-matrix will be used to derive the differential equations for the attitude in Sec. 3.4. The skew-matrix does not involve trigonometric operations and can substitute the DCM matrix for computational requirements but small angles should be assumed.

3.1.3 Quaternion

Euler angles are easy to interpret and widely used. However, when used to represent an attitude in a Cartesian coordinate frame, they possess an intrinsic ambiguity when the angle around the second axis approaches $\pi/2$. This situation is known as gimbal lock [33] and sets the attitude in an unstable state. A solution, instead of using the Euler angle, is to use another attitude representation: the quaternion [34]. The advantages of the quaternion over the Euler angles are manifold. One advantage is that it removes the gimbal lock situation such that the quaternion operations are linear. However, the quaternions have setbacks as they are not unique and are over-parameterized by their definitions. A combination of the two attitude representations can be used for navigation (Sec. 3.2).

A description of the quaternion fundamentals are presented here. A detailed and thorough

definition of the quaternion can be found in [35], pleasing introductions are given by [36, 37] and a graceful presentation of the quaternions in photogrammetry and navigation can be found in [38]. The following summary is inspired from these references.

A quaternion is composed of two complex numbers, $C_1 = x + yi$ and $C_2 = u + vi$, where $\{x, y, u, v\} \in \mathbb{R}$ and $\{i, j\} \in \mathbb{C}$ to obtain as $\mathbf{q} = C_1 + C_2j$. Developing the quaternion \mathbf{q} gives

$$\mathbf{q} = x + yi + uj + vk \text{ with } k \triangleq ij \quad (3.17)$$

where $\mathbf{q} \in \mathbb{H}$ is the quaternion space.

The following relationship can be made

$$ij = k = -ji, jk = i = -kj, ki = j = -ik \quad (3.18)$$

where the order of the product operation matters (i.e., is not commutative).

Another representation of a quaternion omits the combination of real $\{q_0, q_1, q_2, q_3\}$ and imaginary $\{1, i, j, k\}$ components as in Eq. 3.17, to form the sum of a real scalar and an imaginary vector

$$\mathbf{q} = q_0 + q_1i + q_2j + q_3k \iff \mathbf{q} = q_0 + \mathbf{q} \quad (3.19)$$

where q_0 is the real part and $\{q_1, q_2, q_3\}$ is the imaginary vector.

Conventions

The definition with q_0 as the real scalar follows the Hamilton [39] convention and is used in the rest of this manuscript, as well as in the software implementations.¹

3.1.4 Main Quaternion Properties

The main quaternion properties and mathematical operations are presented hereafter.

¹The JPL [40] convention has the real scalar at the fourth position ($\{q_0, q_1, q_2, \mathbf{q}_3\}$) and the quaternion operations have to be redefined accordingly.

Sum

The sum follows the vectorial element-wise operation. Let $\mathbf{p} = [p_0, p_1, p_2, p_3] \in \mathbb{H}$ and $\mathbf{q} = [q_0, q_1, q_2, q_3] \in \mathbb{H}$ two quaternions, the summation is defined as

$$\mathbf{p} \pm \mathbf{q} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \pm \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} p_0 \pm q_0 \\ p_1 \pm q_1 \\ p_2 \pm q_2 \\ p_3 \pm q_3 \end{bmatrix} \quad (3.20)$$

By linearity of the sum operation, the rule of commutativity and associativity are valid.

Product

Using the definition in Eq. 3.18, the simplified quaternion representation in Eq.3.19 and introducing the quaternion product \otimes , the operation is defined as

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \otimes \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{bmatrix} \quad (3.21)$$

The later equation with the proprieties in Eq. 3.18 expose that the commutativity does not apply in the quaterion production. Therefore, $\mathbf{p} \otimes \mathbf{q} \neq \mathbf{q} \otimes \mathbf{p}$ in general.

Inverse, Norm and Unit Quaternion

A quaternion \mathbf{q} times its inverse \mathbf{q}^{-1} equals the identity, therefore $\mathbf{q} \otimes \mathbf{q}^{-1} = \mathbb{1}$.

The quaternion norm is given by

$$\|\mathbf{q}\| = \sqrt{\mathbf{q} \otimes \mathbf{q}^*} = \sqrt{\mathbf{q}^* \otimes \mathbf{q}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \in \mathbb{R} \quad (3.22)$$

where \mathbf{q}^* is the quaternion conjugate and the sign of the complex elements in \mathbf{q} , including q_1, q_2, q_3 , are inverted. With the definition in Eq. 3.22, a unit quaterion is when $\|\mathbf{q}\| = 1$ and by association, its conjugate equals the inverse, $\mathbf{q}^* = \mathbf{q}^{-1}$

Exponential of pure quaternions

A pure quaternion \mathbf{q} possesses zero as the real value q_0 . Rewriting $\mathbf{q} = \mathbf{u}\theta$ where $\theta = \|\mathbf{q}\| \in \mathbb{R}$ and \mathbf{u} is a unit of pure quaternion, the exponential of a pure quaternion is given without proof

by

$$e^{\mathbf{q}} = \begin{bmatrix} \cos(\theta) \\ \mathbf{u} \sin(\theta) \end{bmatrix} \quad (3.23)$$

Note that when $\|e^{\mathbf{q}}\|^2 = \cos(\theta)^2 + \sin(\theta)^2 = 1$, the exponential of a pure quaternion is one.

3.1.5 Differential Quaternion

With the unit quaternion \mathbf{q} , the identity and conjugate proprieties give $q^* \otimes q = 1$. If during a period of time dt the rotation rate is assumed to be constant, both sides can be differentiated to obtain

$$0 = \frac{d(\mathbf{q}^* \otimes \mathbf{q})}{dt} = \dot{\mathbf{q}}^* \otimes \mathbf{q} + \mathbf{q}^* \otimes \dot{\mathbf{q}} \implies \mathbf{q}^* \otimes \dot{\mathbf{q}} = -(\dot{\mathbf{q}}^* \otimes \mathbf{q}) = -(\mathbf{q}^* \otimes \dot{\mathbf{q}}) \quad (3.24)$$

The right side of the equation, $\mathbf{q}^* \otimes \dot{\mathbf{q}}$, is pure quaternion. Thus, the pure quaternion can be rewritten with $\Omega \in \mathbb{H}$ as

$$\mathbf{q}^* \otimes \dot{\mathbf{q}} = \begin{bmatrix} 0 \\ \Omega \end{bmatrix} \triangleq \Omega \in \mathbb{H} \quad (3.25)$$

Left-multiplying both side of Eq. 3.24 by \mathbf{q} gives

$$\dot{\mathbf{q}} = \mathbf{q} \otimes \Omega \quad (3.26)$$

The space \mathbb{H} of pure quaternions constitutes the tangent space (Lie Group [41]) of the unit three dimensional sphere in $SO(3)$ of quaternions in \mathbb{R}^4 . This space covers only half of the angular velocity quantity. Therefore, a rotation quantity $\mathbf{v} \in \mathbb{R}$ is mapped to half the rotation in $SO(3)$. Thus, the angular velocity can be rewritten as $\Omega = \frac{\omega}{2}$ in \mathbb{R} . Then Eq 3.26 becomes

$$\dot{\mathbf{q}} = \frac{1}{2} \mathbf{q} \otimes [\omega]_q \quad (3.27)$$

which forms the Ordinary Differential Equation (ODE) and will be used later on in Sec. 3.4 as attitude equation in the navigation states. The symbol $[\omega]_q$ uses the definition in Eq. 3.25 and is defined as

$$[\omega]_q = \begin{bmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (3.28)$$

3.1.6 Rotation Uncertainty

The operations to convert Euler angles to quaternion and the reverse is given in Appendix. B.3.2. The uncertainty in attitude can be converted from one representation to the other using the covariance law and is repeated hereafter [42].

Covariance: Quaternion to Euler

The covariance from Quaternion to Euler angle is given by

$$\mathbf{P}_{euler} = \mathbf{J}_{3 \times 4} \mathbf{P}_{quat} \mathbf{J}_{4 \times 3} \quad (3.29)$$

where \mathbf{J} is the Jacobian given from the partial derivatives of the Euler angle equations with respect to the quaternion given in Eq B.22, \mathbf{P}_{quat} is the covariance matrix for the quaternions and \mathbf{P}_{Euler} is the equivalent covariance matrix for the Euler angle. The derivation details can be found in [43].

Covariance: Euler to Quaternion

The covariance from Euler to quaternion is given by

$$\mathbf{P}_{quat} = \mathbf{J}_{4 \times 3}^T \mathbf{P}_{Euler} \mathbf{J}_{3 \times 4} \quad (3.30)$$

and is derived in a similar manner than Eq 3.29 and the Jacobian $\mathbf{J}_{4 \times 3}$ are obtained by the partial derivation of Eq B.21 with respect to the Euler angle.

3.2 Quaternion-Euler Error State Extended Kalman Filter

As presented in Sec. 3.1.3, the attitude representation using quaternion has several advantages with respect to the traditional Euler angles. However, the quaternion beauty comes with a prize. By removing some of the Euler angle limitations, the addition of a fourth parameter to represent an attitude, as seen in Eq. 3.17, creates a singularity in the covariance matrix \mathbf{P}_k that is no longer full rank. The covariance matrix \mathbf{P}_k is used when the Kalman gain \mathbf{K}_k is computed and can lead to numerical errors from the inverse determinant of \mathbf{S}_k . This occurrence sets the Kalman Filter in an unstable state which eventually leads to slow convergence or even to the divergence of the state estimation [44].

A solution to avoid numerical instability when \mathbf{K}_k is computed is to use two different attitude representations within the Kalman Filter: the use of the quaternion kinematics for the nominal attitude states \mathbf{q}_b^l , while keeping the Euler representation for the attitude error state $\delta\theta_b^l$ and uncertainty \mathbf{P} . The KF steps have to be adapted accordingly. A detailed and thorough formulation of the continuous and discrete error state Kalman filter employing the Euler-quaternion attitude combination for a INS-derived navigation system can be found in [37]. The

following derivations are inspired from the aforementioned work and adapted to generalize the formulation for any type of system states.

Prediction

The non-linear system dynamic function \mathbf{f} used to compute the states prediction $\tilde{\mathbf{x}}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k, \delta t)$ is formulated in quaternion following the differential equation given in Eq. 3.54 and does not need to be adapted. The discrete-time dynamic matrix Φ_k derived from \mathbf{f} , has to be adapted to be compatible with the dimension of the covariance matrix \mathbf{P} in which attitude uncertainties are represented with Euler angles. It is obtained by taking the Jacobians of the function \mathbf{f} with respect to the error states $\delta \mathbf{x}$ as presented in Eq. 2.25

The time propagation of the state covariance matrix \mathbf{P}_k is given in Eq. 2.19. The process noise matrix \mathbf{Q}_k contains the system dynamic noise of the states. The uncertainties for attitude states are defined in quaternion and have to be propagated with Euler angles to have the same units as the states covariance matrix \mathbf{P}_k . The covariance propagation from/to quaternion with respect to Euler angles are described in Eq. 3.29 and 3.30 with only the attitude states. The difference now is the Jacobians are taken by partial derivatives with respect to the whole augmented states \mathbf{x} . Note that \mathbf{Q}_k is in general a block diagonal matrix and the transformation can be seen as a simple change of units.

Update

The sensor observations serve to correct the states in the update steps of the filter. The sensor observations models $h()$ are in general a nonlinear function of the system states \mathbf{x} using quaternion parametrization. The matrix \mathbf{H} is the Jacobians of $h()$ with respect to the error states $\delta \mathbf{x}$, evaluated at \mathbf{x} and yields

$$\mathbf{H}_k = \left. \frac{\partial h}{\partial \delta \mathbf{x}} \right|_{\mathbf{x}} = \left. \frac{\partial h}{\partial \mathbf{x}_k} \right|_{\mathbf{x}} \left. \frac{\partial \mathbf{x}_k}{\partial \delta \mathbf{x}} \right|_{\mathbf{x}} \quad (3.31)$$

where the derivative chain rule is applied to obtain $\left. \frac{\partial h}{\partial \mathbf{x}_k} \right|_{\mathbf{x}}$, the partial derivative of $h()$ with respect of the true states \mathbf{x} , and $\left. \frac{\partial \mathbf{x}_k}{\partial \delta \mathbf{x}} \right|_{\mathbf{x}}$ is the partial derivative of the true states \mathbf{x} with respect to the error states $\delta \mathbf{x}$. Note that the matrix \mathbf{H}_k is of size $l \times n - 1$ where n is the number of true states in \mathbf{x} and l is the number of measurements in one observation (6 for the IMU, 3 for the GNSS position or velocity, 1 for the airspeed and barometric updates for example). Similarly, the gain matrix \mathbf{K}_k is of size $n - 1 \times l$.

The mixed quaternion-Euler ESEKF has the advantage that it avoids gimbal lock and guarantees the non-singularity of the related covariance matrix \mathbf{P}_k .

3.3 Frames Definitions

The motion of objects is often materialized as points with three axes in a 3D space, which is related to coordinates characterizing its position, velocity, and orientation. A reference or coordinate frame is the materialization of a reference or coordinate system with rules to be respected, e.g. orthogonality and direction of positive rotations. A number of frames are used and their definitions are given in this section.

3.3.1 Inertial Frame

The inertial frame (*i*-frame) is a non-accelerated frame being either at rest or subject to uniform translational motions. Newton's second law $\mathbf{F} = m \cdot \mathbf{a}$ can be written in an inertial frame as

$$\mathbf{F}^i = m \cdot \ddot{\mathbf{x}}^i \quad (3.32)$$

where \mathbf{F}^i represents the total forces applied to the mass m and $\ddot{\mathbf{x}}^i$ the accelerations experienced. Moreover, in the presence of a gravitational field and Einstein's principle of equivalence, Eq. 3.32 is modified as

$$\mathbf{F}^i + \mathbf{G}^i = m \cdot \ddot{\mathbf{x}}^i \quad (3.33)$$

Dividing both sides of Eq. 3.33 by m ^{II} gives

$$\mathbf{f}^i + \mathbf{g}^i = \ddot{\mathbf{x}}^i \Rightarrow \ddot{\mathbf{x}}^i = \mathbf{f}^i - \mathbf{g}^i \quad (3.34)$$

here \mathbf{f}^i is the specific forces. The approximation of an inertial frame for navigation is depicted in Fig. 3.1 in black. The origin of the frame is located at the center of the Earth with the first axis

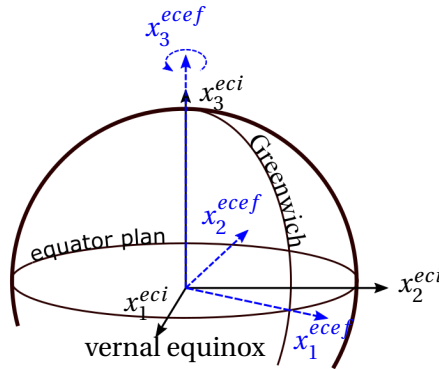


Figure 3.1: Inertial ECI frame in black, rotating ECEF frame in dashed-blue

x_1^i pointing towards the Vernal equinox, the third axis x_3^i in direction to the Earth rotation axis, and the second axis x_2^i parallel to the equatorial plane to complete a right-handed Cartesian

^{II}mass is the same whether inertial or gravitational [45]

coordinate frame. To consider this frame as inertial, the Earth is assumed not to be subject to any acceleration apart from its rotation. The frame is called Earth Center Inertial (ECI). In this approximation, accelerations due to ecliptic motion as well as solar system in the Milky Way galaxy are ignored.

3.3.2 The Earth Frame

The terrestrial equatorial frame (*e*-frame) is a terrestrial frame defined with its origin as the geocenter of the Earth and the third axis x_e^3 pointing toward the pole along the mean rotation axis of the Earth similarly to the ECI frame. However, the first axis x_1^e points towards the Greenwich meridian and therefore rotates together with the Earth, and the second axis x_2^e completes a right-handed Cartesian frame called Earth Center Earth Fixed (ECEF) frame. For practical reasons, as well as for the definition of the local-level frame, it is useful to associate an ellipsoid of (revolution) reference to this frame with latitude ϕ , longitude λ and height h . The conversion from the two representations is given by

$$\begin{bmatrix} x_1^e \\ x_2^e \\ x_3^e \end{bmatrix} = \begin{bmatrix} (R_p + h) \cos(\phi) \cos(\lambda) \\ (R_p + h) \cos(\phi) \sin(\lambda) \\ [R_p (1 - e^2)] \sin(\phi) \end{bmatrix} \quad (3.35)$$

where R_p is the radius of curvature in prime vertical [46] that corresponds to the normal distance from the ellipsoid to the polar axis x_e^3 in an East-West direction. It is computed as

$$R_p = \frac{a}{(1 - e^2 \sin^2(\phi))^{1/2}} \quad (3.36)$$

where a is the semi-major axis of the ellipsoid and e is its eccentricity. The radius of curvature in the meridian plane [46] R_m is given as

$$R_m = \frac{a(1 - e^2)}{(1 - e^2 \sin^2(\phi))^{3/2}} = R_p \frac{1 - e^2}{1 - e^2 \sin^2(\phi)} \quad (3.37)$$

which corresponds to the Earth's radius in a North-South direction and is used in the differential equation for the position states (Sec. 3.4).

Rotation from Inertial to Earth Frame

To transform the coordinates from the inertial i to the Earth frame e , a rotation around the common third axis $x_3^i = x_3^e$ of an angle ω needs to be performed. The ω is the angle between the rotating first axis x_1^e (Greenwich) and the fixed Vernal equinox direction x_1^i . To keep to the right-handed Cartesian convention, the positive rotation is counter-clockwise. The rotation is also symbolized by R_3 (Sec. 3.3) as it is performed around the third axis of the reference frame.

The transformation equations are therefore

$$\mathbf{x}^e = R_i^e(\omega) \cdot \mathbf{x}^i \quad (3.38)$$

with

$$R_i^e(\omega) = R_3(\omega) = \begin{bmatrix} \cos(\omega) & \sin(\omega) & 0 \\ -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.39)$$

3.3.3 Local Level Frame

The local-(mapping) frame (*l*-frame) is a Cartesian frame, tangential to the ellipsoid (e.g. corresponding to ellipsoid defined as part of World Geodetic System 84 (WGS84)) at an arbitrary origin close to a working area. In the so called local North-East-Down (NED) frame, which is used in navigation, the first axis x_1^l points towards the geographic North, the second axis x_2^l towards the East and the third axis x_3^l points down in the direction of surface normal. In the East-North-Up (ENU) definition, the first two axes are swapped while the direction of the third is reversed. A representation of the ENU frame is depicted in Fig. 3.2 with an ellipsoidal elevation of h and in Fig. 3.4 where the NED frame is used as a local-mapping frame.

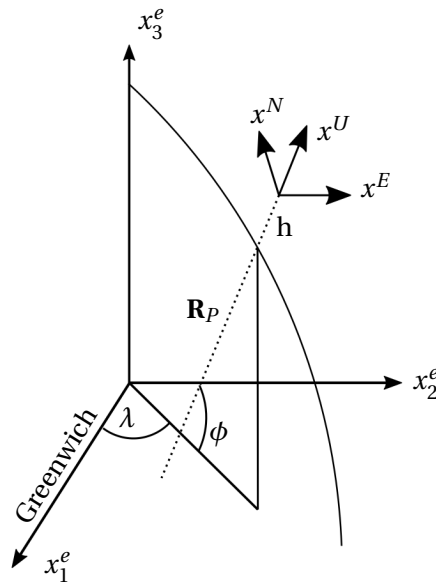


Figure 3.2: Earth *e*-frame and local (ENU) *l*-frame

Rotation from Earth to Local Frame

Without proof, the rotation from a local NED l -frame as depicted in Fig. 3.2 to the e -frame is given by the rotation matrix

$$\mathbf{R}_l^e = [\mathbf{n}^e \mathbf{e}^e \mathbf{d}^e] = \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\lambda) & -\cos(\phi)\cos(\lambda) \\ -\sin(\phi)\sin(\lambda) & \cos(\lambda) & -\cos(\phi)\sin(\lambda) \\ \cos(\lambda) & 0 & -\sin(\lambda) \end{bmatrix} \quad (3.40)$$

where the axes in the l -frame expressed in the e -frame are given by

$$\mathbf{d}^e = \begin{bmatrix} -\cos(\phi)\cos(\lambda) \\ -\cos(\phi)\sin(\lambda) \\ -\sin(\lambda) \end{bmatrix}, \mathbf{n}^e = \begin{bmatrix} -\sin(\phi)\cos(\lambda) \\ -\sin(\phi)\sin(\lambda) \\ \cos(\lambda) \end{bmatrix}, \mathbf{e}^e = \begin{bmatrix} -\sin(\lambda) \\ \cos(\lambda) \\ 0 \end{bmatrix} \quad (3.41)$$

From the skew-matrix definition (Sec. 3.12), we can derive

$$\dot{\mathbf{R}}_l^e = \mathbf{R}_l^e \boldsymbol{\Omega}_{el}^l = [\mathbf{n}^e \mathbf{e}^e \mathbf{d}^e] \begin{bmatrix} \cos(\phi)\dot{\lambda} \\ -\dot{\phi} \\ -\sin(\phi)\dot{\lambda} \end{bmatrix} \quad (3.42)$$

where $\boldsymbol{\Omega}_{el}^l$ is used to compute the latitudinal and longitudinal rates in the strapdown steps in l -frame (Sec. 3.4).

3.3.4 Body Frame

The body frame (b -frame) is often used to express the relative attitude of an object with respect to a local-level frame. Its origin is the same as the local-level frame from which an attitude is defined. In inertial navigation, its location often coincides with the center of an IMU. In VDM, the origin of the body frame is defined at the center of gravity of the platform. The axes form a right-handed Cartesian frame with the first x_b pointing forward, parallel to the direction of motion, the second axis y_b towards the right wing and the third z_b downward. A representation of the body frame is shown in Fig. 3.3.

Rotation from Local to Body Frame - Euler Parametrization

The rotation matrix from the local to body frame \mathbf{R}_l^b using Euler parametrization is defined as the sequence of three rotations around each axis. In aeronautics, these three angles are named roll, pitch and yaw (or heading) and are depicted in Fig 3.3 The order of the rotation matters and in this manuscript is defined with the rotation around the third axis first, following by the second and the first axis at the end

$$\mathbf{R}_l^b = \mathbf{R}_1(r) \cdot \mathbf{R}_2(p) \cdot \mathbf{R}_3(y) \quad (3.43)$$

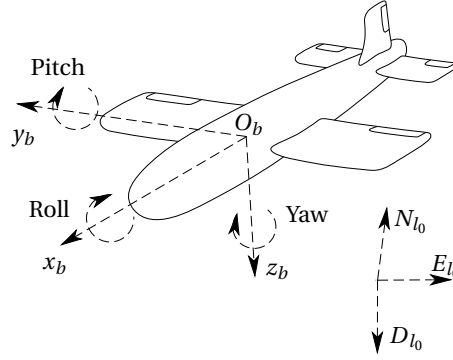


Figure 3.3: Rotation around the three body axes

The rotation is performed as a left matrix multiplication

$$\mathbf{x}^b = R_l^b \mathbf{x}^l. \quad (3.44)$$

3.3.5 Wind Frame

The wind frame (w -frame) has its first axis in the direction of the airspeed vector \mathbf{V} , and its orientation with respect to the body frame is defined by two angles: the angle of attack α and the side-slip angle β . The airflow velocity due to the UAV's inertial velocity \mathbf{v} and wind velocity \mathbf{w} is denoted by the airspeed vector and equals

$$\mathbf{V} = \mathbf{v} - \mathbf{w} \quad (3.45)$$

A representation of the frame is shown in Fig. 3.4 together with the body and local-mapping frames (the latter in NED orientation).

Rotation from Body to Wind Frame

The rotation from the body to the wind frame is given by the matrix \mathbf{R}_b^w

$$\mathbf{R}_b^w = \begin{bmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}, \quad (3.46)$$

3.3.6 Camera and IMU Frames

The camera frame (c -frame) origin is defined with the image sensor's principal axes. A representation of a camera as part of a payload is depicted in Fig. 3.5 The first axis x_c points

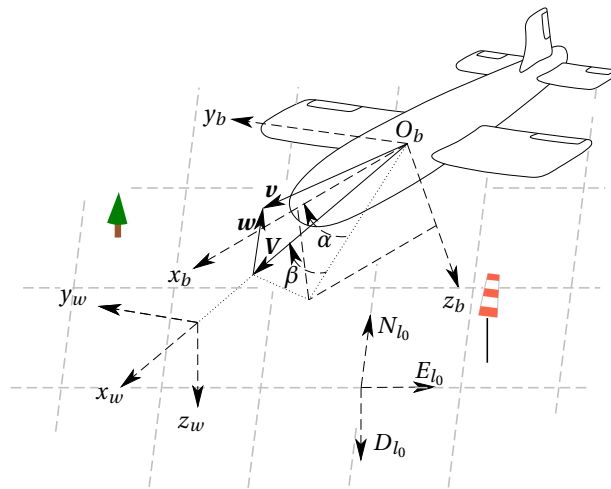


Figure 3.4: Local level, body, and wind frames with airspeed V , wind velocity w , and UAV velocity v , adapted from [47]

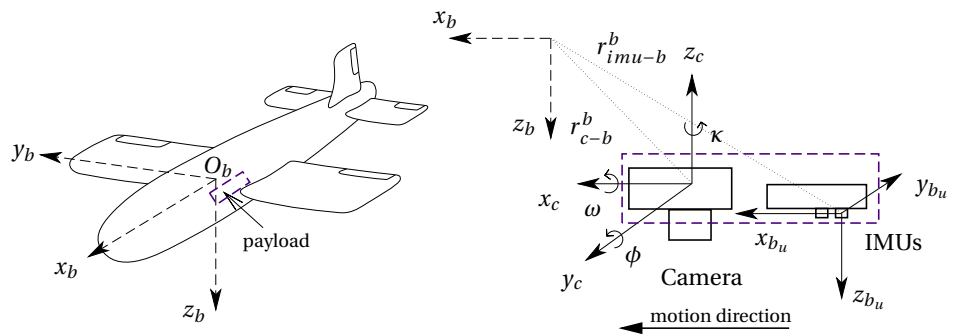


Figure 3.5: Body, camera and IMU frames

towards the UAV direction of motion while the third axis z_c points in the opposite direction of the camera optics. The y_c axis realizes the right-hand Cartesian frame. When calibrating the camera's exterior and interior orientation parameters, the bore-sight \mathbf{R}_c^b and the lever-arm \mathbf{r}_{c-b}^b between the camera and the body can be determined. Accounting for these parameters allows for the camera poses to be transformed to body-frame poses as

$$\mathbf{x}^b = \mathbf{R}_c^b \mathbf{x}^c + \mathbf{r}_{c-b}^b \quad (3.47)$$

The benefit of precise attitude observation obtained by photogrammetry for calibration is presented in Sec. 5.3.2.

The IMU-frame coincides with the triplet of accelerometer and gyroscope axes in the sensor assembly. A representation is given in Fig. 3.5. The origin, definition, and orientation of the IMU-frame is generally given by the constructor and follow the internal placement of each individual sensor. Its relation to the body frame is given by the bore-sight matrix \mathbf{R}_{imu}^b and lever-arm \mathbf{r}_{imu-b}^b .

$$\mathbf{x}^b = \mathbf{R}_{imu}^b \mathbf{x}^{imu} + \mathbf{r}_{imu-b}^b \quad (3.48)$$

Details about the IMUs used in this work are given in Sec. 6.2.

3.4 Navigation Equations

The principle of integrated navigation is based on the integration of measurements of an orthogonal triplet of accelerometers and gyroscopes rigidly mounted on a moving object. Usually, mounted together with supporting electronics, clocks, and communication, they form an IMU that, according to the realization principle, provides the specific forces (or velocity increments) and angular velocities (or angular changes) of the object. The steps presented hereafter are inspired by [17, 48].

Position, Velocity in Local Frame l

Without derivations, the summary of the navigation equation in the l -frame for position and velocity is given below.

Starting with the velocity vector $\mathbf{v}_e^l = [v^N, v^E, v^D]^T$, its differential equation is given by

$$\dot{\mathbf{v}}_e^l = \mathbf{R}_b^l \mathbf{f}^b - \left(\boldsymbol{\Omega}_{le}^l + 2\boldsymbol{\Omega}_{ie}^l \right) \mathbf{v}_e^l + \bar{\mathbf{g}}^l \quad (3.49)$$

and defines the velocity dynamics. $\bar{\mathbf{g}}^l$ embeds the local gravity and centrifugal acceleration as a function of the position of the local frame above the ellipsoid at height h and a global gravity

model (WGS84) [49]. $\boldsymbol{\Omega}_{ie}^l = [\boldsymbol{\omega}_{ie}^l]_{\times}$ is defined as

$$\boldsymbol{\omega}_{ie}^l = \omega_{ie} \begin{bmatrix} \cos(\phi) \\ 0 \\ -\sin(\phi) \end{bmatrix} \quad (3.50)$$

with ω_{ie} is the mean Earth angular velocity^{III}. $\boldsymbol{\Omega}_{el}^l = [\boldsymbol{\omega}_{el}^l]_{\times}$ represents the rotation of the l frame with respect to the e -frame expressed in the l -frame and corresponds to the local frame moving alongside a vehicle, and equals

$$\boldsymbol{\omega}_{el}^l = \begin{bmatrix} \dot{\lambda} \cos(\phi) \\ -\dot{\phi} \\ -\dot{\lambda} \sin(\phi) \end{bmatrix} \quad (3.51)$$

The derivation of $\boldsymbol{\omega}_{ie}^l$ is related to the time derivative of the rotation from l -frame to the e -frame (Eq. 3.42).

Defining the position vector in the l -frame with respect to the ellipsoid gives $r_e^l = [\phi, \lambda, h]$. The differential equation for the position is related to the velocity vector v_e^l as

$$\mathbf{r}_e^l = \mathbf{D}^{-1} \mathbf{v}_e^l \quad (3.52)$$

where the matrix \mathbf{D} is defined as

$$\mathbf{D} = \begin{bmatrix} R_M + h & 0 & 0 \\ 0 & (R_P + h) \cos(\phi) & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (3.53)$$

and R_P and R_M were defined in Eq. 3.36 and 3.37, respectively.

Attitude Equations in l -Frame

The attitude differential equations for Euler angles and quaternions are both used in this research (Sec. 2.3.1). However, for simplicity, the attitude equation will be given here in quaternion whereas the equations using Euler angles is given in Appendix B.3.1. The ODE for quaternion was developed and given in Eq. 3.27. The derivative of the attitude of the vehicle should be given in the l -frame with respect to the b -frame. The attitude is represented with the vector \mathbf{q}_b^l . Therefore the differential equation is given by

$$\dot{\mathbf{q}}_b^l = \frac{1}{2} \mathbf{q}_b^l \otimes [\boldsymbol{\omega}_{lb}^b]_q \quad (3.54)$$

^{III} $\omega_{ie} 7292115.0 \times 10^{-5}$ rad/s [50]

ω_{lb}^b is defined by

$$\omega_{lb}^b = \omega_{ib}^b - \mathbf{R}_l^b \omega_{il}^l \quad (3.55)$$

with ω_{ib}^b the rotation angle given by gyroscope measurements and $\omega_{il}^l = \omega_{ie}^l + \omega_{el}^l$ where both terms are defined in Eq. 3.50 and 3.51.

With Eq. 3.53, 3.49 and 3.54, the rigid body dynamic equations $\dot{\mathbf{x}}_n$ are defined:

$$\begin{bmatrix} \dot{\mathbf{r}}_e^l \\ \dot{\mathbf{v}}_e^l \\ \dot{\mathbf{q}}_b^l \end{bmatrix} = \begin{bmatrix} \mathbf{D}^{-1} \mathbf{v}_e^l \\ \mathbf{R}_b^l \mathbf{f}^b - (2\Omega_{ie}^l + \Omega_{el}^l) \mathbf{v}_e^l + \mathbf{g}^l \\ \frac{1}{2} \mathbf{q}_b^l \otimes [\omega_{ib}^b - (\mathbf{R}_b^l)^T (\omega_{ie}^l + \omega_{el}^l)]_q \end{bmatrix} \quad (3.56)$$

VDM-based navigation uses another quantity: the angular velocity ω_{ib}^b and is introduced in Sec. 3.5.

3.5 Specificities via Vehicle Dynamic Model

A complete derivation of the VDM for a fixed-wing UAV is given in [11]. The main aspects are synthesized here. In the VDM-based navigation, the angular velocity ω_{ib}^b with ODE is defined as

$$\dot{\omega}_{ib}^b = (\mathbf{I}^b)^{-1} [\mathbf{M}^b - \Omega_{ib}^b (\mathbf{I}^b \omega_{ib}^b)], \quad (3.57)$$

where \mathbf{I}^b and \mathbf{M}^b are the UAV matrix of inertia and specific moments. $\dot{\omega}_{ib}^b$ is added with the position, velocity and attitude (Eq. 3.56) equations and completes the rigid body motion definition.

In INS-based navigation, \mathbf{f}^b and ω_{ib}^b are observed with an IMU. In VDM, the specific forces \mathbf{f}^b and moments \mathbf{M}^b are defined as

$$\mathbf{f}^b = \begin{bmatrix} F_T^b \\ 0 \\ 0 \end{bmatrix} + (\mathbf{R}_b^w)^T \begin{bmatrix} F_x^w \\ F_y^w \\ F_z^w \end{bmatrix} \quad (3.58)$$

and

$$\mathbf{M}^b = \begin{bmatrix} M_x^b \\ M_y^b \\ M_z^b \end{bmatrix} \quad (3.59)$$

where

$$F_T^b = \rho n^2 D^4 (C_{F_T1} + C_{F_T2} J + C_{F_T3} J^2) \quad (3.60)$$

$$F_x^w = \bar{q} S C_{F_x} \quad (3.61)$$

$$F_y^w = \bar{q} S C_{F_y} \quad (3.62)$$

$$F_z^w = \bar{q} S C_{F_z} \quad (3.63)$$

$$M_x^b = \bar{q} S b C_{M_x} \quad (3.64)$$

$$M_y^b = \bar{q} S \bar{c} C_{M_y} \quad (3.65)$$

$$M_z^b = \bar{q} S b C_{M_z} \quad (3.66)$$

and \mathbf{R}_b^w is given in Eq. 3.46 and F_T^b denotes the thrust force model. F_x^w, F_y^w, F_z^w denote drag, lateral and lift forces, respectively. M_x^b, M_y^b, M_z^b denote aerodynamic moments. $b, S,$ and \bar{c} are the wing span, wing surface, and mean aerodynamic chord, respectively^{IV}. The air density is denoted by ρ , while \bar{q} is the dynamic pressure defined as

$$\bar{q} = \rho \frac{V^2}{2} \quad (3.67)$$

where V denotes airspeed. The aerodynamic model-parameters represented by C_{\dots} 's for the moments and forces are different for the two platforms used in this thesis. They are given in Tab. 3.1. The conventional fixed-wing UAV aerodynamic model is taken from [51] and is

Table 3.1: Aerodynamic coefficients for the forces and moments for the two platforms: *TP2* and *eBeeX*

Coefs.	TP2	eBeeX
C_{F_x}	$C_{F_x1} + C_{F_x\alpha}\alpha + C_{F_x\alpha2}\alpha^2 + C_{F_x\beta2}\beta^2$	$C_{F_{x0}} + C_{F_{x\alpha}}\alpha + C_{F_{x\alpha2}}\alpha^2 + C_{F_{x\beta}}\beta + C_{F_{x\delta_e}}\delta_e + C\boldsymbol{\theta}_{AF_x}$
C_{F_y}	$C_{F_y1}\beta$	$C_{F_{y\beta}}\beta + C_{F_{y\beta2}}\beta^2 + C_{F_{y\delta_a}}\delta_a + C\boldsymbol{\theta}_{AF_y}$
C_{F_z}	$C_{F_z1} + C_{F_z\alpha}\alpha$	$C_{F_{z0}} + C_{F_{z\alpha}}\alpha + C_{F_{z\alpha2}}\alpha^2 + C_{F_{z\beta}}\beta + C_{F_{z\delta_e}}\delta_e + C\boldsymbol{\theta}_{AF_z}$
C_{M_x}	$C_{M_x\alpha}\delta_a + C_{M_x\beta}\beta + C_{M_x\tilde{\omega}_x}\tilde{\omega}_x + C_{M_x\tilde{\omega}_z}\tilde{\omega}_z$	$C_{M_{x\beta}}\beta + C_{M_{x\beta2}}\beta^2 + C_{M_{x\delta_a}}\delta_a + C\boldsymbol{\theta}_{AM_x}$
C_{M_y}	$C_{M_y1} + C_{M_y\delta_e}\delta_e + C_{M_y\tilde{\omega}_y}\tilde{\omega}_y + C_{M_y\alpha}\alpha$	$C_{M_{y0}} + C_{M_{y\alpha}}\alpha + C_{M_{y\alpha2}}\alpha^2 + C_{M_{y\delta_e}}\delta_e + C\boldsymbol{\theta}_{AM_y}$
C_{M_z}	$C_{M_z\delta_r}\delta_r + C_{M_z\tilde{\omega}_z}\tilde{\omega}_z + C_{M_z\beta}\beta$	$C_{M_{z\beta}}\beta + C_{M_{z\beta2}}\beta^2 + C_{M_{z\delta_a}}\delta_a + C\boldsymbol{\theta}_{AM_z}$

referred to as *TP2* in the remaining part of this work. The delta-wing UAV model, referred to as *eBeeX*, is designed to be as generic as possible. The platforms are depicted in Fig. 3.6 and Fig. 3.7 for the *TP2* and *eBeeX*, respectively.

The thrust model is similar for both platforms with different coefficients C_{F_T} 's. J is defined as $V/(D\pi n)$ with n denoting the propeller rotation speed and D is the propeller diameter. The non-dimensional angular velocities are defined as $\tilde{\omega}_x = b\omega_x(2V)^{-1}$, $\tilde{\omega}_y = \bar{c}\omega_y(2V)^{-1}$, and $\tilde{\omega}_z = b\omega_z(2V)^{-1}$, where $[\omega_x \ \omega_y \ \omega_z]^T$ denotes the angular velocity of UAV with respect to the b -frame. The deflections of control surfaces: aileron, elevator, and rudder are denoted by $\delta_a,$

^{IV}Discussion on these quantities is given in Appendix C.3.1, and a method to determine the wing surface S and length via orthophoto is given in Appendix C.3.2

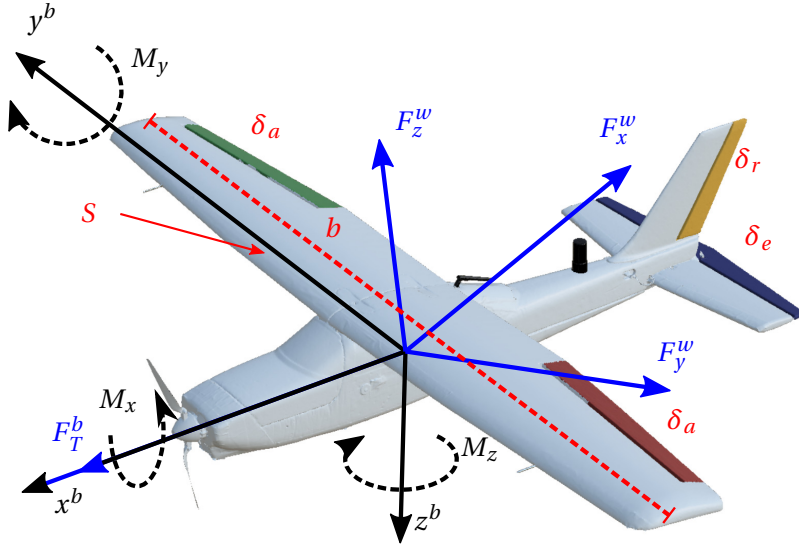


Figure 3.6: $TP2$ control surfaces, positive angular rotations (black) around body frame axis, forces (blue)

δ_e , and δ_r respectively.

For the eBeeX, $\delta_a = \frac{-\delta_L + \delta_R}{2}$, $\delta_e = \frac{\delta_L + \delta_R}{2}$. It should be noted that a delta wing UAV does not have real elevators and ailerons, rather it has two independent control surfaces called elevons, where the deflections are denoted by δ_L and δ_R , with the subscripts L, R represent the left and right deflections, respectively. Additionally, $C\theta$ is defined as

$$C\theta_{A_i} = C_{A_i\tilde{\omega}_x} \tilde{\omega}_x + C_{A_i\tilde{\omega}_y} \tilde{\omega}_y + C_{A_i\tilde{\omega}_z} \tilde{\omega}_z \quad (3.68)$$

where $A \in \{F, M\}$ distinguishes forces from moments, and $i \in \{x, y, z\}$ defines the three axis.

3.5.1 Estimation Scheme

VDM serves as the main process model within the filter as shown in Fig. 3.8. An EKF is chosen to estimate corrections to the states ($\delta\mathbf{x}$) and the associated covariance matrix (\mathbf{P}). As depicted in Fig. 3.8, the VDM provides the navigation solution (\mathbf{x}_n), which is updated as part of the augmented state vector \mathbf{x} (introduced in Equation (3.69)) based on available observations.

As IMU data are treated as observations, the navigation system will stop using IMU data in the case of an IMU failure, and instead provides the navigation solution using VDM. Other sensors can be integrated into the navigation system, such as airspeed, optic flow, and magnetometer data, or precise attitude reference, which provide observations when available.

The VDM is fed with the UAV control input (\mathbf{U}) as commanded by the autopilot and is therefore always available. Wind velocity (\mathbf{x}_w) is also required as the equation for these forces are solved

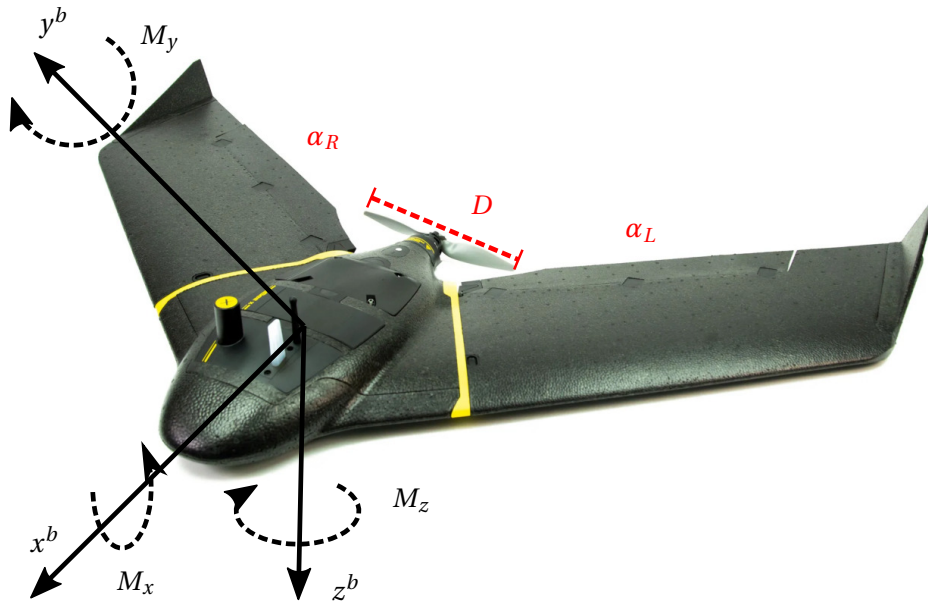


Figure 3.7: eBeeX control surfaces and positive angular rotations around body frame (black)

within this frame (Eq. 3.61 to 3.63). The required VDM parameters (\mathbf{x}_p) are mainly composed of the aerodynamic coefficients presented in Tab. 3.1. The actuator deflections are included as states (\mathbf{x}_a). IMU errors (\mathbf{x}_e) that are generalized as biases are included within the augmented state vector to be estimated. The augmented state vector \mathbf{x} therefore includes the navigation states \mathbf{x}_n , the VDM parameters \mathbf{x}_p , the actuator deflections \mathbf{x}_a , the wind velocity components \mathbf{x}_w and the sensor error states \mathbf{x}_e . More auxiliary states can be added as sensor misalignment and lever-arm.

$$\mathbf{x} = [\mathbf{x}_n^T, \mathbf{x}_p^T, \mathbf{x}_a, \mathbf{x}_w^T, \mathbf{x}_e^T]^T \quad (3.69)$$

More details on these augmented states is given in Sec. 3.5.2. The mass (m) and moments of inertia \mathbf{I}^b are excluded from the VDM parameter auxiliary states \mathbf{x}_p since they appear as scaling factors in the model, meaning that they are completely correlated with the aerodynamic coefficients [47] that are already included. The platform-dependant geometric properties (D , S , b , \bar{c}) are also excluded, because they can be determined a priori with much lower uncertainty compared to aerodynamic coefficients.

3.5.2 State Space

Details of state space and auxiliary states are given below.

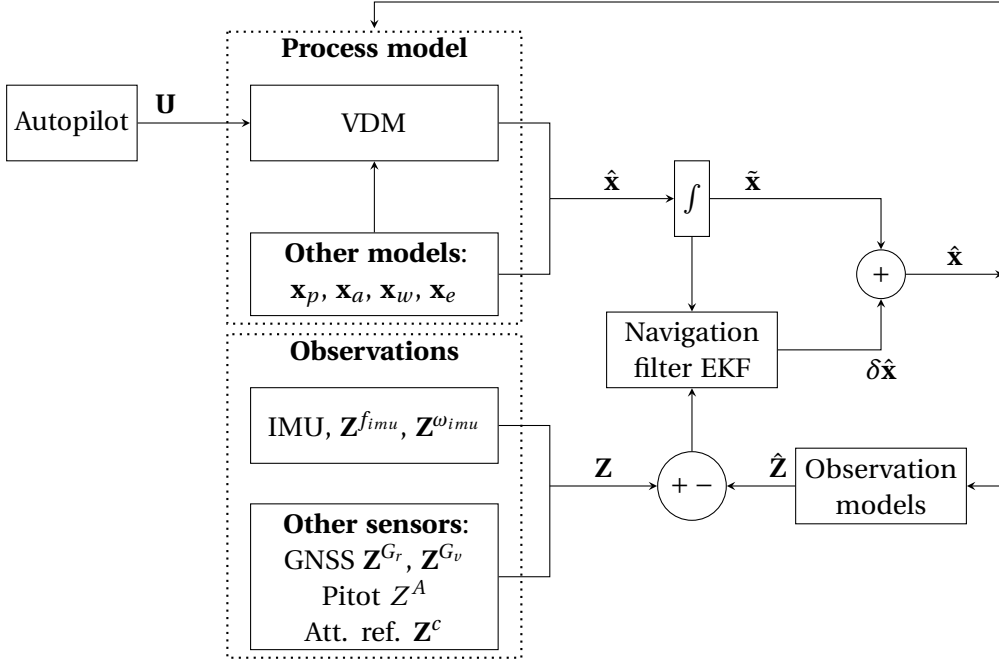


Figure 3.8: VDM-based navigation filter architecture ($\tilde{\mathbf{x}}_k \equiv \hat{\mathbf{x}}_{k|k-1}$), adapted from [47]

Navigation States

The navigation state

$$\mathbf{x}_n = [\mathbf{r}_e^l, \mathbf{v}_e^l, \mathbf{q}_b^l, \boldsymbol{\omega}_{ib}^b]^T \quad (3.70)$$

are composed of: the position $\mathbf{r}_e^l = [\phi, \lambda, h]$ in the e -frame in ellipsoidal coordinates in $[rad]$ for the horizontal position and in $[m]$ for the height. The velocity $\mathbf{v}_e^l = [v_n^l, v_e^l, v_d^l]$ with respect to the e -frame is expressed in the l -frame in $[m/s]$, the attitude $\mathbf{q}_b^l = [q_0, q_1, q_2, q_3]$ of the body with respect to the l -frame, and the body angular velocity $\boldsymbol{\omega}_{ib}^b = [w_x^b, w_y^b, w_z^b]$ with respect to the i -frame, expressed in the b -frame in $[rad/s]$. The dynamic model $\dot{\mathbf{x}}_n$ is given in Eq. 3.56 for position, velocity and attitude, and in Eq. 3.57 for the angular velocity. The VDM-based navigation system works when the UAV is airborne. Thus, the initial navigation states $\mathbf{x}_n(0)$ and their covariances $\mathbf{P}_n(0)$ have to be initialized from another source, e.g an INS/GNSS navigation software (Sec. 7.2) running in parallel.

Actuator States

The augmented actuator states are defined as

$$\mathbf{x}_a = [a, e, r, n]^T \quad (3.71)$$

The UAV control surfaces deflections a , e and r with the propeller speed n , are arguments of the VDM force and moment equations and are added as states. Two dynamic factors k_1 and k_2 with a time delay τ model a first order delay between the flight control input from the autopilot $C_i = [a_c, e_c, r_c]$ and n_c , respectively, and the actual actuator deflection / states \mathbf{x}_a . The dynamic equation $\dot{\mathbf{x}}_a$ is similar for the four actuators and reads

$$\dot{\mathbf{x}}_{a_i} = \mathbf{x}_{a_i} = \frac{k_{1_i} C_i + k_{2_i} + \mathbf{x}_{a_i}}{\tau_i} \quad (3.72)$$

where $i = [a, e, r, n]$. The flight Control Commands (CC)s come from the Pulse-Width Modulation (PWM) (values from 1000 to 2000) accessed from the autopilot. For a , e , and rc these PWM are mapped to a normalized unit-less deflection value $[-1; 1]$ as specified in [51], and to the unit-less values $[0 - 1000]$ for n . The dynamic factors k_{1_i} , k_{2_i} and delays τ_i can be modeled as additional augmented states with their own dynamic model, as proposed in [11]. However, these augmented states are not added in the state space in this research due to poor observability of the actuator dynamics, and only little impact on the overall navigation performance is observed when added as auxiliary states in simulations. Therefore, they are set as constant parameters with values proposed in [11]. However, incorrect time-tagging of the CCs has an impact in the navigation performance (Sec. 9.1.1).

The initial actuator state $\mathbf{x}_a(0)$ is initialized with the flight CC given directly by the autopilot. The initial uncertainty (1σ) $\mathbf{P}_a(0)$ for the control surfaces (a , e , r) is set to 0.016 which relates to $\sim 1[deg]$ of uncertainty considering that the PWM maps from $[-1; 1]$ the control surface deflection ranging between around $+/- 45[deg]$. The uncertainty of the propeller speed corresponds to $20[rad/s]$ following the experiment from [11].

Wind velocities

The force and moment equations are solved with respect to the airspeed vector $\mathbf{V}_{eb}^l = \mathbf{v}_{eb}^l - \mathbf{w}_{ew}^l$ composed of the wind velocity \mathbf{w} and the platform inertial ground velocity \mathbf{v} . The wind velocity therefore needs to be estimated and is added as an augmented state vector

$$\mathbf{x}_w^l = [x_w^N, x_w^E, x_w^D]^T \quad (3.73)$$

The wind components are expressed in the local l -frame, in the same frame as the navigation states. The wind airflow at cruising altitude and, in normal weather conditions^V is assumed to have a slow rate of change during a flight mission. No deterministic part is considered in the dynamic model of the wind $\dot{\mathbf{X}}_w = 0$. However, variations in wind direction and magnitude are captured with its process noise Q_w as white noise; the higher the process noise, the higher the bandwidth of the state changes. The resulting wind velocities are modeled as a random walk with parameterized standard deviation as suggested in [52]. The initial wind velocities \mathbf{x}_w can be either initialized to zeros or approximated if a local weather station (Sec. 6.3.4) or a close

^Vwind less than 10 m/s. The 2.7 kg UAV will not be sent on a mission if the wind conditions are too severe

Meteorological Aerodrome Report (METAR) is available. Their initial uncertainties (1σ) are set to 0.5 [m/s] in horizontal and 0.1 [m/s] in vertical directions unless otherwise specified in a specific experiment.

Aerodynamic Coefficients

Called the VDM parameters more generally, these coefficients characterize the physical behaviour of the platform through the air via the moment and force equations previously defined. Some of these coefficients (parameters) can be easily determined as the weight of the platform m , or the diameter of the propeller D . However, some other coefficients are partially or completely unknown, are complex to obtain, and need to be estimated. In addition, some coefficients can change slightly due to a modification of the payload, the position of the battery in its bay, and the addition of a sensor, potentially modifying the UAV center of mass. The augmented states \mathbf{x}_p consist of all the aerodynamic coefficients C_i 's presented in Tab. 3.1 for the *TP2* or the *eBeeX*. The initial states $\mathbf{x}_p(0)$ are either adapted from a similar platform, estimated via a calibration procedure (Sec. 5.2) or refined (Sec. 5.3.2, Sec. 5.3.1) if an approximated set of parameters is already known. These coefficients are assumed to be constant for a particular UAV and therefore the differential equation for these states is null

$$\dot{\mathbf{x}}_p = 0 \quad (3.74)$$

However, small geometric variation between flights need to be taken into account, which will be captured by the aerodynamic coefficients. Therefore, nonzero uncertainties $\mathbf{P}(0)$ are used for these states and are set (1σ) to 1-2% of their initial values.

3.5.3 Observation Models

The different sensors used for this research are listed in the sequel with their observation models.

IMU

IMUs provide high frequency updates of the navigation states as well as corrections for other augmented ones. IMUs come in a wide range of quality, performance, stability and accuracy. Yet, the observation model for all of them is the same, with the systematic and stochastic error models to differentiate between them. Assuming that the system is rigidly attached to the body, the observation model equations for the accelerometer is given by

$$\mathbf{z}^{f_{imu}} = \mathbf{R}_b^{imu} \left(\mathbf{f}^b + \left(\dot{\Omega}_{ib}^i + \Omega_{ib}^i \Omega_{ib}^i \right) \mathbf{r}_{b-imu}^b \right) + \mathbf{x}_e^{f_{imu}} + \boldsymbol{\epsilon}_{f_{imu}} \quad (3.75)$$

The rotation matrix \mathbf{R}_b^{imu} corresponds to the boresight between the *IMU*-frame and *b*-frame and, \mathbf{r}_{b-imu}^b being the lever-arm between the position of the IMU and the origin of the *b*-frame

(Fig. 3.5).

The two terms $\dot{\Omega}_{ib}^i$ and $\Omega_{ib}^i \Omega_{ib}^i$ are the Coriolis and centrifugal accelerations experienced by the rotating frames with respect to each other^{VI}. Finally, $\mathbf{x}_e^{f_{imu}}$ and $\boldsymbol{\epsilon}_{f_{imu}}$ are the sensor error modeled by a choice of numerous processes (e.g. white, Gauss-Markov, random walk) and the measurement noise, respectively.

For the gyroscope, the observation model equation is given by

$$\mathbf{Z}^{gyr} = \mathbf{R}_b^{imu} \boldsymbol{\omega}_{ib}^b + \mathbf{x}_e^{\omega_{imu}} + \boldsymbol{\epsilon}_{\omega_{imu}} \quad (3.76)$$

The lever-arm \mathbf{r}_{b-imu}^b only affects the accelerometer measurements. The augmented IMU error states are defined as

$$\mathbf{x}_e^{imu} = \left[\mathbf{x}_e^{f_{imu}}, \mathbf{x}_e^{\omega_{imu}} \right]^T \quad (3.77)$$

and both sub-vectors $\mathbf{x}_e^i, i \in [f_{imu}, \omega_{imu}]$ are composed of three biases (random walk) for each triplet of accelerometer and gyroscope $\mathbf{x}_e^i = [e_{RW}^{i1}, e_{RW}^{i2}, e_{RW}^{i3}]$. Therefore, $\dot{\mathbf{x}}_e^{imu} = 0$. The initial values $\mathbf{x}_e^{f_{imu}}(0), \mathbf{x}_e^{\omega_{imu}}(0)$ are set to zero using to a calibration procedure [53] removing switch-on bias, and their strength and uncertainties are taken from the data-sheet of the manufactures and a summary of the error statistics is given in Tab. C.2.1 for the different IMU used in this work.

More complex error models can be used as suggested by [11], e.g. Auto-Regressive (AR), Gauss Markov (GM) or quantization noise processes. For the MEMs-IMU used in the experiments (Sec. 6.2), the presented model seems to be sufficient. The lever-arm \mathbf{r}_{b-acc}^b and boresight \mathbf{R}_b^{acc} can also be added as an augmented state to be estimated. In the current implementation, they are defined as constant parameters where the boresight matrix is assumed to be close to identity and the lever-arm is observed thanks to[54].

GNSS Position and Velocity

The GNSS receiver determines the position and velocity of the antenna using different techniques (e.g. SPP, Precise Point Positioning (PPP), PPK, RTK), which will not be discussed here. The solution can be expressed in different e -frames and datum, e.g. WGS84. The GNSS position observation model is given by

$$\mathbf{Z}^{Gr} = \mathbf{r}_{eb}^l + \mathbf{R}_b^e \mathbf{r}_{bG}^b + \boldsymbol{\epsilon}_{Gr} \quad (3.78)$$

where \mathbf{r}_{bG}^b is the lever-arm from the b -frame origin and the GNSS antenna, $\mathbf{R}_b^e = \mathbf{R}_l^e \mathbf{R}_b^l$ is the rotation matrix from the b -frame to the e -frame (Eq. 3.43 and 3.40), and $\boldsymbol{\epsilon}_{Gr}$ is the measurement noise.

^{VI}The Coriolis accelerations and the varying direction of the gravity vector can accumulated an error of a several hundreds of meters of inertial coasting over a period of 5 minutes if not taken into account [11]

Chapter 3. Background of Integrated Navigation with Vehicle Dynamic Model

The GNSS velocity observation model is given by the derivative over time of the GNSS position observation model \mathbf{Z}^{G_r} and is given by

$$\mathbf{Z}^{G_v} = \mathbf{v}_{eb}^l + \left(\mathbf{R}_b^l \boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{ie}^l \mathbf{R}_b^l \right) \mathbf{r}_{bG}^b + \boldsymbol{\epsilon}_{G_v} \quad (3.79)$$

with the skew and rotation matrices already defined in Sec. 3.1. The observations noise $\boldsymbol{\epsilon}_{G_v}$ depends on the GNSS. Typical noise levels are given in Tab. C.4 for SPP and PPK.

Airspeed Sensor

The Pitot tube or airspeed sensor measures the total moving air mass and estimates the dynamic pressure q by removing the static pressure p

$$p_{tot} = q + p \quad (3.80)$$

The dynamic pressure q is transformed to a velocity as a parameters of the air density ρ . The airspeed sensor (one-hole Pitot tube) relates the velocity $\mathbf{V}^b = \mathbf{v}^b - \mathbf{w}^b$ as defined in Sec. 3.5.2, in the b -frame, reduced to the only direction to where the sensor points, ideally, the forward direction of the UAV body axis x_1^b . A boresight, between the sensor and the toward axis x_1^b is characterized by two angles α_A and β_A . The two angles can be used to project the airspeed vector \mathbf{V}^l in the 1D axis of the airspeed reference frame with

$$\mathbf{D} = [\cos(\alpha_a)\cos(\beta_a), \cos(\alpha_a)\sin(\beta_a), \sin(\beta_a)]^T \quad (3.81)$$

The Pitot tube is located with a lever-arm r_{b-A}^b with respect to the origin of the b -frame. The velocity of the sensor \mathbf{v}_{eA}^l with respect to the l -frame is expressed equivalently as Eq. 3.79. Therefore, the observation equation gives

$$Z_A = s_A \left(\mathbf{v}_{eA}^l + \left(\mathbf{R}_b^l \boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{ie}^l \mathbf{R}_b^l \right) \mathbf{r}_{b-A}^b - \mathbf{w}_{eb}^l \right) \mathbf{R}_l^b \mathbf{D} + \boldsymbol{\epsilon}_A \quad (3.82)$$

where the only term left to be defined is the scale factor s_A . Only a scale factor is assumed as the error term because the initial bias can be removed via calibration. The modification of atmospheric pressure within the UAV's operational altitude (± 150 m) can be approximated as linear [55]. The scale factor also depends on the varying air density ρ and must be calibrated before each flight as proposed in Sec. 9.2. The conversion between the different airspeed indicators is given in Appendix B.1

3.5.4 Linearization

Each platform possesses its specific VDM and set of sensors. The models are implemented as nonlinear functions in the navigation estimator. Most need to be linearized and their respective Jacobian matrices created, e.g. the Jacobian of the dynamic matrix $F()$ is needed to obtain the transition matrix Φ or the linearized measurements matrix $H()$ from the sensor

model $h()$. The complex process (and possibly observation) models and the high number of states render the linearization somewhat challenging if performed without automation. On the other hand, numerical derivations are reliable, but their execution time is not suitable for real-time operation. [11] proposed an automatic linearization scheme using the `symbolic toolbox` [56] from MATLAB. The tool makes the modification and testing of models an easy task (some model linearizations take several hours). Furthermore, the tool creates the matrices needed by the estimation software to execute the ESEKF steps as presented in Tab. 2.2.

Although the current tool is useful, there is a need to generate the C++ source code of these functions to ease real-time application. A second tool was developed and is detailed in Sec. 7.3.3 when real-time VDM-based navigation software is introduced.

Summary

This chapter has covered different reference frames relevant to aerial navigation. Additionally, the differential equations governing the inertial and VDM-based navigation were presented. The aspects improving the numerical stability of the solution are addressed in the next chapter.

VDM Enhancement **Part II**

4 Estimation Enhancements

Overview

This chapter focuses on improving the fusion between VDM and other sensors in terms of numerical stability to obtain an optimal estimation of UAV's trajectory. In the prior art, trajectory estimation is carried by a recursive Bayes filter that sequentially calculates the posterior probabilities of multiple beliefs to infer on position, velocity, and orientation of the drone from observations - GNSS and IMU. As we are considering normally distributed variables in linearized approximation around the previously corrected parameters (states), the Bayes filter becomes equal to the EKF, the structure of which follows the original VDM-based navigation filter described in [11].

The proposed enhancement brings in three aspects that improve the numerical stability of estimation. First, the magnitude of error-state variables is homogenized by adapting their scale (units). This decreases so-called *conditioning number*, in other words, diminishes the sensitivity that small perturbations in innovation cause large changes in the estimated quantities (through oscillating Kalman gain). Second, a numerically stable factorization is implemented (Bierman-Thorthon) that prevents round-off errors and asymmetry in covariance matrices. Third, the partial-Schmidt Kalman filter is proposed to be used in two critical phases of VDM-based navigation: the initialization and the navigation during GNSS outages. It is practically investigated that all together, this methodology i) prevents the numerical instability of the filter, ii) reduces the correlations between estimated parameters and, iii) avoids large variations in the estimated positions during the filtering of the two most challenging phases of the flight (the initialization and GNSS outage). For these reasons the proposed approach is favorable to be used for the drone's guidance.

4.1 Numerical Stability

Starting in the 1960s, the numerical instability of the Kalman filter has been observed using only 6 states [57]. The concept of constraining state space estimators using computers is not

new [58], where this is well analyzed for inertial navigation [59]. Although the general fixes are known [60], the potential numerical weaknesses of the VDM-based navigation implementation needs to be identified and a strategy for their mitigation to be proposed. The two types of problems leading to numerical instabilities are: a) ill-conditioning due to large differences in state values (round-off error) and, b) asymmetry of the covariance matrix. While symmetry can be forced using a simple yet redundant operation such as $P = (P + P^T)/2$, round-off errors are complex to avoid and correct as they occur during mathematical operations with limited precision. Moreover, a large difference in state magnitudes can render the matrix singular, where performing an error-free inversion is almost impossible [60][61].

4.1.1 Indicators

An ill-conditioned indicator proposed in [60] looks at the ratio of the largest (λ_{max}) and the smallest (λ_{min}) eigenvalues of matrix S in the update KF equations (Tab. 2.1)

$$cond(S) = \frac{|\lambda_{max}|}{|\lambda_{min}|} \quad (4.1)$$

The following rule of thumb can be used to ensure a well-conditioned matrix [59]:

$$cond(S) \ll \frac{1}{2^{-N}} \quad (4.2)$$

where N is the number of bits used in the mantissa. The MATLAB simulation environment with a 64-bit architecture uses 52 bits for its mantissa¹ and therefore, the condition number should be below 10^{15} to guarantee numerical stability of the system. In the real-time prototype, the VDM filter runs in a 64-bit architecture companion computer (Sec. 6.2) with a similar floating representation. The round-off errors on the solution of the system $Ax = b$ are bound by the following formula [62]:

$$\|\overline{(A^{-1}b)} - (A^{-1}b)\| \leq \epsilon cond(A) \|(A^{-1}b)\| \quad (4.3)$$

where $\|\cdot\|$ is the norm and $\overline{A^{-1}b}$ is the exact solution of the system. The error in the resolution of a linear system is directly proportional to the condition number of the inverted matrix as shown in Eq. (4.3). Although numerical errors do not necessarily lead to filter divergence, the filter can become momentarily unstable, resulting in a slower steady-state convergence [60] or incorrect state estimation. In addition, small values in the matrices may be rounded to 0 during computations leading to a change in sign for the final result. In such situations, the Kalman gain matrix K (Tab. 2.1) can incorrectly adjust certain states.

¹https://www.mathworks.com/help/matlab/matlab_prog/floating-point-numbers.html

4.1.2 Scaling

In [63], the body frame is chosen as a local tangent plane with its origin fixed at the home position of the drone. Later generalization considered navigation in the global frame (in WGS84 ellipsoidal coordinates) while taking into account the Earth effects: Coriolis acceleration, changes of gravitational force in direction and magnitude [11]. These changes induced the expression of the navigation states in latitude and longitude in $[rad]$ and, height in $[m]$. When using these units for their corrections within the filter states \mathbf{x}_n , the values come close to machine precision. For example, a $30cm$ displacement will, in the case of a spherical Earth approximation, be in the order of a few 10^{-8} radians. Furthermore, advances in GNSS technology have enabled improved pose estimation, decreasing its corresponding variance. For example, when kept in radians, the achievable centimeter accuracy in position with RTK or PPK results in a standard deviation of approximately $10^{-9}[rad]$, which creates a variance in the corresponding covariance matrix P of $10^{-18}[rad^2]$.

The scaling process follows two steps. First, the reduction of the condition number needs identification of the extreme (smallest and largest) variances in the initial covariance matrix $\mathbf{P}(0)$. For the case of VDM, these are the position ($10^{-18}[rad^2]$) and the propeller speed ($10^2[rad^2]$). The initial condition number for $\mathbf{P}(0)$ is therefore large ($\approx 10^{24}$), strongly exceeding the aforementioned limit of 10^{15} . Second, the chosen scaling factors D are applied to the problematic states to reduce the condition number of the covariance matrix $\mathbf{P0}$: $D_1 = 10^8$ for latitude/longitude and $D_2 = 10^{-2}$ of the propeller speed. In practice, a scaled vector \mathbf{x}_s is obtained from the initial state vector \mathbf{x} multiplied by the scaling diagonal matrix $\mathbf{D}_x = [D_1, D_2]^T$. This transformation can be seen as an arbitrary change of units for (a few) selected variables. Scaling the problematic states is done by adapting related matrices as summarized in Tab. 4.1 and detailed below. As seen in Eq. (4.3), the propagation of round-off errors while performing

Table 4.1: Adapted scaled matrix for the discrete Extended Kalman filter steps

Prediction scaled matrices	Update scaled matrices
$\mathbf{x}_s = \mathbf{D}_x \mathbf{x}$	$\mathbf{Z}_s = \mathbf{D}_z \mathbf{Z}$
$\mathbf{P}_s = \mathbf{D}_x \mathbf{P}_u \mathbf{D}_x^T$	$\mathbf{H}_s = \mathbf{D}_z \mathbf{H}_u \mathbf{D}_x^{-1}$
$\Phi_s = \mathbf{D}_x \Phi_u \mathbf{D}_x^{-1}$	$\mathbf{R}_s = \mathbf{D}_z \mathbf{R}_u \mathbf{D}_z^T$
$\mathbf{Q}_s = \mathbf{D}_x \mathbf{Q}_u \mathbf{D}_x^T$	$\mathbf{S}_s = \mathbf{D}_z \mathbf{S}_u \mathbf{D}_z^T$

the matrix inversion is proportional to the condition number. The inversion occurs in the computation of the Kalman gain \mathbf{K}_k , specifically when the expression $\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}$ is inverted. By scaling the states only, the condition number of the inverted matrix \mathbf{S}_s does not change.

$$\mathbf{S}_s = \mathbf{H}_s \mathbf{P}_s \mathbf{H}_s^T + \mathbf{R} = \mathbf{H}_u \mathbf{D}_x^{-1} \mathbf{D}_x \mathbf{P}_u \mathbf{D}_x^T (\mathbf{H}_u \mathbf{D}_x^{-1})^T + \mathbf{R} = \mathbf{H}_u \mathbf{P}_u \mathbf{H}_u^T + \mathbf{R}_u = \mathbf{S}_u \quad (4.4)$$

The units of \mathbf{S} are controlled by the measurement noise \mathbf{R} and observation \mathbf{H} matrices. Hence in the third step, these matrices must also be scaled by the noise scaling diagonal matrix \mathbf{D}_z (note: if the noise and the states have the same units, $\mathbf{D}_z = \text{diag}(\mathbf{D}_x)$) to correspond to the

new state vector \mathbf{x}_s . They are obtained as:

$$\mathbf{H}_s = \mathbf{D}_z \mathbf{H}_u \quad (4.5)$$

$$\mathbf{R}_s = \mathbf{D}_z \mathbf{R}_u \mathbf{D}_z^T \quad (4.6)$$

The potentially problematic matrix becomes:

$$\mathbf{S}_s = \mathbf{D}_z (\mathbf{H}_u \mathbf{P}_u \mathbf{H}_u^T + \mathbf{R}_u) \mathbf{D}_z^T \quad (4.7)$$

By carefully selecting the elements of \mathbf{D}_z , the magnitude of the elements of \mathbf{S} can be adjusted to be more homogeneous, which in turn lowers the condition number of the matrix. In the same manner, the observations z related to the scaled states should be adapted to match the corresponding scaled observation matrix to compute a meaningful innovation: $\mathbf{z} - \mathbf{H}\mathbf{x} \Rightarrow \mathbf{D}_z \mathbf{z} - \mathbf{H}_s \mathbf{x}_s$.

An example of condition number reduction is given in Appendix A.1. The “scaled” version of ESEKF is implemented and tested in MATLAB (post-processed). The C++ (real-time) environments do not yet have the scaled version implemented. The modifications are not heavy: they require only the definition of the new scaled VDM in `Mathematica` and the usage of the automatic source files generation (Sec. 7.3.3).

The state space re-scaling improves the numerical stability during the state estimation. However, it does not improve the estimation stability when the observability is reduced (e.g. at initialization or during a GNSS outage) or when it is not in a steady state. In the following section, a combination of factorization with a modified KF is proposed to improve the apprehensions mentioned above.

4.2 Factorization and Schmidt-Kalman

The classical KF may work well for situations in which only a few states are considered. However, as the state space becomes large, filter divergence and non-positiveness of \mathbf{P} can occur due to nonlinear effects affecting the numerical aspects of the filter [64].

4.2.1 UDU Implementation

The first method exploits one particular form of factorization put forward by [65], also known as *UDU*-factorization, which separates the covariance matrix as $\mathbf{P}(0) = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where \mathbf{U} and \mathbf{D} are the upper triangular and diagonal matrices, respectively. The *UDU* algorithm is presented in Alg. 1 with an extra verification step to ensure the positiveness of the decomposition. The *UDU* decomposition is performed only once at the initialization of the KF. The ESEKF steps have to be adapted to use the new matrices U and D , and are summarized in Tab. 4.2.

Table 4.2: ESEKF with UDU factorization

Prediction steps	Update steps
$\tilde{\mathbf{x}}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k, \delta t)$ $\tilde{\mathbf{U}}_{k+1}, \tilde{\mathbf{D}}_{k+1} = Thor.(\hat{\mathbf{U}}_k, \hat{\mathbf{D}}_k, \mathbf{Q}_k, \mathbf{G}_k)$	$\delta \tilde{\mathbf{z}}_k = \tilde{\mathbf{z}}_k - h(\tilde{\mathbf{x}}_{k+1})$ $\mathbf{K}_k, \hat{\mathbf{U}}_{k+1}, \hat{\mathbf{D}}_{k+1} = Bier.(\tilde{\mathbf{U}}_{k+1}, \tilde{\mathbf{D}}_{k+1}, \mathbf{H}_k, \mathbf{R})$ $\delta \hat{\mathbf{x}}_k = \mathbf{K}_k \delta \tilde{\mathbf{z}}_k$ $\hat{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_{k+1} + \delta \hat{\mathbf{x}}_k$

The Thornton prediction time update (*Thor. in Tab. 4.2*) of the covariance $\tilde{\mathbf{U}}_{k+1}$ and $\tilde{\mathbf{D}}_{k+1}$ requires the updated covariance $\hat{\mathbf{U}}_k$ and $\hat{\mathbf{D}}_k$ at the previous filter step k and the discrete-time process noise \mathbf{Q}_k and noise shaping matrix \mathbf{G}_k (Eq.B.10, Eq.B.16) which already embedded the discrete integrator δt . The Bierman measurement update (*Bier. in Tab. 4.2*) modifies the covariance $\hat{\mathbf{U}}_{k+1}$ and $\hat{\mathbf{D}}_{k+1}$ at the same time as computing the KF gain \mathbf{K}_k used to update the states $\hat{\mathbf{x}}_{k+1}$ in an iterative way with respect to the states. The operation is done sequentially for each state, therefore a new KF gain \mathbf{K}_k and an updated innovation $\delta \tilde{\mathbf{z}}_k$ are computed at each iteration. The measurement matrix \mathbf{H}_k are linearized with respect to the new updated states $\hat{\mathbf{x}}_{k+1}$. The pseudo-code of the Thornton propagation and Bierman update are given in Appendix D.3.10. This factorization reduces the dynamic ranges of the variables, leading to more homogeneous scales in the computations and preserves the symmetry of the covariance matrix $\mathbf{P} = \mathbf{UDU}$ which is recomputed at the end of the estimation or for debugging.

For a filter with 47 states (as is the case for the default VDM system with the minimum of auxiliary states (Eq. 3.69)) the standard time update of the covariance matrix as $\Phi \mathbf{P} \Phi^t$ requires 203228 addition operations and 207646 multiplications. For the Thornton implementations, it requires 143585 summations and 146640 multiplications, representing approximately 30% fewer mathematical operations. There is the possibility to reduce the number of operations even further by separating the original states vector into “states” and “parameters”. When the “states” have their process model, the “parameters” ($\mathbf{x}_e, \mathbf{x}_w$) can be generally modeled as first-order Gauss-Markov processed. The characteristics of these processes considerably reduce the complexity of their covariance update, reducing the total number of operations. This method is proposed in [64]. However, it is not explored here as the real-time computational load is acceptable for the current implementation Sec. 9.2.2, but it could be an exciting extension to add to a future version of the system.

4.2.2 Partial Updates

The second method employs the Schmidt-Kalman [66] or so-called “consider” filter. The implementation is particularly advantageous when the estimated states are composed largely of almost constant values such as sensor biases, or in the case of VDM, the aerodynamic coefficients. The partial Schmidt-Kalman filter is principally defined as a weighted mean between the updated and predicted covariance of the “consider” states with a weight factor

$\gamma \in [0 - 1]$ as

$$\hat{\mathbf{P}}_{yy}^c = \gamma^2 \tilde{\mathbf{P}}_{yy} + (1 - \gamma^2) \hat{\mathbf{P}}_{yy} \quad (4.8)$$

A practical issue with the UDU decomposition is that this factorization is not distributive on additions. In a naive implementation, one can reconstruct the covariance matrix, apply the partial reset, and factorize it again. However, it would lead to a large increase in computations and partially defeat the purpose, which is to maintain the covariance in a numerically stable form. [64] presents a method to apply to a classic Schmidt-Kalman in the case of UDU factorized filters. This development can be adapted to carry out the aforementioned weighted mean. Indeed, by using the symmetry of covariance matrices, Eq. 4.8 can be reformulated as:

$$\begin{aligned} \hat{\mathbf{P}}_{yy}^c &= \hat{\mathbf{P}}_{yy} + (1 - \gamma^2) \mathbf{K} \mathbf{H} \tilde{\mathbf{P}}_{yy} \\ &= \hat{\mathbf{P}}_{yy} + (1 - \gamma^2) \mathbf{K} (\mathbf{H} \mathbf{P} \mathbf{H}^T + \mathbf{R}) \mathbf{K}^T. \end{aligned} \quad (4.9)$$

Defining $\mathbf{S} = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$, the Partial-Schmidt update can be defined as:

$$\hat{\mathbf{P}}^c = \hat{\mathbf{P}} + (1 - \gamma^2) \mathbf{S} \mathbf{K} (\mathbf{H} \mathbf{P} \mathbf{H}^T + \mathbf{R}) \mathbf{K}^T \mathbf{S}^T \quad (4.10)$$

Since the Bierman-Thornton update step requires the scalar processing of the measurements, the dimensions of \mathbf{K} and $(\mathbf{H} \mathbf{P} \mathbf{H}^T + \mathbf{R}) = \mathbf{W}$ are $n \times 1$ and 1×1 , respectively. Therefore one obtains the expression:

$$\begin{aligned} \hat{\mathbf{P}}^c &= \hat{\mathbf{P}} + (1 - \gamma^2) (\mathbf{S} \mathbf{K}) \mathbf{W} (\mathbf{S} \mathbf{K})^T \\ &= \hat{\mathbf{P}} + (1 - \gamma^2) \mathbf{v} \mathbf{W} \mathbf{v}^T \end{aligned} \quad (4.11)$$

The updated equation reduces to a rank one, which in the case of UDU factorization, can be applied by using the Agee-Turner method [64] for covariance propagation. In other words, an update of rank one is applied directly to the decomposed matrices without recomposing them to perform the partial reset. The factorization adds n^2 additions, $n^2 + 3n + 2$ multiplications, and $n - 1$ divisions for each scalar measurement. However, the covariance updates require, in general, much fewer operations than its time update; therefore, these extra operations have no real impact if the time required to perform these steps is inspected (Sec. 9.2.2).

Such a mixed implementation benefits several flight phases, i.e. during the filter initialization and potential GNSS signal outages. Both phases are described in the following subsections. A possibility of using the partial-Schmidt filter is within the aerodynamic coefficients calibration phase (Sec.5.2).

For better clarity, the combination of the UDU factorization with the partial-Schmidt filter implementation will be abbreviated as “partial-update” in the sequels.

4.2.3 Initialization

At the beginning of the filtering process, the Kalman filter passes through a transient phase. This is partly caused by: i) the level of uncertainty normally expressed by a quasi diagonal matrix \mathbf{P}_0 , as most of the correlations between the states are unknown, ii) the fact that most of the initial states are unknown (set to zero) and iii) (some) have large uncertainties relative to the measurement precision. During this phase, there is a high probability that some states converge to a local minimum and remain either incorrectly estimated and/or correlated to other states until the conditions on their observability improve (e.g., due to new dynamics or additional measurements). This, in turn, limits the navigation quality. The “initialization” employing a partial-Schmidt-Kalman filter is proposed to remedy this problem. To allow the states to evolve more smoothly, a growing $\gamma(t)$, from 0 to 1 during an initialization period T_{ini} is defined as

$$\gamma(t) = \frac{1}{T_{ini}}(t - t_0) \quad (4.12)$$

where t_0 is when the initialization period starts, and t is the current estimation time. Such “initialization” is implemented for the VDM parameters \mathbf{x}_p (that are usually pre-calibrated), the wind \mathbf{x}_w and the sensor error states \mathbf{x}_e . These are referenced as “consider” states later on. The duration of T_{ini} is experimentally investigated in Sec. 9.3.2 together with other benefits.

4.2.4 GNSS Outage

During a GNSS outage, the absence of position and velocity observations from a GNSS receiver prevents the direct update of navigation states that are related to other auxiliary states via an observation model, resulting in a constant increase in their uncertainties. On the other hand, the IMU measurements that are still present, update all navigation states \mathbf{x}_n at high frequency. Barometer observations are further present to restrain the vertical channel. Thus, after some time, the elevated variance in horizontal position allows for strong corrections to be applied to the navigation states when only influenced by IMU updates, possibly leading to erratic jumps in the estimation trajectory. Such corrections are even more exaggerated if the IMU biases are poorly estimated. To counter this effect, the partial update is performed by setting the states related to aerodynamic coefficients (\mathbf{x}_p) as “consider” states. These are indeed observable only during the presence of GNSS positioning. The benefits of the presented combination of filtering strategies are presented in Sec. 10.2 for the GNSS outage phases.

More details on the two methods and their implementations are described in [64, 67, 68]. The partial update is implemented in the MATLAB and C++ framework.

State rescaling and factorization improve the numerical stability of the VDM-based navigation filter during operation. In all cases, the large state space challenge inevitably the stability of the filter. A reduction methodology is proposed in the next section to reduce the state space using an unconventional approach by lumping the estimation of highly correlated states.

4.3 State Space Reduction

The state corrections \mathbf{x}_p to aerodynamic coefficients (VDM parameters) are (possibly highly) correlated. When the UAV performs highly dynamic maneuvers, as will be presented in Sec. 8.1, the global observability of the system is increased, where the consequences of which are twofold; on the one hand, correlations between the VDM parameters are accentuated, and on the other, the Persistence of Excitement (PE) [69] allows for the de-correlation of other states.

First, such correlations should be considered with confidence during filter initialization before enabling them to be tuned, if required, while flying the UAV. Second, there is a need to combine sets of highly correlated states to reduce the system to a size that is feasible for microcomputer implementation, as the one used in this research and presented in Sec. 6.2. The time required for the KF propagation and updating steps to be computed with the embedded computer depends on different system state sizes presented in Sec. 9.2.2.

Rather than using well-known approaches of state reduction via suboptimal filtering [58], some correlated parameters can be expressed as combinations. This reduces the number of states used during estimation while maintaining them in the full model for application via the re-projected and previously determined correlations. The goal is to maintain as much of the system accuracy as possible while optimizing the system for real-time implementation. It does this by suppressing some states to reduce the computational load required to update the state vector (the number of multiplications increases with the cube of the state-vector size).

The candidate pairs of coefficients that can be put together are identified by analyzing the evolution of the covariance matrix P in time. If the correlation between two states is high and does not vary temporally, the updated values are expected to change similarly when refined during the state estimation. Linear regression can be used to express one coefficient as a function of the other using $C_j = s_{ij} * C_i + o_{ij}$ where the coefficient C_j is expressed as a scale s_{ij} function of C_i , plus an offset o_{ij} . The aerodynamic equations can be modified accordingly, reducing the total VDM parameters in the augmented state \mathbf{X}_p by the number of candidates found. The choice of the aerodynamic coefficients to be paired and their linear expressions are given in Sec. 5.3.1. The influence of the new aerodynamic model on the VDM navigation performance in nominal flight conditions and under GNSS outage is explored in Sec. 10.4.

Summary

The following enhancements within the estimation have been proposed in this chapter: a rescaling of the states, a factorization technique combined with partial updates, and a reduction of the aerodynamic model. These propositions have addressed: (i) the numerical instability of the estimator; (ii) the oscillations of state variables during the (in-air) initialization; (iii) the prospect of maintaining the same filtering methodology during nominal and autonomous operations. Yet, with these improvements, the correct VDM-based navigation can only work if a set of (correct) aerodynamic coefficients is available. The next chapter focuses on estimating these coefficients directly from the flight data, thereby avoiding the need for additional tools such as expensive and time-consuming wind tunnel experiments, CFD.

5 Aerodynamic Coefficient Estimation

Overview

The performance of VDM-based navigation is largely dependent on the accuracy of the model-dependant aerodynamic coefficients characterizing the behavior of the platform. These coefficients can be either (i) known or determined in advance, for example, from CFD analysis or via experiments in wind tunnel testing; (ii) approximated from a model describing the platform of a similar shape that needs to be re-calibrated; (iii) completely unknown and therefore they need to be first approximated for estimation/linearization and later refined. This chapter covers the aspect of aerodynamic coefficient estimation without prior approximation and subsequent calibration (e.g., non-linear estimation). It describes a multistep methodology using the recorded sensor data from calibration flights. The proposed procedure avoids the expensive and time-consuming experimental setup (e.g., in a wind tunnel) and/or aerodynamic modeling software. The proposed methodology allows estimating a set of aerodynamic coefficients independently of the platform type. The methodology first proposes estimating an approximate set of coefficients using the recorded sensor data and trajectory reference from a calibration flight. The procedure is designed into three linear estimators to obtain first the wind, followed by the moment and the force coefficients estimation. An observability criterion employing the Observability Grammian matrix is used to limit the “artificial” decrease of the covariance matrix caused by the relatively large number of observations - which does not necessarily increase the system observability. This guarantees that the parameters for the three estimators are only estimated when the system experiences sufficient excitation. Further or alternative refinements are proposed within VDM-based state space estimation, employing either precise attitude obtained via photogrammetry or precise IMU. The subsequent use of optimal smoother permits decorrelating some coefficients while improving their estimation (i.e., confidence levels). The resulting covariance matrix can be used as initial covariance for navigation, possibly after scaling that allows for small adaptations between flights. Some of the presented aspects are taken from [14, 28]

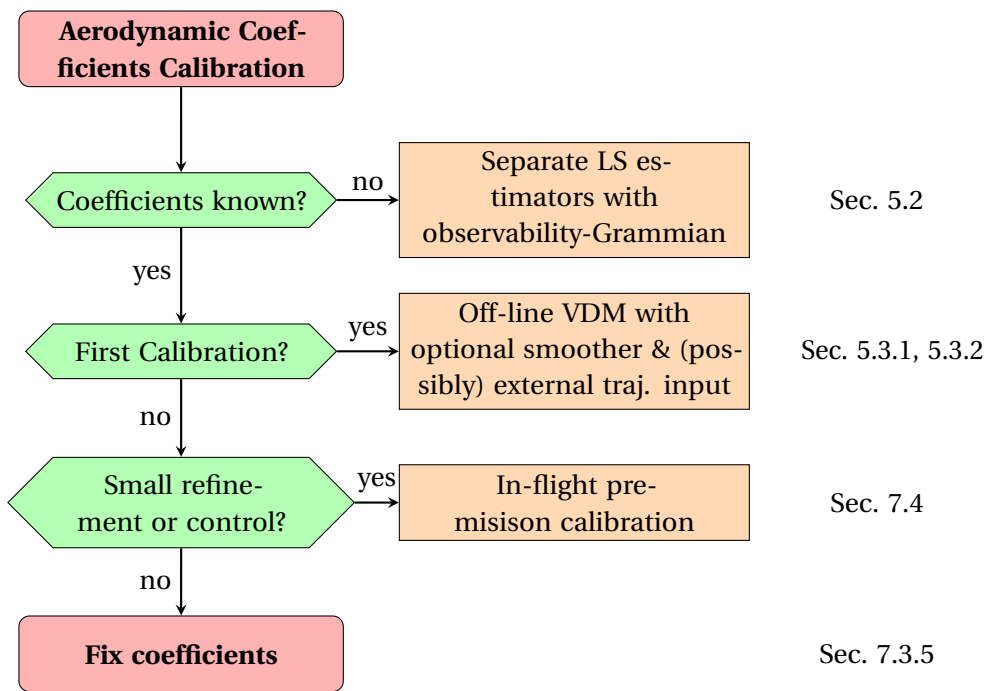


Figure 5.1: Aerodynamic coefficient calibration flowchart

5.1 General Calibration Methodology

The flow chart presented in Fig. 5.1 explains the coefficient estimation/calibration procedure with the different techniques based on their a priori “levels” of knowledge of the coefficients. If the aerodynamic coefficients are unknown, separate least squares estimates are proposed using observability analysis to select suitable data (Sec. 5.2). With this method, a set of coefficients can be obtained for further refinement using state space enhancement. Indeed, calibration via state space augmentation can only be performed when the aerodynamic coefficients are known with some confidence. Otherwise, convergence is not guaranteed [18]. Monte Carlo simulations find the convergence threshold to be approximately within 30% of their correct values. The results of this analysis are presented in Sec. 8.1.6. Then, methods using state space augmentation are proposed: the first one uses ‘pose’ references from sufficiently precise INS/GNSS (Sec. 2.4); the second one employs precise attitude observation from photogrammetry (Sec.5.3.2). When the aerodynamic coefficients are calibrated for a platform with one of these methods, they can be further refined in real-time to cope with small changes in the physical geometry of the UAV between flights (Sec. 8.1).

5.2 Approximate Coefficients Determination

Knowledge of the aerodynamic coefficients of the platform is needed to derive the navigation states that are linearized in the EKF steps. If the linearized states are far from the true states, the filter can diverge [18], and the VDM-based navigation approach cannot be used. Therefore,

it is necessary to obtain an initial set of coefficients that is sufficiently close to reality. Special setups can be used to determine most of them as wind tunnel experiments with a robotic arm and load cells observing the applied forces and moments, CFD analysis when the 3D model of the platform is available, or the coefficients with a similar known shaped-aircraft can be adapted as an initial guess. The latter method is used in this investigation with the model-based navigation described in [51]. However, to generalize the VDM-based navigation method to any platform, a methodology is needed to determine a set of coefficients without additional tools.

The proposed procedure utilizes a reference trajectory with its post-processed INS/GNSS navigation solution with centimeter-level accuracy for the position, ~ 0.02 [m/s] for velocity and attitude below 0.1 [deg]. This is needed to correct for the aerodynamic coefficients with which the VDM process model is expected to produce a similar level of accuracy. Furthermore, the autopilot logs the flight control commands, and all geometric parameters (including the inertial tensor, where [70] proposes its addition in the state space to be estimated) are already determined or assumed to be known. Airspeed measurements and raw inertial data are also needed.

These data are then used in a cascade scheme of LS estimators to estimate the aerodynamic coefficients (VDM parameters). Not all epochs of the calibration trajectory are used to determine the parameters. Instead, an observability criterion based on the observation model Grammian selects the regression points. A partial-Schmidt update is employed to determine which of the available parameters should be updated. Wind velocities, moments, and forces are then estimated sequentially.

5.2.1 Estimation Separation

The starting point for identifying the aerodynamic coefficients comes from the equation for the accelerometer observation model given in Eq. 3.75 and provided again here as

$$\mathbf{z}^{fimu} = \mathbf{R}_b^{imu} \left(\mathbf{f}^b + \left(\dot{\boldsymbol{\Omega}}_{ib}^i + \boldsymbol{\Omega}_{ib}^i \boldsymbol{\Omega}_{ib}^i \right) \mathbf{r}_{b-acc}^b \right) + \mathbf{x}_e^{fimu} + \boldsymbol{\epsilon}_{acc}$$

where \mathbf{f}^b and $\dot{\boldsymbol{\Omega}}_{ib}^i$ are functions of the coefficients to be estimated. Then, by rotating the IMU measurement to be in the body frame (\mathbf{R}_b^{imu}) and ignoring the remaining stochastic/time-correlated error terms, the observation equation can be simplified as

$$\mathbf{z}^{fimu} + \mathbf{v}^{imu} = \mathbf{f}^b + \left(\dot{\boldsymbol{\Omega}}_{ib}^i + \boldsymbol{\Omega}_{ib}^i \boldsymbol{\Omega}_{ib}^i \right) \mathbf{r}_{b-imu}^b \quad \text{with } \mathbf{v} \sim N(0, \sigma_{acc}) \quad (5.1)$$

The force specification via VDM is again given by

$$\mathbf{f}^b = \left(\begin{bmatrix} F_T^b \\ 0 \\ 0 \end{bmatrix} + \mathbf{C}_w^b \begin{bmatrix} F_x^w \\ F_y^w \\ F_z^w \end{bmatrix} \right) \frac{1}{m}$$

with the force coefficients to be identified in F_T^b and F^w . The skew matrix is $\hat{\Omega}_{ib}^b = [\dot{\omega}_{ib}^i]_{\times}$, where $\dot{\omega}_{ib}^i$ is defined in (Eq. 3.57) as

$$\dot{\omega}_{ib}^b = \left(\mathbf{I}^b\right)^{-1} \left[\mathbf{M}^b - \Omega_{ib}^b \left(\mathbf{I}^b \omega_{ib}^b\right)\right] \quad (5.2)$$

and \mathbf{M}^b contains all moment terms. Furthermore, all forces and moments are directly proportional to the dynamic pressure \bar{q} or airspeed magnitude V . With the equation of airspeed-ground speed-wind as $\mathbf{V} = \mathbf{v} - \mathbf{w}$, it is possible to determine wind velocity \mathbf{w} knowing the airspeed (e.g. via Pitot tube measurements) and the ground velocity \mathbf{v} of the UAV. Therefore, knowing the wind velocity \mathbf{w} and by numerical derivation of the gyro data, the moment coefficients can be identified using Eq. 5.2. When the wind and the moments are known, the force coefficients can be determined using Eq. 5.1. The three steps (wind, moments, and forces estimation) are described hereafter.

The substitution of navigation states, control commands and geometric parameters in Eq. 5.1 results in a nonlinear observation model involving aerodynamic model parameters and wind ($21 + \{3 \times \text{number of observations}\}$ unknowns). Such a substitution does not have a decoupling effect. Moreover, a similar substitution in Eq. 5.2 results in a nonlinear model involving aerodynamic moment-parameters and wind ($11 + \{3 \times \text{number of observations}\}$ unknowns). For such a substitution to work, it is assumed that gyroscope measurements can be reasonably differentiated. Though this substitution has no decoupling effect, it gives credible evidence that if the wind is estimated a priori (thanks to [52]), the model Eq. 5.2 is linear in aerodynamic moment-parameters. Consequently, if both wind and aerodynamic moment-parameters are known, then the model 5.1 is linear in aerodynamic force-parameters. This results in three linear and decoupled estimators to effectuate aerodynamic calibration of UAV using inertial data and Pitot's tube.

5.2.2 Generic Model Structure

The general calibration procedure is depicted in Fig. 5.2 and the process model for each estimator is of the following form:

$$\mathbf{x}_{k+1} = \begin{bmatrix} \mathbf{x}_{k+1}^i \\ \mathbf{x}_{k+1}^c \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k^i \\ \mathbf{x}_k^c \end{bmatrix} = \mathbf{x}_k \quad (5.3)$$

where the observation model is in the form of a recursive least squares equation as presented in 2.1.2. The previously reviewed works of [52, 71] have shown that observability of aerodynamic model-parameters and wind is dependent on the trajectory, and hence they are only mildly observable. Studies from [68] have shown that estimating nuisance states in a traditional setting can affect filter accuracy and consistency. A Partial-update Schmidt Kalman Filter (PSKF) [68] is used, effectively estimating constant and time-varying nuisance states. PSKF facilitates a generalized estimation approach in which only a portion of the traditional full filter update can be applied to selected states. According to observability conditions, the

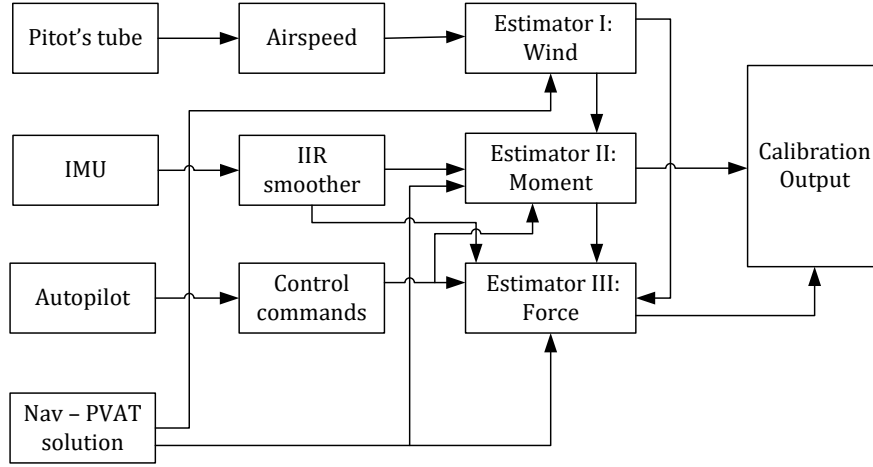


Figure 5.2: Calibration procedure as a sequence of three decoupled estimators

nuisance terms can sometimes be treated as full filter states and updated. Other times, they can be *considered* and not updated. Such a framework has been proven to be unbiased and consistent [68].

5.2.3 Estimator I -Wind

As highlighted in 5.4, wind serves as one of the forcing inputs of the differential equation modelling the VDM state dynamics and is indispensable for linearizing Eq. 5.2; therefore its estimation is important for a reliable navigation solution. The UAV-based wind estimation methodology from [52] is adapted in discrete-time. It has been theoretically and practically validated on three different platforms. The methodology makes use of Pitot's tube as an external measurement. Let $\mathbf{x}_w \in \mathbb{R}^3$ be the wind velocity in the body-frame, and $\gamma \in \mathbb{R}^+$ be the scale factor of Pitot's tube, then the process model is of the following form:

$$\begin{bmatrix} \mathbf{x}_{w_{k+1}} \\ \gamma_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{w_k} \\ \gamma_k \end{bmatrix} + \mathbf{w}_k \quad (5.4)$$

The observation model is of the following form:

$$z_k = \mathbf{d}^T \mathbf{R}_l^b \mathbf{x}_w + z_{a_k} \gamma_k + v_k, \quad (5.5)$$

where $\mathbf{d} = [1 \ 0 \ 0]$, z_{a_k} is the airspeed measured by Pitot's tube, \mathbf{R}_l^b is the rotation matrix from local frame to body frame (Eq. 3.43) and z_k is the longitudinal velocity of the aircraft, obtained as a result of sensor fusion (INS/GNSS). The incorporation of Eq. (5.4) and (5.5) as the process and observation model of KF, respectively, yields wind velocity.

To ensure observability, the fixed-wing drone must keep changing its attitude, as the related states are estimated only when they are observable. This estimation is carried out using a

Chapter 5. Aerodynamic Coefficient Estimation

PSKF, contrary to the standard KF proposed in [52] where the reported results showed that the planar wind is estimated with a lower uncertainty compared to vertical wind.

Moreover, this methodology assumes that the wind will vary slowly over time. These variations can be estimated because the Kalman gain effectively defines a cutoff frequency. Wind velocity frequencies lower than this cut-off frequency are captured, whereas the others are filtered.

5.2.4 Estimator II - Moments

When the wind-related entities are substituted and Eq. (5.2) is rearranged, the following observation models are obtained:

$$\mathbf{z}_m = \mathbf{H}_m \mathbf{C}_m \text{ with} \quad (5.6)$$

$$\mathbf{H}_m = \mathbf{D}_m \begin{bmatrix} \delta_a & \beta & \tilde{\omega}_x & \tilde{\omega}_z & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \delta_e & \tilde{\omega}_y & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_r & \tilde{\omega}_z & \beta \end{bmatrix}$$

$$\mathbf{D}_m = \begin{bmatrix} b & 0 & 0 \\ 0 & \bar{c} & 0 \\ 0 & 0 & b \end{bmatrix}, \mathbf{C}_m = \begin{bmatrix} C_{M_x a} & C_{M_x \beta} & C_{M_x \tilde{\omega}_x} & C_{M_z \tilde{\omega}_z} & C_{M_y 1} & C_{M_y e} & \dots \\ \dots & C_{M_y \tilde{\omega}_y} & C_{M_x \alpha} & C_{M_z \delta_r} & C_{M_z \tilde{\omega}_z} & C_{M_z \beta} & \dots \end{bmatrix}^T$$

and

$$\mathbf{z}_m = \frac{1}{\bar{q}S} \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}_b + \boldsymbol{\Omega}_{ib}^b \left(\mathbf{I}^b \boldsymbol{\omega}_{ib}^b \right) \right) \quad (5.7)$$

where $\dot{\boldsymbol{\omega}}_b$ is obtained by employing a high-order (e.g. eight, in our case) differentiator on a sequence of gyro observations with deterministic errors removed (during pre-calibration and INS/GNSS integration), and α , β and V (used in the term \bar{q}) are taken from Estimator I.

The process model as depicted in Eq. 5.3, however, results in an 11x11 process-noise covariance matrix, which is extremely cumbersome to tune in a practical setting. Additionally, as the model parameters associated with aerodynamic moments remain virtually constant [71], the complexity can be reduced by setting this process noise to zero. This effectively turns the KF to an RLS moment-parameter estimator. It should be noted that white noise can also be used. However, choosing its form and strength is not straightforward and requires adaptation to the stability of the meteorological conditions. In the end, a partial update framework addresses the observability concerns.

5.2.5 Estimator III - Forces

By substituting wind and moment parameters in Eq. 5.1 and subsequently rearranging the model, the following observation model is obtained

$$\mathbf{z}'_f = \mathbf{H}_f \mathbf{C}_f \text{ with, } \mathbf{H}_f = \frac{1}{m_0} \left[\mathbf{A}_{33} \quad (\mathbf{R}_b^w)^T \mathbf{A}_{37} \right] \quad (5.8)$$

where m_0 is the mass of UAV,

$$\mathbf{A}_{33} = \rho D^4 \begin{bmatrix} n^2 & \frac{n^2 V}{D\pi} & \left(\frac{V}{D\pi}\right)^2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{A}_{37} = \bar{q} S \begin{bmatrix} 1 & \alpha & \alpha^2 & \beta^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \alpha \end{bmatrix}$$

and

$$\mathbf{C}_f = [C_{F_T1} \ C_{F_T2} \ C_{F_T3} \ C_{F_x1} \ C_{F_x\alpha} \ C_{F_x\alpha2} \ C_{F_x\beta2} \ C_{F_y1} \ C_{F_z1} \ C_{F_z\alpha}]^T$$

with the observation model

$$\mathbf{z}'_f = \mathbf{f}^b - \boldsymbol{\Omega}_{ib}^i \boldsymbol{\Omega}_{ib}^i \mathbf{r}_{b-acc}^b - \left[(\mathbf{I}^b)^{-1} \left(\mathbf{M}^b - \boldsymbol{\Omega}_{ib}^b \left(\mathbf{I}^b \boldsymbol{\omega}_{ib}^b \right) \right) \right] \times \mathbf{r}_{b-acc}^b \quad (5.9)$$

where \mathbf{f}^b and $\boldsymbol{\omega}_{ib}^b$ are the readings of the accelerometer and gyroscope, respectively, $\mathbf{M}^b = [M_x^b \ M_y^b \ M_z^b]^T$. The other terms were defined elsewhere. The process model is the same as in Eq. (5.3). Following the same reasoning, the process noise is chosen to be zero, resulting in an RLS force-parameter estimator based on a PSKF framework.

5.2.6 Observability and Uniqueness

A heuristic is presented to segregate the system state of these estimators into nuisanced and observable states. It is inspired by [52], where the observability of the wind is evaluated only at the end of the trajectory by computing the Observability Grammian and finding it to be of full rank. This metric asserts that the system is observable at the end; however, it does not convey any information regarding what and when states are observable (or not). The following empirical approach evaluates these questions.

For the chosen generic model structure, applicable to all three estimators (wind, moments, and forces), with $\mathbf{F} = \mathbf{I}$, the observability Grammian can be computed after each observation by the following recursive summation of the normal equation:

$$\mathbf{W}_0(k) = \mathbf{W}_0(k-1) + \mathbf{H}_k^T \mathbf{H}_k \quad (5.10)$$

Let $\mathbf{V}_k = [\mathbf{v}_1^k \ \cdots \ \mathbf{v}_n^k]$ and $\Lambda_k = \text{diag}[\lambda_1^k, \dots, \lambda_n^k]$ such that $\mathbf{W}_0(k) \mathbf{V}_k = \mathbf{V}_k \Lambda_k$. $(\mathbf{v}_j^k, \lambda_j^k)$ denotes the j^{th} eigen-vector/eigenvalue pair of $\mathbf{W}_0(k)$. These eigenvalues are sorted in ascending order (1 denoting the smallest value and n the largest). They are computed using the

Chapter 5. Aerodynamic Coefficient Estimation

MATLAB function eig. Let $k = K$ denote the last observation, a closeness matrix is defined:

$$\Gamma_q = \mathbf{V}_q^T \mathbf{V}_K \quad \forall q \in \{1, 2, \dots, K\} = \begin{bmatrix} (\mathbf{v}_1^q)^T \mathbf{v}_1^K & (\mathbf{v}_1^q)^T \mathbf{v}_2^K & \dots & (\mathbf{v}_1^q)^T \mathbf{v}_n^K \\ (\mathbf{v}_2^q)^T \mathbf{v}_1^K & (\mathbf{v}_2^q)^T \mathbf{v}_2^K & \dots & (\mathbf{v}_2^q)^T \mathbf{v}_n^K \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{v}_n^q)^T \mathbf{v}_1^K & (\mathbf{v}_n^q)^T \mathbf{v}_2^K & \dots & (\mathbf{v}_n^q)^T \mathbf{v}_n^K \end{bmatrix} \quad (5.11)$$

$$(5.12)$$

Note that Γ_K is an identity matrix. Moreover, each element of Γ_q is a scalar product between the eigenvectors of $\mathbf{W}_0(q)$ and $\mathbf{W}_0(K)$. Then, Γ_q is written in the following form :

$$\Gamma_q = \begin{bmatrix} \mathbf{g}_q^1 & \mathbf{g}_q^2 & \dots & \mathbf{g}_q^n \end{bmatrix} \quad (5.13)$$

Subsequently, the maximum value along each column vector is computed as $\mathbf{g}_q^j \quad \forall j \in \{1, 2, \dots, n\}$

$$\max(\Gamma_q) = \begin{bmatrix} \max(|\mathbf{g}_q^1|) & \max(|\mathbf{g}_q^2|) & \dots & \max(|\mathbf{g}_q^n|) \end{bmatrix} \quad (5.14)$$

where $|\mathbf{g}_q^j|$ denotes element-wise absolute value of \mathbf{g}_q^j .

In other words, the a^{th} eigenvector of $\mathbf{W}_0(q)$ is closest to the first eigenvector of $\mathbf{W}_0(K)$ and so on. In this way, it is numerically possible to approximately associate the eigenvalues of observability-Grammian at each epoch with a chosen set of basis, which are the eigenvectors of $\mathbf{W}_0(K)$.

At the end of this step, n time series (of length K) of eigenvalues of \mathbf{W}_0 are obtained, and each is characterized by one of the eigenvectors of $\mathbf{W}_0(K)$. Let $\underline{\lambda}_j(k)$ be such a time series. As the system satisfies the conditions of PE for uniform observability, there exists $\epsilon > 0$ such that $\mathbf{W}_0(K) > \epsilon \mathbf{I}$ [72]. This means that for some sufficiently large K , $\mathbf{W}_0(K)$ is positive definite. Therefore, eigenvalues of observability Grammian tend to grow with each observation subject to excitation/dynamics, and this behavior is presented for the wind estimator in Sec. 8.2.

To ascertain the uniqueness of information brought in by each observation, the difference between adjacent elements of each time series is computed, which gives a feeling of the observability evolution in time

$$\nabla \underline{\lambda}_j(k) = \underline{\lambda}_j(k) - \underline{\lambda}_j(k-1) \quad (5.15)$$

If $\nabla \underline{\lambda}_j(k) > \text{threshold}$, then new information is available, and some states are observable. To use this heuristic for the generic model structure, it is imperative to first change the basis of state space to the basis created with the eigenvectors of $\mathbf{W}_0(K)$. This leads to the update of the states corresponding to a sufficiently large $\nabla \underline{\lambda}_j(k)$ and the other is considered.

For any of the three estimators, a general procedure is summarized in the form of an algorithm:

1. **Build observability Grammian.** For the epoch $k \in \{1, 2, \dots, K\}$
 - (a) Compute \mathbf{H}_k and $\mathbf{W}_0(k)$
 - (b) Compute the eigenvalues (Λ_k) and eigenvectors (\mathbf{V}_k) of $\mathbf{W}_0(k)$
2. **Form the new basis.** At the last epoch K
 - (a) With $\mathbf{W}_0(K)$, find the basis \mathbf{V}_K
 - (b) Compute the state-transformation matrix: $\mathbf{T} = \mathbf{V}_K^T$
3. **Compute the evolution in observability.** For epoch $k \in \{1, 2, \dots, K\}$
 - (a) Compute $\Gamma_k = \mathbf{V}_k^T \mathbf{V}_K$
 - (b) Compute $\max(\Gamma_k)$
 - (c) Re-arrange the eigenvalues into n time-series $\underline{\lambda}_j(k)$ for $j \in \{1, 2, \dots, n\}$
 - (d) Compute the difference $\nabla \underline{\lambda}_j(k)$
4. **Initialize the state \mathbf{x} estimator.** For $k \in \{1\}$
 - (a) Transform the initial states, initial covariance and process noise (if non-zero).
 $\mathbf{x}' \leftarrow \mathbf{T}\mathbf{x}, \mathbf{P}' \leftarrow \mathbf{T}\mathbf{P}\mathbf{T}^T, \mathbf{Q}' \leftarrow \mathbf{T}\mathbf{Q}\mathbf{T}^T$
5. **Estimate the "transformed" states \mathbf{x}' .** For $k \in \{2, \dots, K\}$
 - (a) If $\nabla \underline{\lambda}_j(k) > \text{threshold} \forall j \in \{1, 2, \dots, n\}$, update the state else *consider*. This is where partial update is applied.
6. **Transform back to original basis.** For $k \in \{1, \dots, K\}$
 - (a) Invert the basis, to obtain the estimated states $\mathbf{x} \leftarrow \mathbf{T}^{-1}\mathbf{x}'$ and covariance $\mathbf{P} \leftarrow \mathbf{T}^T\mathbf{P}'\mathbf{T}$ in their original set of basis

The partial update algorithm detailed above to effectuate aerodynamic calibration is carried out sequentially for the three estimators in the order depicted in Fig 5.3: A similar procedure

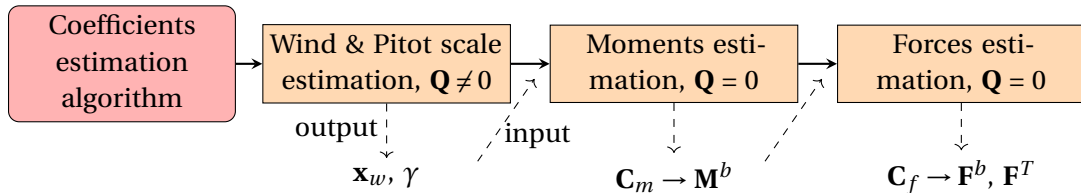


Figure 5.3: Coefficient estimation algorithm flowchart with three estimators for the wind (plus Pitot scale factor), moments, and forces

can be followed for the delta wing platform to effectuate its calibration. The coefficients obtained are then considered sufficiently close for linearization in a compound estimation

methodology for further fine-tuning as described in Sec. 5.3.1 and 5.3.2. This methodology will be referred to as Wind Moments Forces (WMF) in the following sections. When an approximate set of coefficients is available, it can be further refined using the techniques proposed in the next sections.

5.3 Coefficients Refinement

Two methods for coefficient refinement are proposed here using post-processed methods and precise observation inputs. It is important to note that the following techniques do not work due to the augmented-states estimation method. The filter tends to diverge (Sec. 8.1.6) if the initial coefficients to be refined (\mathbf{x}_p) are not first approximated with a coarse method such as WMF or adapted from a similar aircraft model.

5.3.1 Calibration with External 'pose'

If a set of coefficients are approximately known, either (i) after adaptation from a similar platform or (ii) using an estimation method as proposed in Sec. 5.2 with a low-grade IMU, the aerodynamic coefficients can be further tuned using the augmented system space with the possibility of using of a 'pose' sensor based on external/reference of INS/GNSS giving direct updates to the navigation states \mathbf{x}_n . The observation model for this sensor is given by

$$\mathbf{z}_{pose} + \mathbf{v}_{pose} = \begin{bmatrix} r_{nav} \\ v_{pose} \\ \Psi_{pose} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_e^l + D^{-1} \mathbf{R}_b^{pose} \mathbf{r}_{b-pose}^b \\ \mathbf{v}_{el}^l + \left(\mathbf{R}_l^b \boldsymbol{\Omega}_{ib}^b - \boldsymbol{\Omega}_{ie}^l \mathbf{R}_l^b \right) \mathbf{r}_{b-pose}^b \\ \mathbf{R}_b^{pose} \mathbf{R}_l^b \end{bmatrix} \quad (5.16)$$

where $\mathbf{R}_b^{pose} = \mathbf{R}_b^{imu}$ is a potential boresight and $\mathbf{r}_{b-pose}^b = \mathbf{r}_{b-nav}^b = \mathbf{r}_{b-imu}^b$ is the lever-arm between the b -frame origin and the 'pose' sensor or the INS system, respectively. All other terms are defined in Sec. 3.5.3.

The 'pose' sensor injects reference post-processed navigation observations, including position, velocity, and attitude observations, updating the states at different epochs. The frequency at which these corrections are available is normally forced to be equal to the IMU observations being, in general, the sensor generating observations at the highest frequency. In addition, in post-processing, the state estimation and in particular, the VDM parameters can be significantly improved by filtering in a backward direction with respect to time and then combining the results of the forward and backward solution by an optimal smoother as presented in Sec. 2.4. The optimal smoother improves the estimation of the parameters, among them the aerodynamic coefficients. For better results and fewer fluctuations, the optimal smoother is implemented to combine the estimation states of the backward filter and the 2nd-forward stage filter. The fixed-step smoother combines the forward and backward solutions in the

least-square sense:

$$\mathbf{x}_k^N = (\mathbf{x}_b)_k + \left[(\mathbf{P}_b^{-1})_k + (\mathbf{P}_f^{-1})_k \right]^{-1} (\mathbf{P}_f^{-1})_k [(\mathbf{x}_f)_k - (\mathbf{x}_b)_k] \quad (5.17)$$

where $(\mathbf{P}_.)_k$ and $(\mathbf{x}_.)_k$ are the covariance matrix and the state vector at time epoch k , respectively. The indexes b and f show the results of the second stage backward and forward filters.

The application of such a method is presented in Sec. 8.3.2 where a set of coefficients is obtained for *TP2* and in Appendix A.6 for *eBeeX*.

5.3.2 Attitude References from Photogrammetry

The method previously presented using state space augmentation has the advantage of not requiring any additional sensors. Investigations of this approach address certain challenges in separating estimates of aerodynamic coefficients related to specific forces or moments [47]. Moment estimation, in particular, proved to be especially difficult and required multiple iterations [73] when the aerodynamic coefficients are adapted from a platform of similar shape. The quality of the previous method is proportional to the IMU quality and cm-level positioning. This may not always be available, which motivates the use of precise alternative observations of absolute attitude, which can be used in an off-line coefficients calibration.

Photogrammetry represents one of the best options available to obtain such information at 0.01 deg-level or better accuracy at an extra weight of ~ 0.2 kg for a relatively good camera [74] and associated Bundle Adjustment (BA). In most situations, the camera attitude determination when an image is taken via adjustment surpasses the accuracy of a small inertial system. The camera-derived attitude can be used as an additional observation for calibration:

$$\mathbf{Z}_c + \mathbf{v}_c = \mathbf{R}_c^b \mathbf{R}_l^c(t) \quad (5.18)$$

with \mathbf{R}_l^b the rotation matrix representing the rotation from the local l -frame to the camera c -frame and \mathbf{R}_c^b the boresight between the c -frame to the body b -frame. The latter is usually derived concurrently with \mathbf{Z}_c within the BA. The camera is rigidly attached to the drone, and the attitude solution is translation-independent. Therefore, a lever arm between the camera and body frame origin is not needed here, but possibly in the BA using INS/GNSS poses. As BA determines the camera rotation for some fixed-ground system, the rotation from the position-dependent local-level frame l to the c -frame is obtained via successive transformations.

The procedure to obtain the attitude reference of an IMU from camera data, camera External Orientation (EO), IMU errors calibration, and precise INS/GNSS integration with photogrammetry treatments, is schematically depicted in Fig. 5.4. The superscript $\tilde{\cdot}$ and $\hat{\cdot}$ are used to specify estimated and updated results, respectively, and the procedure is described in more

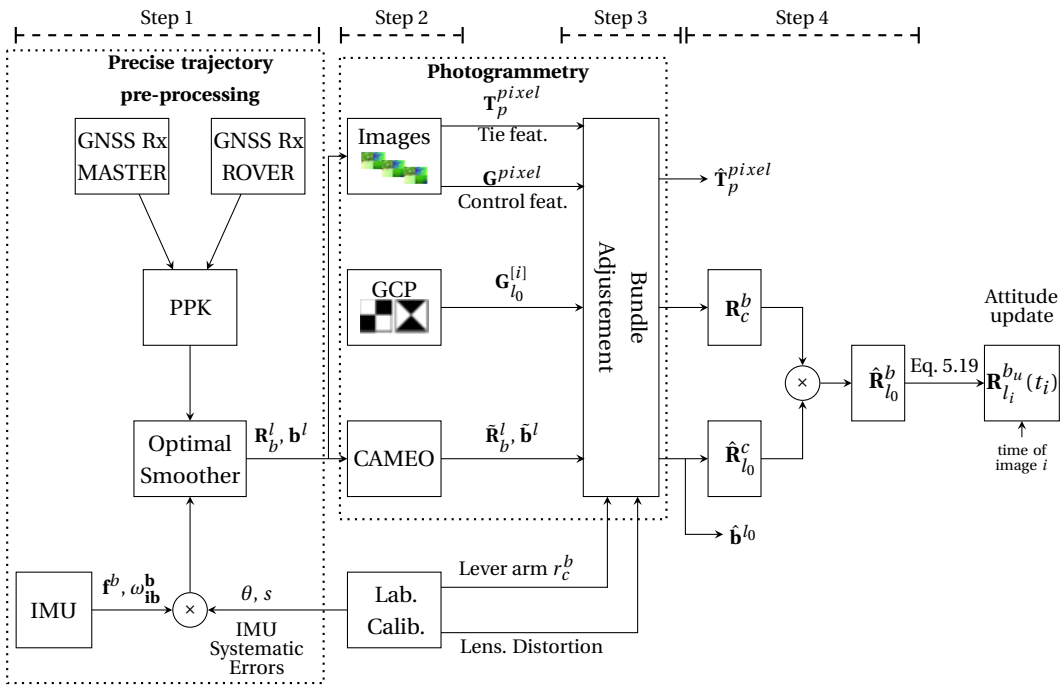


Figure 5.4: Steps required to obtain attitude updates in the body-frame using the precise pre-processing trajectory treatment and photogrammetry

detail in the following.

Methodology

Four main steps are necessary to produce precise attitude observations from overlapping images acquired by the UAV, which can later be used to improve the calibration of the UAV aerodynamic coefficients:

1. First, the position and attitude (pose) of each image are approximately obtained from the camera exposure times using onboard sensors and INS/GNSS integrated trajectory with (if possible) cm-level (RTK/PPK) relative GNSS positioning and sufficient flight geometry (overlap, altitude variation).
2. Second, the photogrammetry suite software analyses each image separately to detect key points, the centroids of local image features. Then, with the help of approximated pose and camera interior orientation values, the software determines correspondences (matches) between some of these points on several images.
3. Third, the previously matched image observations are confronted with approximate observations of camera poses in a bundle adjustment (BA). At this point, the observations of signalized points, called ground control points (Ground Control Point (GCP)s),

can be added (both in the object and in the image space). This additional information improves the simultaneous determination of camera poses and interior orientation, but is not necessarily indispensable if the geometry of the image configuration is strong. The aerial control during flight (via GNSS or INS/GNSS) is of sufficient accuracy [74].

4. The final step transforms the adjusted camera exterior orientation at these instances to the IMU or body frame to use the absolute attitude references in the estimator. The last step is subsequently detailed while its implementation on a particular flight is described in Sec. 8.3.1.

Transformation back to IMU-Body (l) frame

Within one flight, the area covered by the UAV corresponds to a small portion of the Earth's surface. Thus, a local tangent frame (l_0) with *ENU* axes orientation can be used (with negligible influences of Earth's curvature) as the mapping frame. Its origin, denoted as ENU_o , is set either at the center of the mapping zone or at its extremity to maintain positive coordinates. After running the photogrammetry process and its BA (with possibly INS/GNSS input), the adjusted camera attitude $R_{l_0}^{c(j)}$ is obtained for each photo $j \in [j = 1..J]$ and possibly also R_b^c .

To obtain IMU attitudes $R_{b_u(i)}^{l_i}$ from oriented photos $R_{l_0}^{c(j=i)}$, the following sequence of transformations must be performed:

$$R_{b_u}^{l_i} = R_e^{l_i}(\varphi_i, \lambda_i) \cdot R_{l_0}^e(\varphi_0, \lambda_0) \cdot T_{ENU}^{NED} \cdot \left(R_c^{b_u} \cdot R_{l_0}^{c(i)} \right)^T \quad (5.19)$$

The camera attitude is rotated from the mapping frame (here local-Cartesian system, ENU) back to the body-frame (local-level on a reference ellipsoid, NED and expressed with respect to IMU-frame b_u). This process is somewhat subtle and needs to be modified when the mapping-frame includes projection, and national reference frame [29].

The bore-sight between the camera and the IMU representing their orientation offset (R_c^b in Eq.5.19) is also estimated within the BA, possibly in a different flight using INS/GNSS derived orientation parameters [75].

The matrix T_{ENU}^{NED} in Eq. 5.19 defined as

$$T_{ENU}^{NED} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}^T \quad (5.20)$$

transforms the axis orientation between NED and ENU.

Chapter 5. Aerodynamic Coefficient Estimation

The matrix

$$R_{l_0}^e = \begin{bmatrix} -\sin(\varphi_0) \cdot \cos(\lambda_0) & -\sin(\lambda_0) & -\cos(\varphi_0) \cdot \cos(\lambda_0) \\ -\sin(\varphi_0) \cdot \sin(\lambda_0) & \cos(\lambda_0) & -\cos(\varphi_0) \cdot \sin(\lambda_0) \\ \cos(\varphi_0) & 0 & -\sin(\varphi_0) \end{bmatrix} \quad (5.21)$$

is a constant rotation from the mapping-local frame to the e -frame and

$$R_e^{NED_i} = R_{NED}^e(\varphi_i, \lambda_i)^T \quad (5.22)$$

where (φ_i, λ_i) represent the IMU position at the time the image is taken.

Sec. 6.2 details the experimental setup used to investigate the methodology, and Sec. 8.3.1 exposes the practical results when using attitude references as observations and summarizes the findings.

Summary

In this chapter, a methodology to estimate, calibrate and refine the platform-dependant aerodynamic coefficients has been proposed. First, a coarse set of coefficients was obtained via three separate estimators that did not require prior knowledge. In a second step, these coefficients can be calibrated offline, benefiting state space augmentation with precise attitude observations from photogrammetry, or high-quality reference trajectory from an IMU of higher quality. These coefficients can be further refined in flight to absorb the modification of the platform geometry between missions. To evaluate these coefficients, they will be used as initial values for the real-time VDM-based navigation system whose setup is detailed in the next chapter.

Real-time framework and setup **Part III**

6 Platforms and Experiments

Overview

Most of the current results regarding the proposed model-based navigation solution for small UAV are either derived from simulations, or from recorded flight data replayed in post-processing. Therefore, there is a need to demonstrate the feasibility of this new approach to autonomous navigation in a real-time experimental setup. Its design, implementation, and validation are one main research objectives of this thesis. Moreover, the real-time implementation should be portable between platforms of different shapes, and also adaptable to inertial data of varying quality. To test the feasibility of the navigation system for different platforms, conventional fixed-wing and delta-wing drones are used, weighing less than 3kg . Their main physical characteristics are presented in this chapter, while their respective aerodynamic models were introduced in Sec. 3.5. Alongside the platform aerodynamic model that serves as a main process model within the estimation scheme, sensor observations are needed in the estimation. In this work, IMU and GNSS sensors are used for navigation. In contrast, the camera and the Pitot tube are used for the determination/calibration of aerodynamic coefficients and the validation of wind estimation. Most of the sensors and devices are available on the market. These sensors are described with their respective, placement, interfaces, and main characteristics. The used payloads were developed internally to house the required sensors and fit the geometric specifications of the UAVs. Furthermore, to operate the drone, ground equipment is also needed: a GCS to plan and monitor the mission and a radio controller to switch rapidly between flight modes and to manually control the aircraft. This equipment is briefly described here, while more details about the GCS software are given in the next chapter. Finally, during this research, multiple flight campaigns are conducted to validate the new theoretical additions and the real-time aspects of the implementation (the latter are covered in the next chapter). The selected set of flights is described with the relevant mission information on the used platform and payload, the validation results obtained as well as related publications. Most of the descriptions are based on project reports and published material [28, 76].

6.1 Platforms

To implement and validate the different research objectives, three platforms are used, as shown in Fig. 6.1. The two fixed-wing platforms, called TOPOPlanes (*TP*), are depicted in Fig. 6.1(a) with the first (top) and the second version (bottom). The two *TPs* [74] are based



Figure 6.1: Fixed-wing platform with 5 control surfaces (left): the two versions of the TOPOplane (a) version-1 (top) and version-2 (bottom), and the *eBeeX* from senseFly (b)

on a hobby model platform [77]. They have a shape of a conventional aircraft with 4 control surfaces (*aileron*s, *elevator* and *rudder*) following the model presented in Sec. 3.5. The second platform (Fig. 6.1(b)) is a delta-wing produced by senseFly SA with a modified payload and firmware. The model for such a platform was presented in Sec. 3.5. Tab. 6.1 summarizes the main characteristics of all three platforms. The empty mass of the platform does not include

Table 6.1: General characteristics of the UAVs

	TP1	TP2	eBeeX
empty mass [g]	1791	1819	946
length [m]	1.148		0.706
width (wing Span b) [m]	1.623		1.175
wing Span S [m ²]	0.343		0.32
Mean aerodynamic chord \bar{b} [m]	0.225		0.295
Propeller diameter D [m]	0.362		0.23
autopilot HW	PixHawk FMU2		Proprietary
autopilot SW	ArduPilot - custom	PX4 - custom	Proprietary
GNSS receiver	μ blox + Javad TRE	TOPcon B11 (B125 later)	Septentrio AsteRx-m2
GNSS Antenna	Maxa3	AN306-1	Maxtena M1227

the weight of the removable payload and the battery. These may change per configuration,

modifying the total mass m , which is used as a VDM parameter. The platform length is the distance between the nose (further point of the engine/propeller) and the rearmost element on the platform structure. The width of the platform corresponds to the wing span b introduced in Sec. 3.5. The autopilot board with its software performs several tasks, which can be generalized as, gathering the data from internal and some external sensors, controlling the actuators in flight, guiding and navigating the platform during its mission, and communicating with the ground control. More details on the autopilot are given in Sec. 6.3.2. As the main findings are based on the *TP2* platforms, a detailed architecture of the experimental setup is given below. The payload and some of the system for the delta-wing drone *eBeeX* are provided in the Appendix C.4

6.2 Payloads

Some objectives of the research group, in parallel with the research objectives of this work, require the use of hardware of sufficient accuracy (e.g. IMU) or computational capacity. Some of these requirements can be listed as follows

- receive, decode and re-transmit high frequency IMU data of a quality superior to autopilot IMU,
- execute a pre-flight calibration for (online) determination of random biases in all inertial sensors (Sec. D.3.3),
- perform INS-based (kinematic) navigation that initializes VDM-based (dynamic) navigation in the air and serves as a backup in the case of resets and terminal phase (landing) of the flight (Sec. 7.2),
- dispose of a sufficient computational capacity for implementing and executing the real-time VDM-based navigation (Sec. 7.3),
- live streaming the control commands from the autopilot to the real-time navigation software and return the navigation solution to the ground.

Several payloads are utilized to validate the research objectives of this thesis. An overview of the different payloads is given below, Tab. 6.2 summarizes the hardware components, and Fig. 6.2 depicts the three payloads used. The mass of the payload "IGN-GECKO" was not measured but was the total mass of the UAV with the payload.

6.2.1 "IGN-GECKO"

Displayed in Fig. 6.2(a), this payload carried by *TP2* is composed of: i) a custom, 20 Mpx camera for aerial photogrammetry developed by IGN, France [79], ii) a Gecko4Nav redundant IMU board [80] with two Intersense NavChip MEMs IMUs.

Table 6.2: Payload characteristics

name	IMU	CAMERA	PC	Other	Weight [kg]
IGN-GECKO	2× NCv1	DigiCam	raspberry-Pi		-
SODA-GECKO	4× NCv1	SODAv1	UpBoard		0.443
SODA-STIM	1×STIM 318	SODAv1	UpBoard	Dahu4NAV & 4× <i>ADIS-16475</i> [78]	0.473

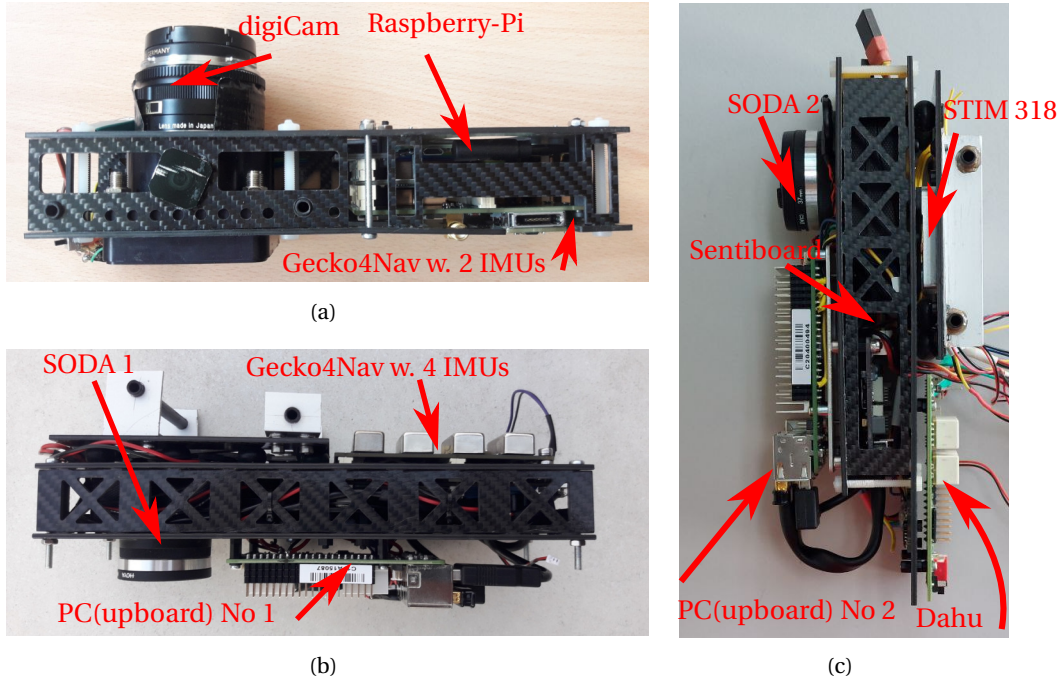


Figure 6.2: (a) "IGN-Gecko" with a high-quality camera (digiCAM), board with 2 redundant IMU and embedded 32-bit micro-computer, (b) "SODA-Gecko" with SODA camera, 4 redundant IMUs, and embedded 64-bit computer UpBoard, (c) "SODA-STIM" with an UpBoard, SODA camera, *STIM-318* IMU, SentiBoard and Dahu board

The calibration of the external and internal parameters of the camera was performed thanks to [75]. The weight and size of the camera allowed to use of only a μ PC (Raspberry-Pi), which is not used for VDM. All data from Gecko4Nav board are stored in internal memory together with selected GNSS data. The stochastic error model of this IMU is described in [81] and the prior-mission calibration in [53]. The on-board GNSS multi-frequency (Global Positioning System (GPS)/GLObalnaïa Navigatsionnaïa Spoutnikovaïa Sistéma (GLONASS)) receiver TOPcon B110 has its storage for all observations, including phase, phase rate, and pseudo-ranges on multiple frequencies. These are required to obtain cm-level positioning and cm/s velocity accuracy, which are used for calibration and reference. This receiver time-tags the pulse-signalized events of camera shutter openings. It also provides the Pulse Per Second (PPS) to the Gecko4Nav board and the autopilot (PixHawk) to associate the IMU data and the autopilot (Pixhawk) control commands in the GPS time scale, respectively. The payload is attached

directly and rigidly to the *TPs* via two carbon rods.

6.2.2 "SODA-GECKO"

A replacement system is developed that i) can provide an external attitude reference of sufficient quality and ii) has sufficient computational capacity for the requirements mentioned at the beginning of the section. A smaller and lighter camera, referenced as "SODA" is acquired from senseFly SA^I. The optical properties of the camera are somewhat lower quality than those of digiCAM, yet sufficient for external attitude referencing. The lower size and weight of the camera allowed to the integration of an embedded computer. In contrast to "IGN-GECKO", it is attached to the plane via rubber vibration dampeners. A similar Gecko4Nav IMU board with four redundant IMUs is presented in the payload. A 64-bits companion computer [82] is incorporated to handle the real-time acquisition of the different sensor measurements, as well as to run real-time applications. Apart from other functionality described in Sec. 7.2 and Sec. D.3.3, this computer interfaces the communication with the camera to acquire the images. The mounting hardware is depicted in Fig. 6.2(b). More details on each hardware component are provided in Sec. 6.3, the communication between them in Appendix C.1, and the interaction with the rest of the UAV in Sec. 7.2.

6.2.3 "SODA-STIM"

To improve the calibration of the aerodynamic coefficients used in the VDM, accurate inertial data is required. Together with the prospect of using several small IMUs of new generation, the new payload replaced the R-IMU board containing Navchips (the release of which dates back to 2010 [80]). A *STIM-318* from Sensoror^{II} representing one of the best options currently on the market in terms of the performance per size and weight is accommodated (6.2(c)) alongside the "DAHU" [83] board hosting 4 *ADIS-16475* IMU from Analog Devices [78] (noise characteristic given in Tab. C.2). The new payload carries the same type of SODA camera and a similar embedded computer^{III} as the "SODA-GECKO" payload.

A final assembly is presented in Fig. 6.2(c) with a carbon fiber structure, rubber dampeners, and aluminum anchor points. The IMU is accompanied by an electronics board called the Sentiboard (Fig. 6.2(c)), which is developed at NTNU [84] for sensor data acquisition and synchronization. The payload is used to collect data for the aerodynamic coefficient estimation methodology presented in Sec. 5.2, and the flights are described in Sec. 7.4. The board wiring is given in Appendix C.6.

^I<https://www.sensefly.com/>

^{II}<https://www.sensoror.com/products/inertial-measurement-units/stim318/>

^{III}up-board

6.3 Sensors and Subsystems

The real-time implementation of a demonstrator is complex in terms of hardware, software, and communication architecture. Realization of the full system required substantial development and component customization. This section describes the hardware layer of sensors and subsystems with the architecture, while the detailed software layer and implementation is presented in Chp. 7.

6.3.1 Overview

The whole system is divided into two distinct components: the fixed-wing UAV and the GCS, required for issuing commands to the UAV and visualizing telemetry including the VDM-based performance and other measured parameters. Fig. 6.3 summarizes all hardware components with their firmware developed or modified to meet the real-time implementation objectives.

Plane:

1. the PixHawk flight controller, hosting the autopilot software,
2. the Upboard, a powerful yet miniaturized computer dedicated to running computationally intense tasks related to VDM-based navigation,
3. IMU payload:
 - (a) the Gecko4Nav board with the NavChip IMU for the "SODA-GECKO"
 - (b) the combination of *STIM-318* + SentiBoard for the "SODA-STIM"
4. A GNSS receiver, in order of usage:
 - (a) JAVAD TRE
 - (b) TOPcon B110
 - (c) TOPcon B112
5. A SODA camera (in payload "SODA-GECKO" and "SODA-STIM"), a digiCam (in payload "IGN-GECKO")
6. Two airspeed sensor of different quality
 - (a) JDrone
 - (b) Surrey

Components 2, 3 and 5 are assembled in one single removable payload depicted in Fig. 6.2(b) already introduced in Sec. 6.2.

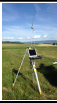








Hardware	Software	Custom Modifications
	QGC	Custom Commands
		Plot nav. solution
	Custom	Custom
	PX4	Javad / Topcon driver
		GNSS time clock sync
	Ubuntu + ROS	MAVROS
		Clock Sync + BroadCast
		NavServer
		Giinav
	Gecko + Custom FW	NavServer
	SentiBoard + Custom FW	NavServer
	Original	NavServer
		B110 split board
		B125 split board
	soda.service	EV and TRIG
	PX4	NavServer
	SentiBoard + Custom FW	SurreyNode

Figure 6.3: A list of hardware (HW) and software (SW) modules

Ground:

1. regular computer hosts the GCS software, required to issue commands to the UAV, plot the navigation solution, and display other significant parameters, such as the results of the real-time navigation filters. The ground station will be detailed in Sec. 6.3.3,
2. the custom-made weather station to observe the horizontal local wind velocity (Sec. 6.3.4)
3. a radio command to control the plane manually, especially for take-off and landing.

The hardware mentioned above with the wiring/radio link between the components for the TP platforms and the ground segment is shown in Fig. 6.4 in a general and simplified scheme.

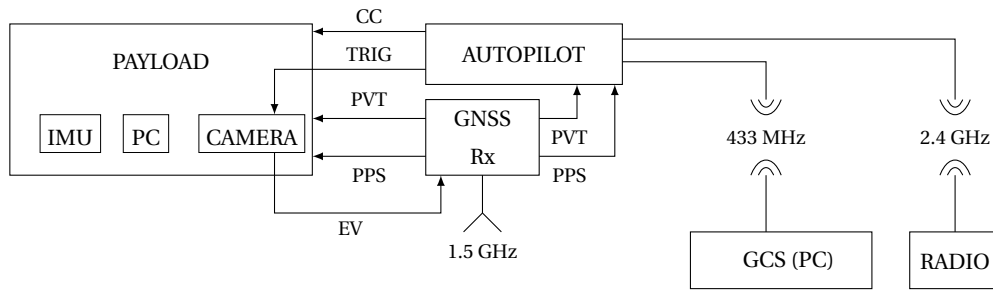


Figure 6.4: General connection between hardware components

6.3.2 In the UAV

The PixHawk Flight Controller and PX4 Autopilot Software

The Pixhawk autopilot is a popular general-purpose flight controller based on the Pixhawk-project [85] FMUv2 open hardware design. It is based on an embedded μ controller and can run different implementations of autopilot software. It hosts all the electronics to interface with other flight hardware, such as an embedded computer, a GNSS receiver, radio links, etc. In particular, it is directly connected to the servo motors that position the flight surfaces and to the Electronic Speed Controller (ESC) that drives the main propeller.

To achieve research objectives, the customization of the autopilot is a priority. PX4 [86], an advanced open-software autopilot software, has been selected. The architecture of the PX4 autopilot, developed in academia and now the standard in the UAV robotics community, ensures long-term support and high-quality standards, and it is more suited to implement the features required for this project. The PixHawk board is used as the flight controller in the constructed plane. The autopilot software is modified to be compatible with the GNSS receiver and for enabling time-stamping of autopilot commands and sensor data in GPS time-frame.

In particular, PX4 has been extended concerning the following functionalities:

1. **Javad GNSS driver.** No available autopilot software is already equipped with the necessary drivers to integrate the measurements from Javad receivers. Although Javad is a leader in high-end receivers for geomatic applications, such products are seldom considered on UAVs due to price limitations. A general driver has been implemented for the whole Javad product family, based on the GREIS binary protocol, also implemented on Topcon receivers.
2. **PPS acquisition.** The Javad GNSS receiver outputs a PPS synchronous with the GPS clock. A software module has been implemented in the PX4 autopilot to enable the timestamping of such pulses, i.e., recording the exact time (with respect to the autopilot clock) of the pulse rising edge.
3. **Real-time clock synchronization.** Once the exact time of the PPS coming from the

receiver has been recorded in Autopilot (AP) time, this information can be used to establish the relation between the low-quality autopilot clock and the GPS time reference, in terms of offset and drift. A software module has been implemented to compute these two parameters in real-time based on a history of pulse timestamps and GNSS epoch times. These two parameters are encoded in an μ ORB message on the `debug_vect` topic. Thanks to the Micro Air Vehicle communication protocol LINK (MAVLink) [87] bridge, this message is also exposed to the embedded computer and made available to the ROS environment by MAVLink extendable communication node for ROS (MAVROS) (see Fig. 7.1).

The PixHawk also receives data from an airspeed sensor (Pitot tube) which is used to control the velocity of the UAV.

The UpBoard Embedded Computer

The embedded computer chosen for this project is the UpBoard [82]. This miniaturized board hosts an Intel Atom 64-bit processor, and it can run computationally intense tasks, such as those required by VDM-based navigation, in real-time.

A more detailed description of the software running in the Upboard and the ROS environment is given in Sec. 7.2. The installation of the UpBoard embedded computer can be found in a brief Wiki^{IV}.

NavChip IMU and Gecko4Nav

The IMU sensor(s) used for VDM-based navigation are part of the Gecko4Nav board embedded in the "IGN-GECKO" and "SODA-GECKO" payloads depicted in Fig.6.2(a), 6.2(b). The Gecko4Nav board [80] was developed in collaboration between EPFL and The Bern University of Applied Science. The version flown with the "IGN" payload has two integrated IMUs, while the current version ("SODA-GECKO") has four IMUs.

The Gecko4Nav board includes an FPGA and an integrated circuit to host up to four IMUs and a port to receive GNSS data for time reference. Their multiplication improves redundancy and potentially decreases noise, and thus improves accuracy, as studied in [88]. The connected receiver sends a PPS to the Gecko4Nav board to synchronize the acquisition of the respective IMUs with GPS time scale. All data from Gecko4Nav board are stored to board internal memory together with selected GNSS data. Its noise characteristics are analyzed in [89] and are given in Tab. C.3.

^{IV}<https://wiki.epfl.ch/topoupboard>

STIM-318 and SentiBoard

The IMU streams the observations at 500 Hz to the SentiBoard, which takes care of the time tagging of the data. The SentiBoard board has several configurable ports of different protocols to connect sensors as input and a USB port to stream sensor data as output. One serial port handles the arrival of IMU data. A pulse so-called “Time Of Validity (TOV)” is issued by the IMU for each outputted data packet, which is time-stamped by an internal clock. This timestamp is encapsulated with the data and sent to the PC. A second serial port handles messages from the GNSS receiver, where the TOV is provided by the PPS signal issued at the same frequency as the GNSS messages. Incoming messages are encapsulated with GPS time and sent to the PC. One of these messages contains information about the absolute GPS time. The developed parsing software finds the relation between the board clocks and the GPS time (Sec. 7.1).

GNSS Receivers

The three GNSS receivers used during the different campaigns with the *TPs* are summarized in Tab. 6.3.

Table 6.3: GNSS receivers used in chronological order

Device	constellation	frequency	Platforms
JAVAD TRE	GPS/GLO	L1/L2	TP1
TOPcon B110	GPS/GLO	L1/L2	TP2
TOPcon B112	GPS/GLO/GAL/BEI	L1/L2/E5	TP2 (after IGN campaign)

They allow the recording of raw GNSS observations in their storage. These observations can be fused with other data of a stationary receiver of a similar type to obtain an accurate position and velocity for reference and calibration. The position and velocity determined by the receiver are communicated to the autopilot and the payload via serial port, from which are forwarded together with IMU data to the Upboard embedded computer. The receiver also generates the PPS signal to synchronize the system clocks on other components, e.g. IMUs and flight CC.

Camera

The camera is an auxiliary sensor that can be used to derive a reference attitude through photogrammetry and to determine the VDM parameters further as described in Sec. 5.3.2.

The *SODA* camera is a proprietary device of senseFly^V with a CMOS sensor of 5272×3648 pixels on 12.75×8.5 mm footprint. The lens is fixed rigidly to the camera body and has a nominal focal length of 10 mm. The ground resolution is about 2.33 cm/pix when the image is

^Vwww.sensefly.com

taken 100 *m* above the terrain. The weight of the camera is 75 *g*.

The digiCAM [90] from Institut National de l'Information Géographique et Forestière (IGN) is likely one of the best small camera systems (in terms of resolution, quality, and stability) that such UAV platform can carry. It comprises a full-frame-sized CMOS sensor and GPS-time-ready electronics on a stable mount to which optics of excellent quality are attached (Zeiss Biogon 2.5/35 mm lens). It, however, weighs 300 *g*. The cameras are triggered to take pictures via a mechanism described in Sec. C.1.4.

Airspeed Sensor

Two airspeed sensors are installed on the *TP2*. The *Jdrone* airspeed sensor is a low-cost PixHawk-compatible used to control the airspeed of the UAV during the mission. The sensor estimates the wind velocity for the calibration method WMF. The sensor is, however, not used for VDM-based navigation.

The second Pitot tube from SurreySensor^{VI}, is of high quality and connected to the Senti-Board. It outputs measurement at 100 Hz and is used to determine the real-time air density ρ , transmitted to the VDM-based navigation system (Sec. D.3.3).

^{VI}<https://www.surreysensors.com/>

6.3.3 Devices on the Ground

Radio Control

To operate manually the UAV, a *MC-32* from *Graupner*^{VII} is used as RC. It is connected to the *3DR Radio v1* receiver module from the *SiK* project^{VIII} linked to the autopilot. The system used the WiFi bands at 2.4 GHz . The controller permits the arming and disarming of the UAV and change between flight modes (full manual, stabilized, orbits) programmed in the autopilot. There is no automatic take-off and landing capability used for the *TPs*, thus, the UAV operator has to control the UAV for take-off and landing. The *eBeeX* however, can fly fully automatically from launch/takeoff to landing, no RC is needed.



Figure 6.5: RC used by the *TP* operator

Ground Control Station - Mission Planner

The GCS comprises a field computer and antennas to maintain duplex communication with the UAV throughout the mission. A typical field setup is shown in Fig. 6.6 with the operator computer and the 443MHz transceiver as presented in Fig. D.2(b). The



Figure 6.6: Ground Control Station set up with the operator's computer running QGC and the telemetry transceiver

software adopted on the remote PC is QGC [91]. It provides a complete flight control setup for the linked autopilot (PX4). It communicates to the UAV via MAVLink protocol. Some of the messages are described in Appendix C.1. The software handles entire mission planning for autonomous flight and, thus, allows total control of the UAV trajectory.

A Graphical User Interface (GUI) display, as seen in Fig. D.2(a), eases the operator to send commands to the UAV and monitors in real-time the vehicle position, flight track with defined waypoints as well as the UAV instruments and subsystems.

^{VII}<https://www.graupner.com/>

^{VIII}<https://github.com/ArduPilot/SiK>

6.3.4 Portable Weather Station

The air density ρ varies substantially at the same location with different weather conditions (> 20%). These variations depend on the time of day, the altitude at which the UAV flies, and the season. Variations above ground are relatively small for UAVs flying below 150 [m] Attitude Above Ground Level (AGL) and can be modeled with relatively good confidence. The large variations of ρ due to current meteorological conditions at the take-off area are deterministic when observing the temperature T , pressure p , and humidity as

$$\rho = \frac{p_d}{R_d + T} + \frac{p_v}{R_v + T} \quad (6.1)$$

where ρ is the density of humid air (kg/m^3), T is the temperature in Kelvin, R_d and R_v are the specific gas constants for dry air and water vapor, respectively. The constants and derivations of dry air p_d and water vapor p_v can be found in [92]. These observations motivated the installation of a small portable weather station that monitors the values mentioned above together with the direction and magnitude of wind to compare its estimation with the VDM-based navigation system.

Sensors

A small portable weather station is developed in three copies. Each accommodates the sensors observing the local air parameters, as shown in Fig. 6.7(a) and wind velocity in 2D direction. On

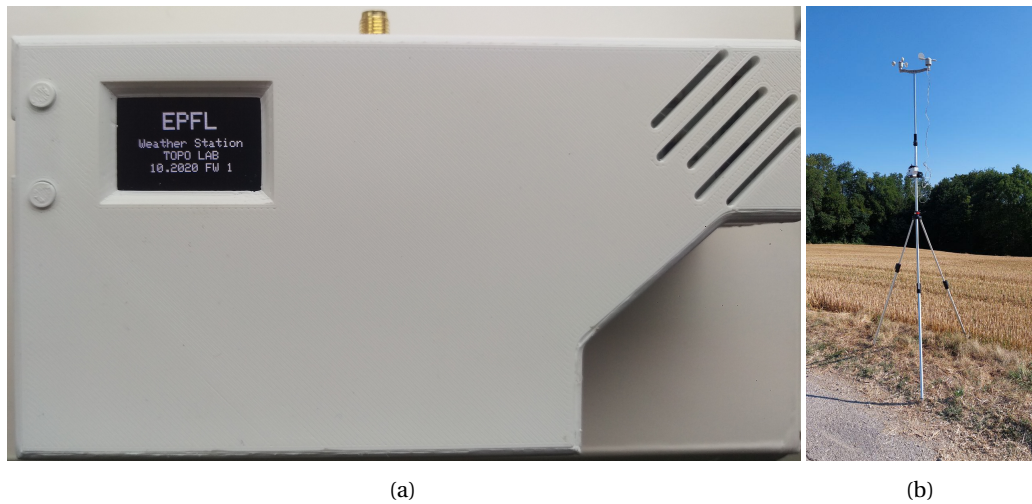


Figure 6.7: (a) Portable Weather Station main module and (b) acquisition set-up with tripod

the surface of less than 20 cm^2 the weather station contains the sensors and devices presented in Tab. 6.4 An anemometer indicating wind direction and velocity is attached to the setup as depicted in Fig. 6.7(b). The anemometer on the weather station can identify 16 cardinal directions thus, the resolution is 22.5deg . As a result, the output is a coarser estimate than the

Table 6.4: Weather station specifications

Sensor type	Reference	Details
Static pressure	MS583702BA06-50	
Humidity	DEL-SHT85	
Temperature	DEL-SHT85	
GNSS receiver pressure	uBlox CAM-M8	for data GPS time-tagging
μ processor	ESP32	w. 1.14 Inch LCD display and WiFi (2.4 [GHz])
SD card		Internal storage of observation
Battery		1 Cell 3.5V
Anemometer	SEN-08942	
I/O	Analog I2C Serial RS485 USB	RJ11 - Anemometer and rain gauge, USB - future uses

estimated wind. However, it can already give some approximate knowledge of the wind, which can be helpful for the initialization of the states as presented in Sec. 9.4.1. The anemometer must be calibrated before each use for velocity and North heading. The weather station source files and the guide to update the firmware, the electronic schematic, layout, and the 3D printed cover can be found at [93]. Some of the diagrams are presented in Appendix C.5.

Estimation of Atmospheric Parameters

In the Sec. 3.5, the force and moment equations were defined for the *TP2* and the *eBeeX*. All equations are correlated with the air density ρ . Moreover, the specific forces (Eq. 3.58) are expressed in the wind frame. While the wind velocities are estimated as auxiliary states, the air density ρ can be set as a constant parameter or added as a new state when the aerodynamic coefficients are removed from the system-states as presented in Sec. 7.3.5. This new state behaves as a scale factor s that compensates and corrects the air density variations due to altitude changes from a model or a slow change in the weather conditions.

To increase the trust in estimating the aerodynamic coefficients using the different methods presented in the Chp. 5, the air density ρ can be set before the takeoff according to the current atmospheric parameters observed by the portable weather station. The air density can also be observed in real-time with the onboard embedded sensors such as a barometer, thermometer, and humidity via to Eq. 6.1. In either case, it is believed that the estimated VDM parameters values are closer to their true values with the observed ρ rather than that obtained by the standard atmospheric model. Even though an incorrect initial ρ gets compensated by the re-estimation (via in-flight calibration) of the aerodynamic coefficients, a value approaching the current true air density is in the best interest for the VDM parameters consistency between flights occurring at different time, altitude, and weather conditions.

The same reasoning applies to the initial wind velocities. With approximated values (Sec.9.4.1)

(e.g. taken thanks to the weather station), the convergence time to estimate the wind velocity can be accelerated. In turn, the estimation of the other states benefits from it (navigation, VDM parameters, IMU bias states). In the case of GNSS outage, improved wind estimation is translated to higher accuracy of autonomous navigation.

6.4 Flights

This section gives an overview of the flights performed during the thesis with their main characteristics presented in Tab. 6.5.

6.4.1 VDM Concept Validation with Experimental Data

In 2017, Dr. Khaghani [11] used the two flights called *20170821_nx5id1* and *20170822_nx5id1* and shown in the Fig. 6.8 to confirm experimentally the VDM-based navigation concept, which was still in an early stage. The two flights demonstrate the importance of correct sensors and

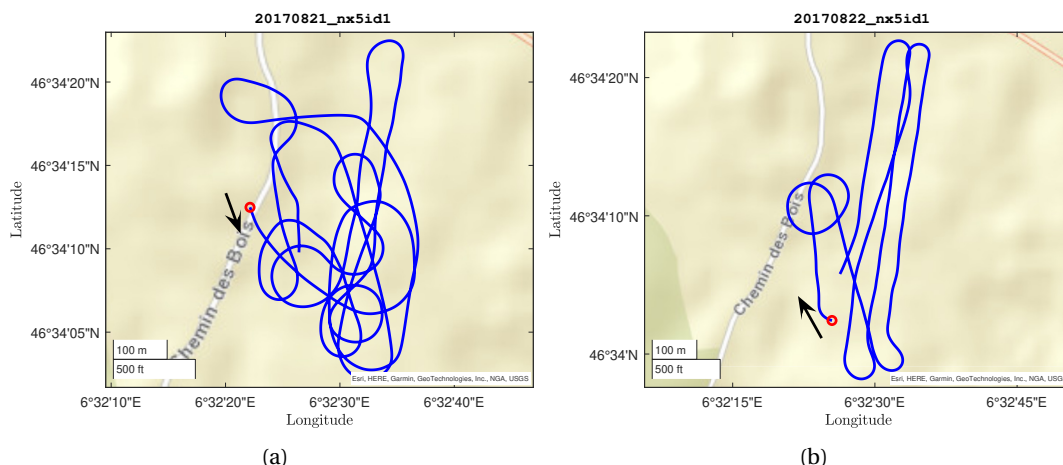


Figure 6.8: Flight performed in August 2017 with the *TP1* under the names (a) *20170821_nx5id1*, and (b) *20170822_nx5id1*

time-tagging of autopilot flight control commands for the VDM-based navigation system to perform correctly. These results are given in Sec. 9.1.1. The payload comprises the MEMS-IMU Navchip with the Gecko4Nav introduced in Sec. 6.2. A geodetic grade GNSS receiver is on board, capable of working in RTK mode, although only the stand-alone solution is used in navigation. The sampling frequencies are 100 *Hz* for the IMU, 10 *Hz* for the barometer, and 1 *Hz* for the stand-alone GNSS position and velocity data.

Table 6.5: Flight summaries

Flight Name	Date	L. [min]	UAV	Payload	Weather (WS)	Sec.	Reference	Details
20170821_nx5id1	2017-08-21	8	TP2	Legacy	SKC, 0-1m/s V, (×)	9.1.2	[95]	Test calibration maneuvers
20170822_nx5id1	2017-08-22	10	TP2	Legacy	SKC, 0-1m/s V, (×)	9.1.2	[95]	Test calibration maneuvers
16X	2018-06-21	23	TP2	IGN-GECKO	SKC, 0-1m/s V, (×)	8.3.19.3.2	[28]	
16U	"	18	"	"	"	9.3.2		
17	2018-07-02	29	TP2	IGN-GECKO	SKC, 0-1m/s N, (×)	8.3.19.3.2	[28]	
18	2018-07-03	40	TP2	IGN-GECKO	SKC, 0-1m/s V, (×)	8.3.19.3.2	[28]	
eBeeX-652	2018-09-26	34	eBeeX	eBee-GECKO	-	8.2	[76]	
eBeeX-757	2019-02-20	32	eBeeX	eBee-GECKO	-	8.2		
GiiNav2	2019-02-28	-	TP1	SODA-GECKO	SKC, 0-2m/s SW, (×)			Test modified NavServer and GiiNav in RT with ROS
GiiNav3	2019-09-17	-	TP2	SODA-GECKO	SKC, 4-5m/s N, (×)			NavServer and GiiNav in RT with ROS
sodaCorrL	2019-10-01	18	TP2	SODA-GECKO	SKC, 4-5m/s N, (×)			Test Soda + UpBoard
GiiNav4	2020-04-23	-	TP1	SODA-GECKO	SKC, 0-2m/s N, (×)			Test Tel2 with CC, CM from QGC, correct rosbags
t1px4-rosb3	2020-06-03	-	TP1	SODA-GECKO	SKC, 0-2m/s S, (×)			2 flights: without payload; Test custom QGC, rosbags for CC, GiiNav
t1px4-soda1	2020-07-06	-	TP1	SODA-GECKO	BRK, 0-4m/s SE, (×)			3 flights: 2×without payload; Test soda with UpBoard services
tp2s1-met1	2020-10-07	23	TP2	SODA-GECKO	SKC, 0-4m/s S, (✓)			Test CC, weather station
tp2s1-met2	2020-11-09	34	TP2	SODA-GECKO	FOG, 0-1m/s V, (✓)			WS with new time tag, Camera new wiring, GiiNav Topics
STIM3	2021-04-01	-	TP2	SODA-STIM	-, (✓)	A.8		Test new weather station FW
STIM5	2021-09-27	30	TP2	SODA-STIM	BRK, 1-3m/s S, (✓)	8.2		RT ROS topics for VDMc
STIM6	2021-10-12	32	TP2	SODA-STIM	SCT, 3-5m/s N, (✓)	8.2		RT ROS topics for VDMc
STIM7	2021-11-11	16	TP2	SODA-STIM	OVC, 1-2m/s N, (✓)	8.2		RT air Data
STIM8	2022-07-01	32	TP2	SODA-STIM	FEW, 1-2m/s V, (✓)	7.3.4		Test all nodes in RT
STIM_12	2022-08-30	20	TP2	SODA-STIM	SKC, 0-1m/s V, (✓)	9.2		VDM demonstrator, wind estimation and alignment
STIM_13	2022-08-30	23	TP2	SODA-STIM	SKC, 1m/s S, (✓)	9.2		VDM demonstrator, wind estimation and alignment
STIM_14	2022-10-25	28	TP2	SODA-STIM	SKC, 0-1m/s SE, (✓)	9.2		VDM demonstrator, gnss outages
STIM_15	2022-10-25	16	TP2	SODA-STIM	SKC, 1-2m/s S, (✓)	9.2		VDM demonstrator, gnss outages

6.4.2 IGN-GECKO

During the summer of 2018, several flights with the payload "IGN-GECKO" (Sec. 6.2.1) and the platform *TP2*. Initially, these flights were carried out for mapping and redundant MEMS-IMU related research [5, 75]. However, the advantage of the high-resolution camera created the opportunity to explore the improvement of the estimation of the coefficients due to accurate attitude observations. Four of them were selected, and their trajectories are shown in Fig. 6.9. Two flights (*IGN8*, *IGN6X*) were released as open-source data [94]. The processing of these

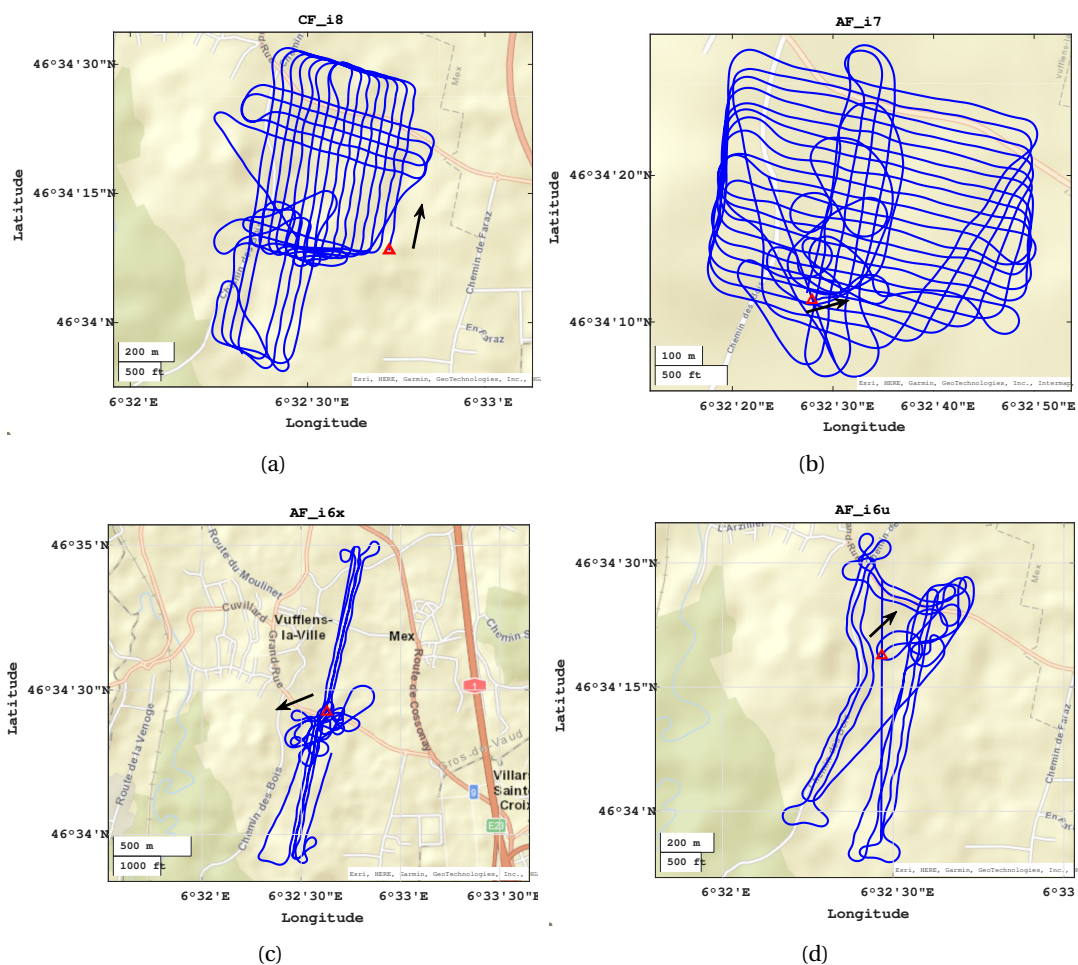


Figure 6.9: Experimental flight references (blue): a) *IGN8*, b) *IGN7*, c) *IGN6X* and d) *IGN6u*, beginning of the trajectory (red triangle)

flights required data logging and was processed in the VDM MATLAB framework.

6.4.3 SODA-GECKO

Starting from July 2019 to November 2020, the payload "SODA-GECKO" (Sec. 6.2.2) was embedded first in *TP1* (up to July 2020), then in *TP2* to test the change of autopilot firmware (from

Ardupilot [96] to PX4) and the installed real-time environment ROS with the new embedded computer. At the same time, the hardware (Sec. 6.3) was replaced in the *TP2* (GNSS receiver from Topcon B110 to B125 with a change of antenna from Maxtena [97] to AN306-1) and required further testing. Due to this payload, the data recorded (from ROS bags) are replayed in a simulated real-time environment (Sec. 7.3) to validate the possibility of executing the VDM-based navigation prototype in term of computational loads with the real-time navigation software, the description of which is given in Sec. 7.3. Among all flights, a subset is described in Tab. 6.5 for their key steps toward achieving research objectives.

6.4.4 SODA-STIM

Mounting, installation, and testing of the new "SODA-STIM" (Sec. 6.2.3) payload started from November 2020. Using the *TP2* and this payload, high-quality data were acquired during numerous flights to validate the proposed VDM parameter estimations (Sec. 5.2). Some of the following flights were used to validate the calibration method proposed in Sec. 5.2 and to demonstrate the real-time VDM-based navigation software (Sec. 7.3) with an online demonstrator (Sec. 9.2). Fig 6.10 shows these flights, and their details are in Tab. 6.5.

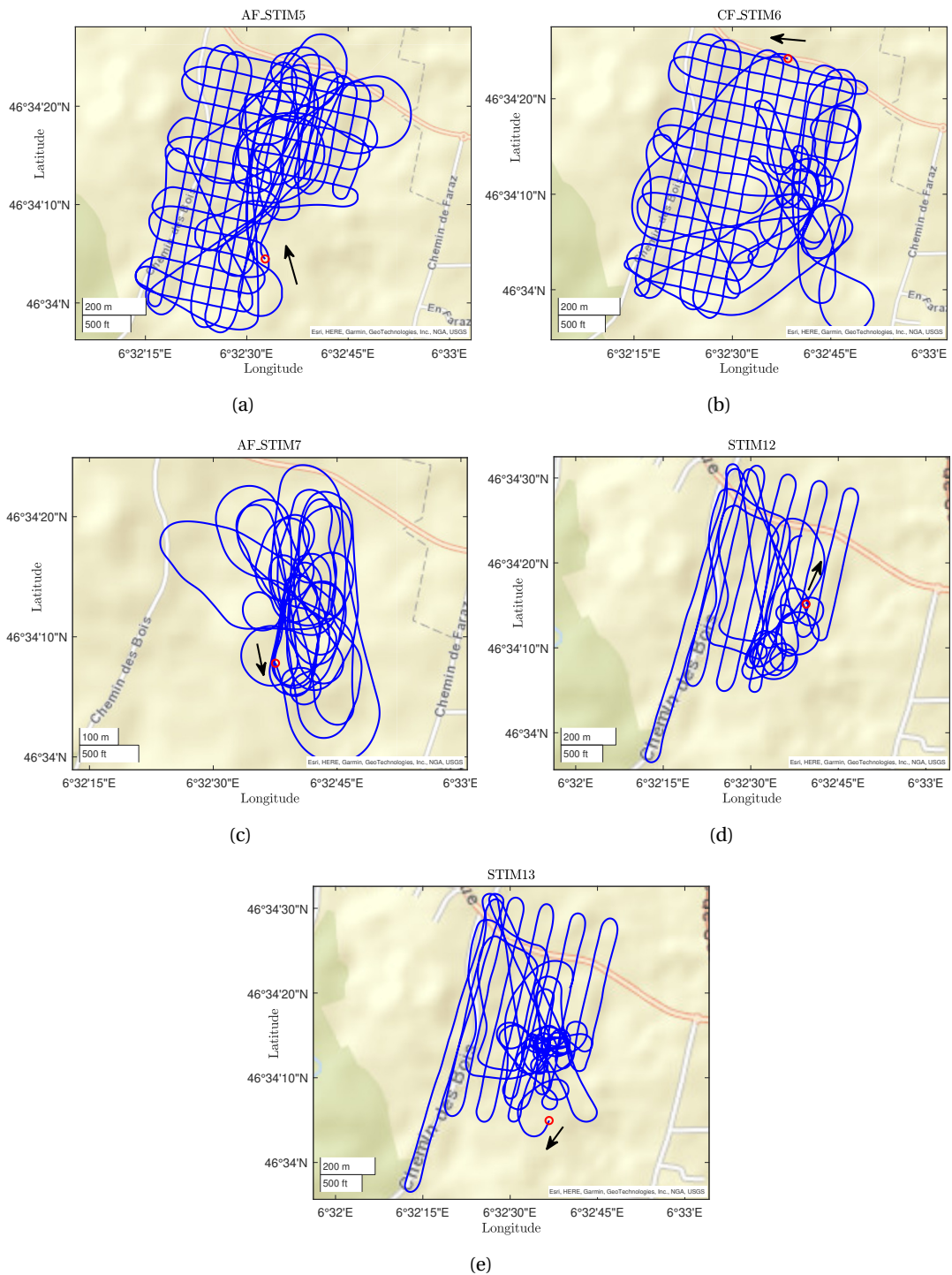


Figure 6.10: Experimental flights references (blue): a) *STIM5*, b) *STIM6*, c) *STIM7*, d) *STIM_12*, and e) *STIM_13*, beginning of the trajectory (red circle)

6.4.5 eBeeX Campaign

While the research on autonomous navigation with the fixed-wing platform *TP2* was progressing, a parallel project ^{IX} with the delta-wing *eBeeX* started in 2017. From the numerous flight attempts to gather all required and correctly time-tagged autopilot and payload data (more than 25!), between October 2017 and February 2019, two flights are used in this work. Their trajectories are depicted in Fig. 6.11 while some details of the flights are given in Tab. 6.5. No

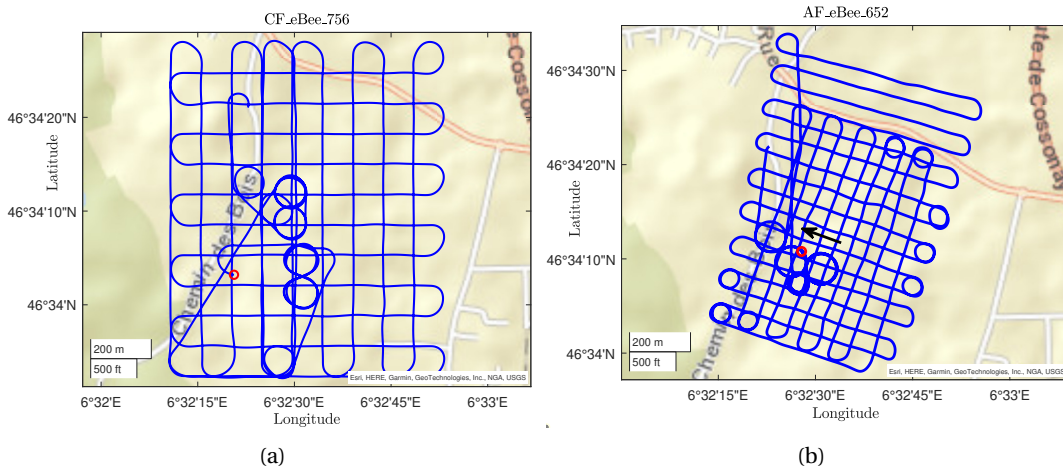


Figure 6.11: *eBeeX* flight trajectories with *CF_eBee_756* (a) used as calibration and *AF_eBee_652* (b) as application flights

real-time environment is used for these flights, and all data are recorded for post-processing and the validation of the WMF methodology (Sec. A.3.1) with the delta-wing *eBeeX*. A description of the payload is given in Appendix C.4.

Summary

In this chapter, the main hardware components for the experimental setup have been described. The detailed narrative has emphasized the complexity of designing a working VDM-based navigation system while demonstrating that most parts can be easily obtained on the market. The VDM-based navigation system has been tested on a large number of flights. The numerous flights acknowledged the complexity of testing and validating the real-time design. The software application and constraints related to VDM-based navigation real-time aspects are introduced in the next chapter.

^{IX}CTI project number 25800.1 PFIW-IW, *VDM2NAV*

7 Real-Time Implementation

Overview

The real-time implementation of the VDM-based navigation system requires new software development and several adaptations of the experimental setup presented in the previous chapter. This chapter presents the chosen strategy for handling data exchange from multiple sources, and details the necessary software applications to achieve a working real-time environment. To begin with, the system needs a common reference time frame to fuse the flight control commands provided by the autopilot and the different sensor observations (that are not directly part of AP electronics). By doing so, the sensor fusion filter is guaranteed to deal with the data at the correct (common) time. Second, it is described how the data are exchanged between sensors is performed. This is achieved using ROS-topics, which is a data communication layer managed by the ROS environment. It makes the sensor-to-application interface independent from the hardware. The publishing of data messages (ROS-topics) and their reception (ROS-subscription) is managed by applications (ROS-nodes). With this scheme, the I/O of the navigation software does not need to be adapted for different hardware realizations. In addition, the sensor observations originating from different subsystems are generated asynchronously and ROS guarantees the exchange between nodes with the same reference time. The transmission of messages from/to the embedded computer, the autopilot, and the GCS are briefly presented. Within the embedded computer, the different applications, called the ROS nodes, are described too. Then, the VDM-based navigation filter, called VDMc, is described in more detail together with its real-time features. This software runs VDM navigator and its corresponding EKF for the particular platform and sensors at hand. It is responsible for fusing all sensor observations and flight control command inputs to estimate the navigation solution and the auxiliary states (wind, IMU errors, aerodynamic coefficients). Finally, the different phases of flight are discussed for a typical UAV mission. Each phase implies different software and filter configuration that assures the best navigation performance and filter stability, in particular during VDM initialization, GNSS outages, and landing.

7.1 Data in GPS-Time

VDM-based navigation requires providing a common time frame to all observation sources (IMU, GNSS, barometer, airspeed, etc) with respect to autopilot commands. As the sensor observations from MEMs-IMU refer to the absolute time frame driven by the GNSS receiver via the NavServer application, the autopilot needs to be modified to express its control commands in GPS time as well as observations from connected sensors such as the Pitot tube and the barometer. These observations are further sent to the UpBoard computer via the MAVlink-MAVRos bridge to the VDMc application as updates or references. Their fusions with other observations and GNSS also require association with GPS time. Tagging the CC (and other sensors data) with GPS-time requires finding the relation between the internal clocks of the autopilot and GPS time, scale of which is signaled by the PPS with associated time message with registered in the AP system time. To accomplish this task, time series of PPS is continuously analyzed by a linear regression with two parameters:

- Bias b : representing the offset of the autopilot internal clock with respect to UTC.
- Scale factor s : accounts for the differences in the autopilot and Coordinated Universal Time (UTC)/GPS time scales¹.

$$t_j^{UTC} = b + \frac{1}{s} t_j^{PX4} \quad (7.1)$$

Online parametric estimation of b and s is implemented with the modified PX4 autopilot. These two values are transferred to the UpBoard computer via the ROS topic `time_sync`. The GPS time is then computed by taking into account the *day of GPS week* and the current *leap second*:

$$t_j^{GPS} = t_j^{UTC} + dow \cdot (24 \cdot 3600) + leapsecond \quad (7.2)$$

This operation is performed in the ROS node `TimeSynch` where the values of b and s are periodically updated. Finally, the time-tag of every message (control command, airspeed and barometer) is replaced by the computed t_j^{GPS} and a new ROS topic is generated with the suffix `_tagged` (Sec. 7.2). The residuals of the conversion over a flight of 30 minutes are given in the result Sec. 9.1.2.

7.2 ROS Architecture in the Embedded Computer

The embedded computer runs the Linux OS Ubuntu Desktop (16.04 for "Soda-Gecko" payload and 18.04 for the "SODA-STIM" payload) to manage two main applications: 1) the INS- and 2) the VDM-based navigation (Gi iNav and VDMc). Each application is implemented separately as a stand-alone executable (ROS node) and communicates with the other nodes via the

¹The scales of UTC and GPS are the same.

7.2 ROS Architecture in the Embedded Computer

ROS publish/subscribe network. Fig. 7.1 shows the schematic of the ROS nodes and the ROS topics used to provide the required information, as well as the detailed structure of the embedded computed software organization. Within this diagram two principal entities can be

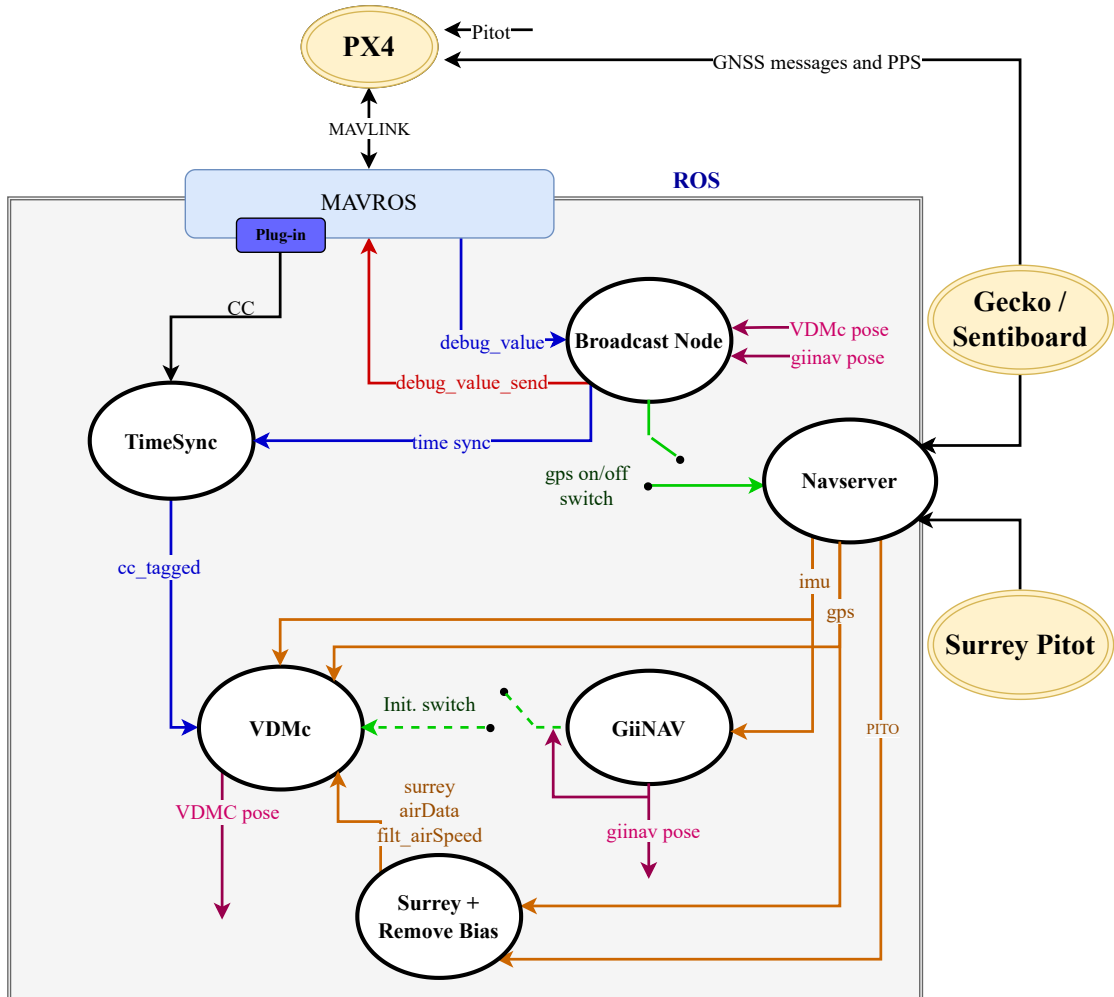


Figure 7.1: ROS architecture of the embedded PC

distinguished:

- ROS Nodes: processes capable of performing tasks
- ROS Topics: information exchanged among nodes

In Fig. 7.1, nodes are visually represented by circles, and the topics are represented by arrows whose direction indicates which node subscribes or publishes to the attached topics. The description of the ROS nodes is given hereafter:

1. **MAVROS.** MAVROS [98] is the ROS node providing the communication interface for the PX4 autopilot with MAVLink communication protocol. MAVROS decodes the incoming

MAVLink stream and makes that available on equivalent ROS topics. Among others, a specific topic type, `debug_value`, composed of a header and a payload, has been used in this project to broadcast specific information such as control commands, status, and clock synchronization parameters.

2. **Broadcast node.** This custom node subscribes to the `debug_value` topic containing the custom commands to/from the ground control station and sends its content, according to the header, to a set of specific topics.
3. **TimeSync node.** This custom node re-tags all the autopilot control commands received via the MAVROS node thanks to a custom plug-in. It converts the autopilot time to GPS time thanks to the information retrieved from the `time_sync` topic, which contains the bias and scale between these two systems times (Sec. 7.1). Then it publishes the GPS time-tagged control commands on the `cc_tagged` topic.
4. **NavServer.** This custom software publishes sensor data used by the `GiiNav` and `VDMc` applications. It was initially designed to output the parsed data from the `Gecko4Nav` board to sockets and was adapted to the ROS environment. The `NavServer` is connected (via USB) to the `Gecko4Nav/SentiBoard` and gets data from its sensors at the GNSS receiver PPS rate. The software has been modified to extract and publish IMU and GNSS data (and airspeed data for the `SentiBoard`) on separated ROS topics.
5. **GiiNav.** `GiiNav` [99] is an internally maintained software that runs an INS/GNSS navigation filter using an EKF as estimator. The software handles delayed sensor outputs. `GiiNav` has been modified to subscribe to ROS topics to acquired IMU and GNSS data produced by the `NavServer`. It saves its navigation solutions (position, velocity and attitude) and status of the filter in separate files as well as publishing the former to a ROS topic (`giinav_pose`) proving the initial navigation states for the `VDMc` software. The topic is also published to MAVROS and follows the pipeline explained in Appendix C.1, eventually reaching the GCS. The solution is then displayed on the QGC GUI to follow the INS/GNSS-based navigation in real-time.
6. **VDMc.** Is the ROS node running the VDM-based navigation estimator. The different sensor measurements are fused within a EKF and the navigation solution is published on the topic `vdmc_pose` (also forwarded to QGC. The `VDMc` node is detailed in Sec. 7.3.
7. **Surrey.** The custom `Surrey` node is responsible for managing the measurements from the `Surrey` Pitot tube. The node also serves as ROS action client to calibrate the dynamic and static pressure sensor against GNSS. This is necessary to obtain accurate real airspeed and altitude measurements. Both observations are used for calibration but are not added as extra updates for real-time implementation. The node computes the corrected air density ρ which is updated in real-time in the `VDMc` software.
8. **RemoveSensorBias.** This custom application provides a ROS action server implementation to compute the airspeed and barometer sensor bias. The results are subsequently

communicated to the Surrey node to correct the raw measurements to calibrated values.

More details of the nodes and their related topics can be found in the Appendix D.2.2.

7.3 VDMc - Software



Figure 7.2: VDMc logo

The implementation of the first version of VDM-based navigation in C++ is presented in the following sub-sections. The software is available in EPFL gitlab [100]. The implementation of the design and the details are written in the dedicated Wiki pages ^{II}.

7.3.1 Operation Modes

The Linux computer in the payload (Sec. 6.2) runs the VDMc implementation and interacts with measurements, including the autopilot flight control commands. They are encapsulated as ROS topics. The VDMc has two main modes of operation: post-processing (debug) and real-time, as shown in Fig. 7.3. The

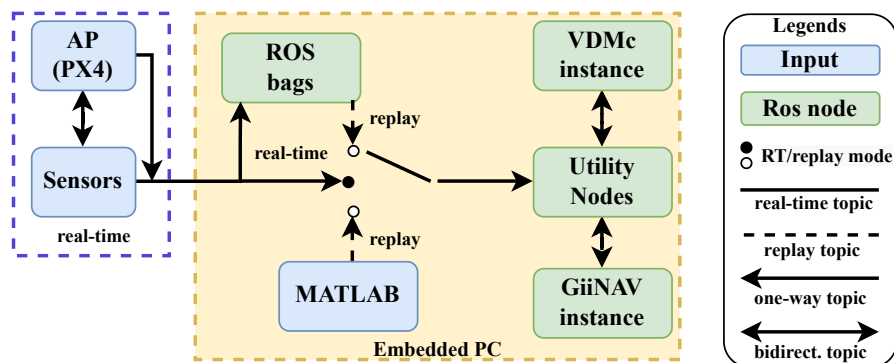


Figure 7.3: Modes of the VDM C-based operation

first mode allows the utilization of data from previous flights to be replayed in a debug mode. As only the very recent flights contain the log via ROS bags, a pipe line is created to interface older flights with sensor-internal storage and thus compare the C++ and MATLAB implementations of the VDM-based navigator. The second mode operates similarly, but in real time. In off-line mode, the software allows testing different configurations of the filter and sensors. For example, the output frequencies of sensor data can be modified, GNSS outages can be simulated, or a subset of filter states can be activated.

In online mode, the navigator is initiated when the UAV is in the air using an INS-based solution that is executed in parallel (GiiNav). As the availability of different sensor data

^{II}https://gitlab.epfl.ch/laupre/vdm_c/-/wikis/home

and autopilot commands is asynchronous and depends on communication delays and data parsing, a particular sensor data management is implemented to cope with such situations. The navigation solution is presented as a ROS (`vdmc_pose`) topic to the modified version of QGC, where it can be displayed together with the inertial-based (`GiNav`) and autopilot position estimates.

7.3.2 Architecture

This section introduces the general VDMc architecture with the I/O flow as depicted in Fig. 7.4. The C-class architecture is visualized in the Appendices D.3.1. The VDMc receives the initial

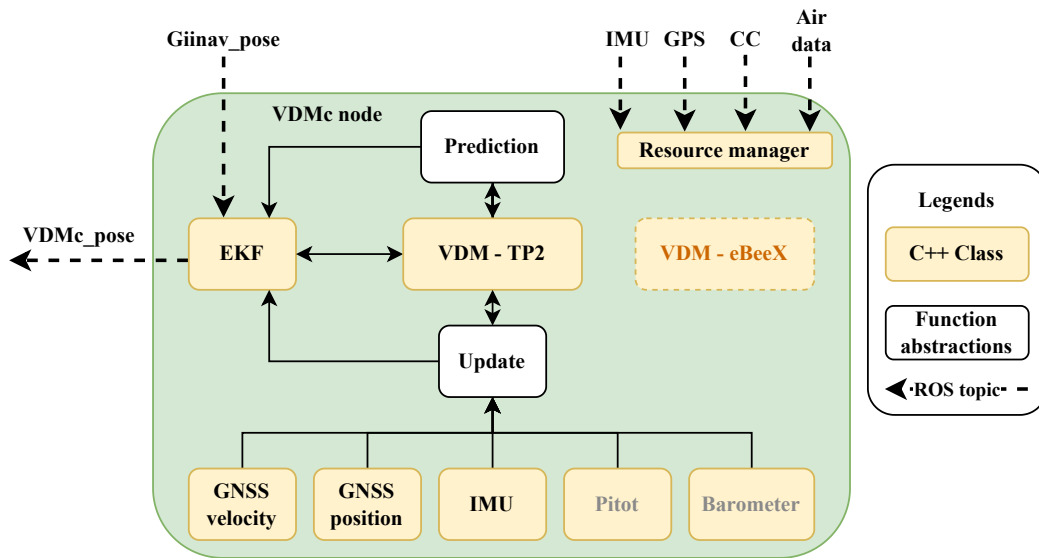


Figure 7.4: Simplified design of VDMc I/O

conditions of the navigation states via a ROS topic `giNav_pose`. Sensor data (for example, IMU measurements, GNSS position and velocity) and flight control commands are received asynchronously, as shown at the top of Fig. 7.4. These inputs are treated in real-time (Sec. 7.3.4). A platform-dependent vehicle dynamic model (currently implemented for TP2) predicts the plane's trajectory together with its uncertainty. The specific forces and angular velocities provided by an IMU are implemented as observations (which arrive at regular intervals), while the GNSS positions and velocities may be intermittent. Other measurements such as air speed and/or static air pressure can be easily added as an optional observation following a template C++, but are not used in the current real-time implementation. The internal status of the software is currently logged internally. A pipeline similar to that for the `vdmc_pose` topic is ready for the status forwarding to the QGC but the parsing of the status messages is not yet implemented to be displayed.

7.3.3 Automatic Linearization in C++

The definition of the platforms and the related model equations are now written using Mathematica [101] (Fig. 7.5) including, the auxiliary and sensor models. These are defined within a

```
Aerodynamic moment in body frame

Mb = ({
  {Mxb},
  {Myb},
  {Mzb}
});

dotVel = Cbl . fb + gl - SkewSymMat[omegaell] . vel - 2 SkewSymMat[omegaiel] . vel;
dotomegaibb = Inverse[Ib] . Mb - Inverse[Ib] . SkewSymMat[omegaibb] . Ib . omegaibb;
```

Figure 7.5: Definition of the velocity and angular rate differential equations (Eq. 3.57 and Eq. 3.49) using Mathematica environment

file dedicated to the particular platform model. A first python script (`generate_model_ini.py`) is called to populate the config file `model.ini` (Appendix D.3.4) specifying the state space, i.e. the auxiliary models as presented in Sec. 3.5.2. The different functions are then evaluated with Mathematica and linearization is performed when necessary. The output function files define the mathematical tools needed to perform the different Kalman operations. Then, two python scripts are run: (i) to convert these mathematical operations in C++ standard and complete different simplification to reduce the number of lines to be executed (`fixMathematicaOut_v2.ini`); (ii) for some functions, the output is transformed with the second script (`sparseMatrix.py`) to sparse matrices for optimization using the Eigen library [102]. The functions produced `.cpp` are ready to be used directly as input in the VDMc application functions. The linearization flow is depicted in Fig. 7.6. This allows automatizing the generation of the C++ function when some model modification is applied to the platform without changing the structure of the filter. The modified source files need to be considered before running the application. Complete Mathematica VDM of TP and the different sensor models are available in [100]. The C++ source file for the GNSS sensor is shown in Fig. 7.7, where the `include` macro imports the autogenerated functions. Additional sensors have to follow this simple template and is presented in Sec. D.3.5.

7.3.4 Asynchronous Observations Handler

The autopilot commands and sensors observations are presented through ROS topics to which VDMc subscribes. The availability of these data is principally asynchronous, and this fact needs to be handled in real-time. A naive implementation of the VDM would wait for the required data to perform the EKF updates (in the case of sensor data) or the state prediction (in the case of control command data). In such a situation, the EKF is forced to idle for a certain amount of time, which is undesirable if the navigation solution is used for real-time guidance and control (e.g., updating the navigation states of the autopilot). Similarly, if a sensor provides several

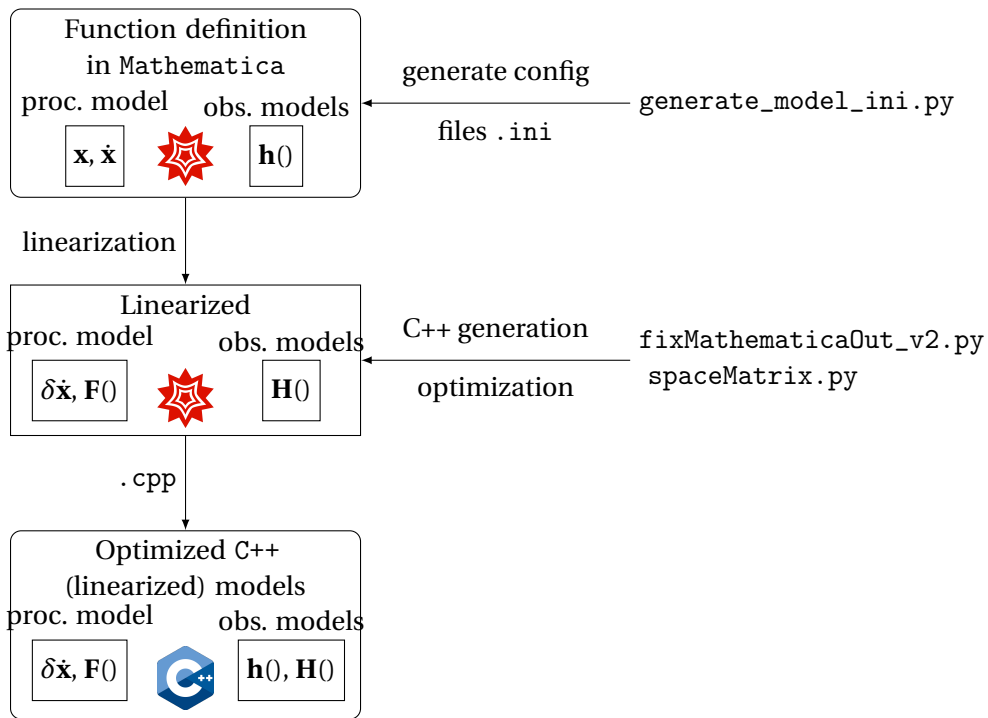


Figure 7.6: Linearization flowchart

observations in one data chunk (e.g., due to communication restrictions), the VDM-EKF would postpone its execution until the data becomes available.

VDM and its EKF are implemented with timers to ensure the continuous flow of the navigation solution. In addition, a backpropagation mechanism is implemented to deal with delayed observations. With these additions, the VDM-based navigator/estimator continues processing data as they arrive, performs its predictions regardless of current availability or absence of sensor observations (even if no more observations arrive!) and saves the current states and covariances when an update is expected to happen thanks to dedicated sensor timer. The delay pending until the filter has to wait for the expected data to arrive before saving the states (variable `saveStateDelay` in `filter.ini`), and the maximum time during the saved states and covariance are kept in the memory (variable `max_delay` in `filter.ini`), are variable to be set in the configuration file `filter.ini`. Fig. 7.8 shows a situation where several IMU measurements arrive as one packet (to limit overhead in communication). In this figure, the arrival of the data from the GNSS receiver (orange), the IMU (blue), and the autopilot control commands (black) is represented on a horizontal timeline with vertical arrows. In the upper part of Fig. 7.8(a), the current state and time of the filter is delineated with the dashed red vertical line, while the current time is displayed with the green dashed line. Between the current time and the last EKF time, two GNSS measurements are transmitted along with a control command from the autopilot. However, the packet of IMU data is still expected to arrive. When a delayed observation packet is detected, EKF returns to the closest previous state using the saved states and covariances to perform the updates with the newly arrived

```

#include "TP2/measprocessorGNSSpos.h"
#include "TP2/TP2_config.h"
#include <iostream>

using namespace TOPOPLANE2;
using namespace Eigen;

SparseMatrix<double> measprocessorGNSSpos::linH_func(const VectorXd &x)
{
    int _OFF = -1;
    SparseMatrix<double> out(3, nbStates);
    #include "../functionGenerator/H_of_GNSSpos.cppready"
    return out;
}

VectorXd measprocessorGNSSpos::h_func(const VectorXd &x)
{
    int _OFF = -1;
    VectorXd out(3);
    #include "../functionGenerator/h_GNSSpos.cppready"
    return out;
}

```

Figure 7.7: Include of automatically-generated functions for the GNSS position model `h_func` and Jacobian `linH_func` methods

observation(s) and all other observations received from this time to the current time (Fig.7.8(b) and 7.8(c)). The “nondelayed data” is stored in a circular buffer. Lst. 7.1 shows a simulated delayed IMU observation after 2 seconds of flight *STIM12* replayed via ROSbag. Related states are saved and later updated following the backpropagation mechanism.

Listing 7.1: Delayed IMU observation

```

20 INFO: Initializing the command control at time 378144.0000510000
21 WARNING: saving state at time 378146.3219932391 for timer IMU
22 WARNING: saving state at time 378146.3319932391 for timer IMU
23 INFO: backtracking from t=378146.3319932391 to t=378146.3219932391 for timer IMU

```

If the data arrive with a delay greater than the set (`max_delay`), or a delay greater than what the dedicated buffer size can store, they will be discarded for the sake of real-time consistency.

As another (unfortunate) example, `NavServer` was not configured correctly in flight *STIM8*, and GNSS messages with incorrect time tags (zero) were sent along with the correct ones. The VDMc software can reject all these wrong observations, as presented by Lst. 7.2.

Listing 7.2: Rejection of incorrect time-tagged GNSS observations

```

20 WARNING: received GNSS data for time 0.0000000000 but the filter is already at
    time 470183.7031287090
21 WARNING: The filter is already at time 470183.9031307390 but a timer requested
    time 0.0000000000
22 WARNING: received GNSS data for time 0.0000000000 but the filter is already at
    time 470183.9031307390

```

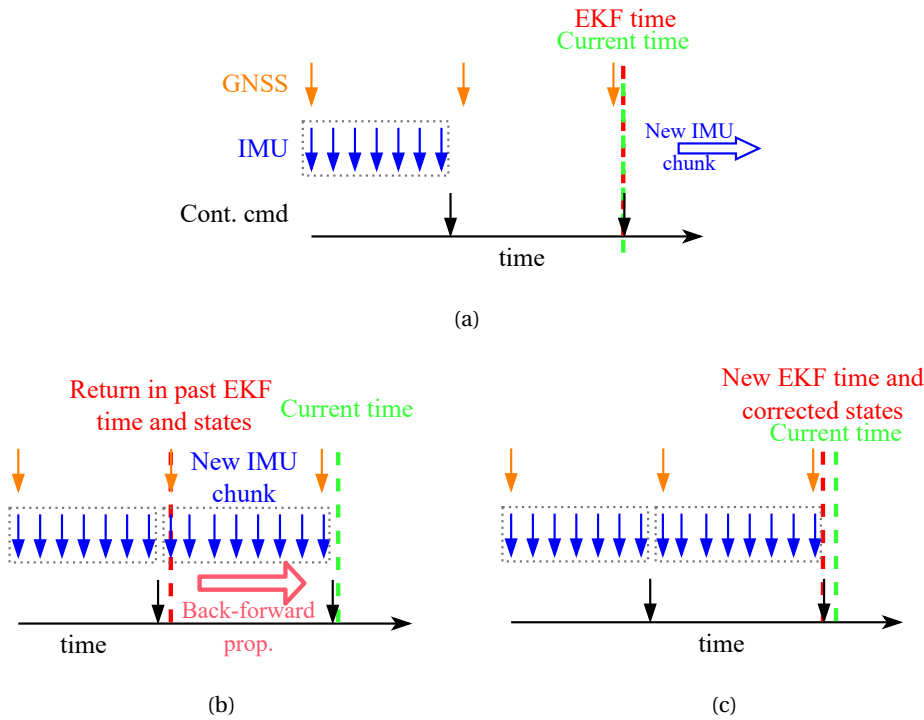


Figure 7.8: The EKF processing the GNSS and CC data as they arrive computing new states (a) returning in the past EKF states corresponding to the first new observation (b) and back-propagating the states with the new corrections (c)

```

23 | WARNING: The filter is already at time 470184.1131328990 but a timer requested
    | time 0.0000000000
24 | WARNING: received GNSS data for time 0.0000000000 but the filter is already at
    | time 470184.1131328990
25 | WARNING: The filter is already at time 470184.3131349500 but a timer requested
    | time 0.0000000000
26 | [...]
    
```

In case of multiple data messages transmitted simultaneously (to reduce the overhead in the communication channels), it is possible to specify the number of data messages that are expected to be received in one packet. The size of the packet should be set in the sensor .ini configuration file for the specific sensor. The timer mechanism and its implementation are explained in detail in [100], the main EKF loop algorithm, the related class descriptions, are given in Appendices D.3.2 and D.3.10, respectively.

7.3.5 Dynamic State Reduction

The current implementation of the filter includes a set of submodels (with varying state length), as presented in Sec. 3.5.2 the use of which can be decided at the execution time (model.ini).

This allows dynamic incorporation of states into the filter according to their prior knowledge (of flight phase) while considering others as known (constant) parameters. The dimension of the full state vector and the corresponding matrices inside the software is configurable by the user. The complete model and submodels are represented in Eq. 7.3 and correspond to the possible software configuration during the flight phases as shown in Fig. 7.9.

$$\begin{array}{c}
 \begin{array}{c} \xrightarrow{\text{Calib.}} \\ \left(\begin{array}{c} \mathbf{X}_n \\ \mathbf{X}_p \\ \mathbf{X}_a \\ \mathbf{X}_{a_d} \\ \mathbf{X}_w \\ \mathbf{X}_e \\ \mathbf{X}_{L_{imu}} \\ \mathbf{X}_{B_{imu}} \\ \mathbf{X}_{L_{gps}} \end{array} \right)_{68 \times 1} \end{array} \\
 \xrightarrow{\substack{\text{set as const.} \\ \text{or measured}}} \\
 \begin{array}{c} \left(\begin{array}{c} \mathbf{X}_n \\ \mathbf{X}_p \\ \mathbf{X}_a \\ \mathbf{X}_w \\ \mathbf{X}_e \end{array} \right)_{47 \times 1} \end{array} \\
 \xrightarrow{\substack{\text{pre-calib. } + \rho \\ \text{in-flight}}} \\
 \begin{array}{c} \left(\begin{array}{c} \mathbf{X}_n \\ \boldsymbol{\rho} \\ \mathbf{X}_a \\ \mathbf{X}_w \\ \mathbf{X}_e \end{array} \right)_{27 \times 1} \end{array} \\
 \xrightarrow{\substack{\text{RTL} \\ \text{outages}}} \\
 \begin{array}{c} \left(\begin{array}{c} \mathbf{X}_n \\ \mathbf{X}_a \\ \mathbf{X}_w \end{array} \right)_{20 \times 1} \end{array}
 \end{array} \quad (7.3)$$

where the navigation states \mathbf{x}_n are composed of position, velocity, attitude, and angular velocity; \mathbf{x}_p contains the VDM parameters (aerodynamic coefficients); \mathbf{x}_a and \mathbf{x}_{a_d} are the actuator states (propeller speed, aileron, elevator, and ruder deflections) and parameters describing their respective dynamic; \mathbf{x}_w is the wind vector; \mathbf{x}_e contains parameters modeling biases of accelerometers and gyroscopes within the IMU; $\mathbf{x}_{L_{imu}}$, $\mathbf{x}_{B_{imu}}$ and $\mathbf{x}_{L_{gps}}$ are the lever arm and bore-sights of the IMU and GNSS antenna relative to platform frame, respectively. When some parameters can be determined by other means (e.g., by direct observation or pre-calibration), the respective submodel is removed from the estimation (via modification of the `.ini` file). This reduces the filter size as highlighted in Eq. 7.3. For instance, when the IMU and GPS lever arms are measured, and the IMU-boresight and actuator dynamic are known, the filter size is reduced from 67 to 47 parameters. Furthermore, when the VDM parameters \mathbf{x}_p are precalibrated in a separate flight with precise GNSS positioning or refined during a time-limited calibration phase before the mission, they can be removed further from the filter. In this situation, only 27 parameters are kept in a state vector, considerably reducing the processing load (Sec 9.2.2). Estimation of some of these parameters can be further blocked during a detected outage (Fig. 7.9) using the proposed partial update method (Sec. 4.2).

7.4 Flights Phases

A normal UAV flight mission using VDM-based navigation goes through different flight phases presented in Fig. 7.9. The following subsection outlines the implemented configurations of the VDMc software.

Pre-flight calibration (optional) The pre-flight calibration procedure estimates the random biases in inertial sensors after the system switch-on. For accelerometers, this is achieved by parametric compensation when observing the gravity signal projected on the IMU

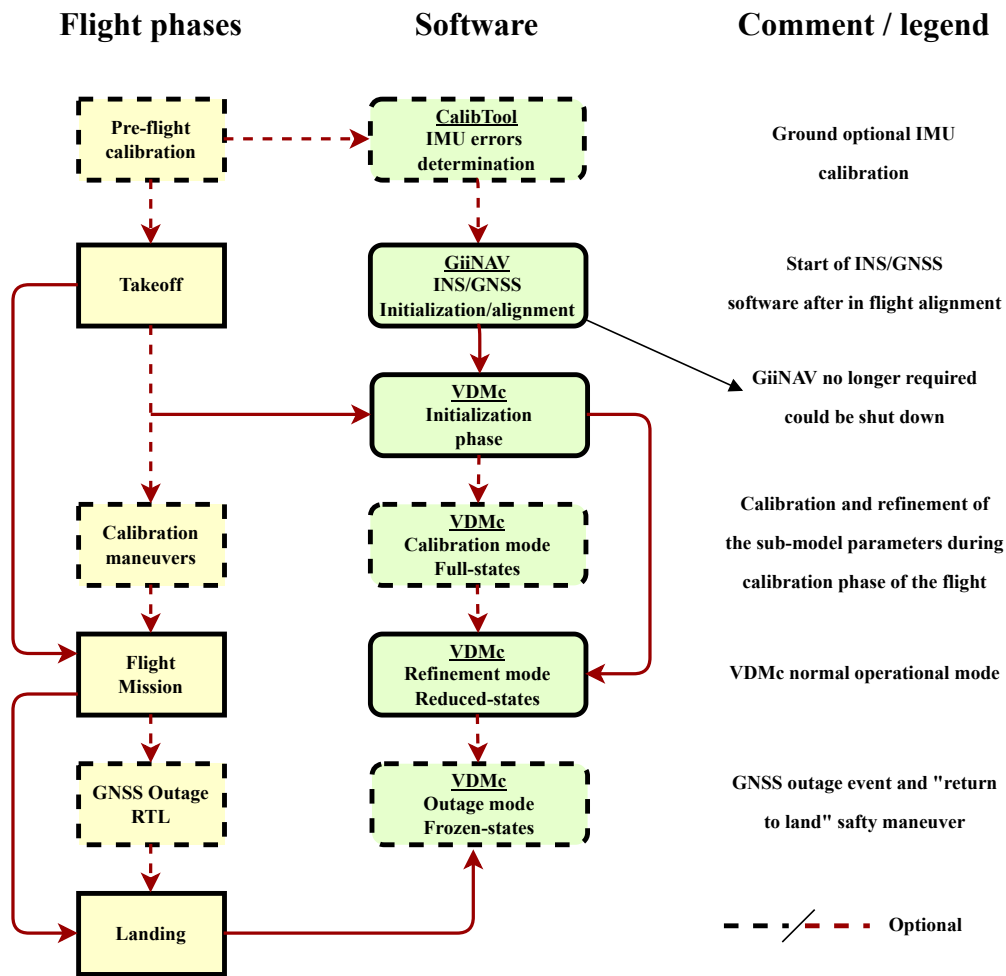


Figure 7.9: Flight phases and corresponding software configuration

axes at 6-distinct orientations of the UAV. For gyros with turn-on biases greater than zero, they are reset to zero. Estimated biases are subtracted from the IMU signals before further processing. The procedure is described in detail in [103]. This calibration is currently performed offline when data are replayed. Future developments can adapt the existing procedure to be performed in real-time. Barometer bias can be determined when the UAV is static on the ground and the altitude obtained from the GNSS receiver. Similarly, local air pressure can be used to calibrate the true airspeed computed from the dynamic pressure given from the pitot tube. These implementations are proposed with the specific RemoveBias ROS node as presented in Sec. 7.2 and the procedure is presented in Sec. D.3.3.

Takeoff - initialization A local-level strapdown navigation runs on the embedded PC, together with a 16-state extended Kalman Filter (*GiiNav* [99], adapted for the ROS environment). Once the pre-flight calibration is completed, the operator position the UAV against the wind (Fig. 7.10(a)). The two embedded navigation software applica-



(a)

(b)

```

12:04:07: Info (CMainThread::EKFPProc): EKF algorithm: not enough data. Retrying...
12:04:07: Info (CMainThread::EKFPProc): Initializing EKF algorithm...
12:04:07: Info (CMainThread::EKFPProc): EKF algorithm: not enough data. Retrying...
12:04:07: Info (CMainThread::EKFPProc): Initializing EKF algorithm...
12:04:07: Info (CMainThread::EKFPProc): EKF algorithm: not enough data. Retrying...
12:04:07: Info (CMainThread::EKFPProc): Initializing EKF algorithm...
12:04:07: Info (CMainThread::EKFPProc): EKF algorithm: initialization successful...
12:04:08: Info (CMainThread::EKFPProc): In-flight alignment completed...

```

(c)

```

=====
| VDMNav - connects to ROS topic, then Kalman Filter
| (c) Gabriel Laupré, Simon Gilgien, EPFL, 2020
| Inspired from ORIGINAL GiiNav-Linux-Version
| Version compiled on Jul 17 2022, 12:32:10
|=====
INFO (EKF): read and Parse the configuration files
INFO (EKF): init the filter
INFO (TP2): Platform created
INFO (EKFPThread): Waiting for GiiNav initial alignment...
INFO: Initializing the navigation at time 469898.0100000000
Initializing navigation states:
 0.8128075161
 0.1141702804
521.8330054632
-20.8234624246
-1.4019516904
-0.4637647989
 0.0015432798
-0.0182532524
-0.0126110413
-0.9997526688
 0.0000000000
 0.0000000000
 0.0000000000
INFO: Initializing the command control at time 469898.0026340000

```

(d)

Figure 7.10: (a) Launch of the *TP2* against the wind, (b) first second of the flight, and (c) *GiiNav* (d) *VDMc* initialization

tions (*GiiNav* and *VDMc*) are in standby mode. *GiiNav* waits for the right conditions to initiate its attitude. As the VDM-based navigation specifies the aerodynamic forces

and moments, it requires that the UAV is airborne. On the other hand, the quality of MEMS-IMU does not allow static initialization of attitude in heading with sufficient confidence. For that, forward GPS velocity is needed. Hence, *GiiNav* performs a coarse alignment and starts estimating the UAV's attitude, velocity, and position. The filter can cope with sizeable initial uncertainty of attitude in all axes. The *GiiNav* initialization is fast and happens during the first seconds on the flight (Fig. 7.10(b) and 7.10(c)). When the confidence of all navigation states (position, velocity, and attitude) is increased with time, the VDMc estimation starts by transfer-alignment (Fig. 7.10(d)). The traditional INS/GNSS filtering procedure is used during the initial and terminal phases of the flight is run parallel to VDM-based integration. Nevertheless, as soon as VDMc is initialized and runs, the *GiiNav* software could be stopped. Practically, it is kept running to compare the performance of both navigation software under GNSS outage later on.

In flight calibration (optional) The performance of VDM-based navigation depends on accurately determining aerodynamic coefficients that may not be completely known a priori. A procedure with high dynamic (mainly rotational) can be performed for their refinement (via state space augmentation) using all available observations (IMU, GNSS, Pitot tube). It has been demonstrated [14] that a combination of maneuvers is essential for that purpose and examples is given in Sec 8.1. Fig. 7.11 depicts in 2D and 3D the in-flight calibration maneuvers performed with the eBeeX (flight eBeeX_756). Different

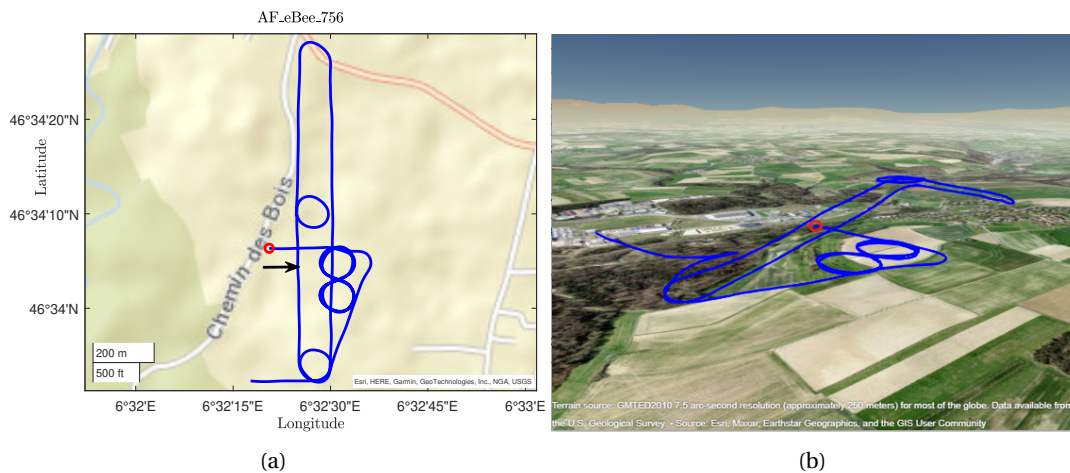


Figure 7.11: 2D (a) and 3D (b) dynamic maneuvers during the first 6 minutes of the flight ebeeX_756 for aerodynamic coefficient refinement

“flying patterns” (highlighted in Sec. 8.1) can be identified as leveled, ascending, and descending straight lines, "8-loop" and orbits. Further dynamics on trajectories used in this work can be observed at the beginning of the flights presented in Sec. 6.4. Once refined to sufficient confidence, they can be fixed and removed from the filter state space.

Flight mission In the normal operational mode, the UAV performs its mission (e.g. mapping)

with a sequence of long straight lines with cruising velocity and/or few altitude changes. This reduces the observability of several auxiliary parameters (states) as these are related to aerodynamic coefficients. Therefore, in this mode, a potential reduction in the number of states can be performed and, simultaneously, reduced the computational load of VDMc. Therefore, a reduced number of aerodynamic coefficients is kept as states in the filter. In the current implementation, all aerodynamic coefficients are removed from the estimated states after a variable period defined by the operator. The general idea is presented in Sec. 7.3.5. An automatic state removal decision based on their uncertainties is an interesting further development to be implemented for the real-time navigation software VDMc.

"Return-to-Land" (optional) In the event of GNSS outage, the UAV interrupts the mission and performs a failsafe procedure and possibly a Return to Land (RTL). In this case, the VDMc filter stops refining/estimating several auxiliary states related to aerodynamic coefficients (if still present in the state space) to estimate only the navigation states and the wind. And this until GNSS signals are again present. The mechanism of “considered” states is explained in Sec. 4.2. The automatic detection of GNSS outages is not implemented in VDMc. A detection possibility is to observe the GNSS observations uncertainty and receiver status, or declare an outage after the absence of GNSS solution for a predefined number of seconds. The navigation performance at this stage under GNSS outage with “considered states” is presented in the results Sec. 10.2.

Landing There is an automatic landing procedure on the PX4 autopilot, but for the fixed-wing platform *TP2*, manual (Fig 7.12) is preferable. Fully automated landing is mandatory in the eBeeX series. As long as the UAV is flying under normal (no stall) conditions, the



Figure 7.12: Successful manual landing on a wheat field in Echendans with the *TP2*

VDMc software provides navigation information to guide the UAV to its landing area.

Summary

This chapter has covered real-time aspects of the VDM-based navigation system. In particular, the accessibility of flight control is essential, as it is their correct time-stamping w.r.t. (external) sensor data. Alongside the autopilot-generated control commands, the data flow from/to the UAV, the GCS, and the embedded computer have been detailed. This description allowed the understanding of the communication channels in the overall system. Subsequently, the real-time applications in the embedded computer have been described. Particularly, the navigation software VDMc has been documented with its main components. Finally, the distinguished flight phases of a typical UAV mission have been described with their specific use and configurations of navigation software. As the experimental setup has been explained with its enhancements in the foregoing chapters, the following part of this thesis will evaluate the theoretical and engineering contributions: first, the validation of the calibration methodology will be discussed. Then, real-time investigations will be presented. Finally, the performances of VDM-based navigation will be analyzed under GNSS outages.

Results and Analysis **Part IV**

8 Determination of Coefficients

Overview

VDM-based navigation uses the UAV's aerodynamic model to derive the forces and moments driving the airborne physical behavior of the aircraft. These forces and moments, in turn, parameterize the navigation equations. The accuracy of the model-generated moments and forces depends on the models themselves and the aerodynamic coefficients that compose these equations. This chapter summarizes the results obtained with the proposed coefficient identification and refinement methodology presented in Chp. 5 for the two platforms. At first, the importance of flight dynamics in the estimation of coefficients using recorded sensor data and reference is highlighted through simulations. The necessity to have a coarse prior knowledge of the aerodynamic coefficients is emphasized: if a set of coefficients is not approximated well enough, the VDM-based navigation system can not work. This is due to challenges related to non-linear navigation equations and estimation techniques that may lead to local minima or even to filter divergence. A coarse set of VDM parameters is determined for the *TP2* and *eBeeX* drones with the three consecutive linear estimators of the WMF method - employing observability Grammian criteria. These coefficients are used to generate the linear and angular acceleration of the platform during validation flights and are compared to the IMU readings to acknowledge the correctness of the estimation procedure. Still, in an offline environment, the two refinement methods are proposed to obtain calibrated coefficients via optimal smoothing. Additionally, in the state-correlation matrix obtained after optimal smoothing, a set of highly correlated pairs of coefficients is identified. These pairs of coefficients are then lumped together. This strategy reduces the size of the state space, and the new reduced model is evaluated using different flights. Most of the results come from [14, 28].

Table 8.1: List of results for coefficient calibration

Method / Data	Simu.	TP2	eBeeX	Section
Dynamic and initialization importance	✓			8.1
Approximate coefficients				
RLS with Grammian		✓	✓	8.2
Refined coefficients				
Optimal smoother		✓		8.3.2
with attitude from photogrammetry		✓		8.3.1
'pose' sensor			✓	A.6

8.1 Effect of Flight Dynamics and Initialization

Monte Carlo simulations are performed to understand the impact of aircraft maneuvers on coefficient self-calibration (using only flight data) via state space augmentation. The simulations grasp the correlation of the aerodynamic coefficients among themselves and the other states.

The steps of the self-calibration methodology are shown in Fig. 8.1 and each block will be presented in the following sections. As the flying conditions during calibration can be chosen, GNSS observations are continuously available, and the wind is assumed to be zero or negligible. Therefore, the wind is not estimated to better isolate the influence of the trajectory dynamics in the coefficient determination without the influence of additional states related to the wind. The auxiliary states \mathbf{x}_e , \mathbf{x}_p , \mathbf{x}_a are added to the state space, as presented in Sec. 3.5.2. Their initial values and uncertainties are given later.

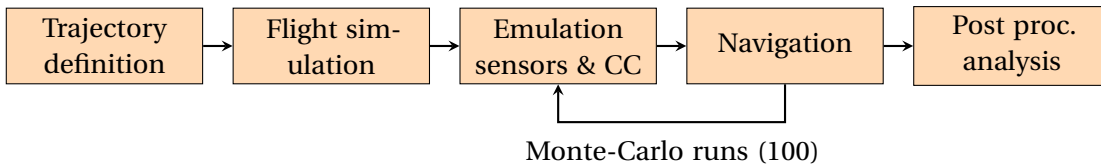


Figure 8.1: Simulation workflow in self-calibration

8.1.1 Trajectory Definition

Some aspects considered in the trajectory definition are execution time, the feasibility of maneuvers for the autopilot and / or the drone operator, and the continuity of GNSS signal tracking. Therefore, the designed trajectories last less than 5 minutes and avoid steep turns and attitude changes that could lead to a loss of GNSS signal reception in actual flights.

Seven trajectory segments are defined and separated into three categories. These are: straight line, circular orbit (loiter) and “figure-eight” (also called “infinite” loop). The last trajectory is a combination of several maneuvers. A straight line is characterized by two waypoints that

8.1 Effect of Flight Dynamics and Initialization

define its start and end. The actual path is not a straight line per se because the autopilot tries to reach the waypoints by compensations of different actuators. The resulting behavior depends on a particular control parameters, but generally resembles some oscillations around the straight segment, as presented in the first row of Tab. 8.2.

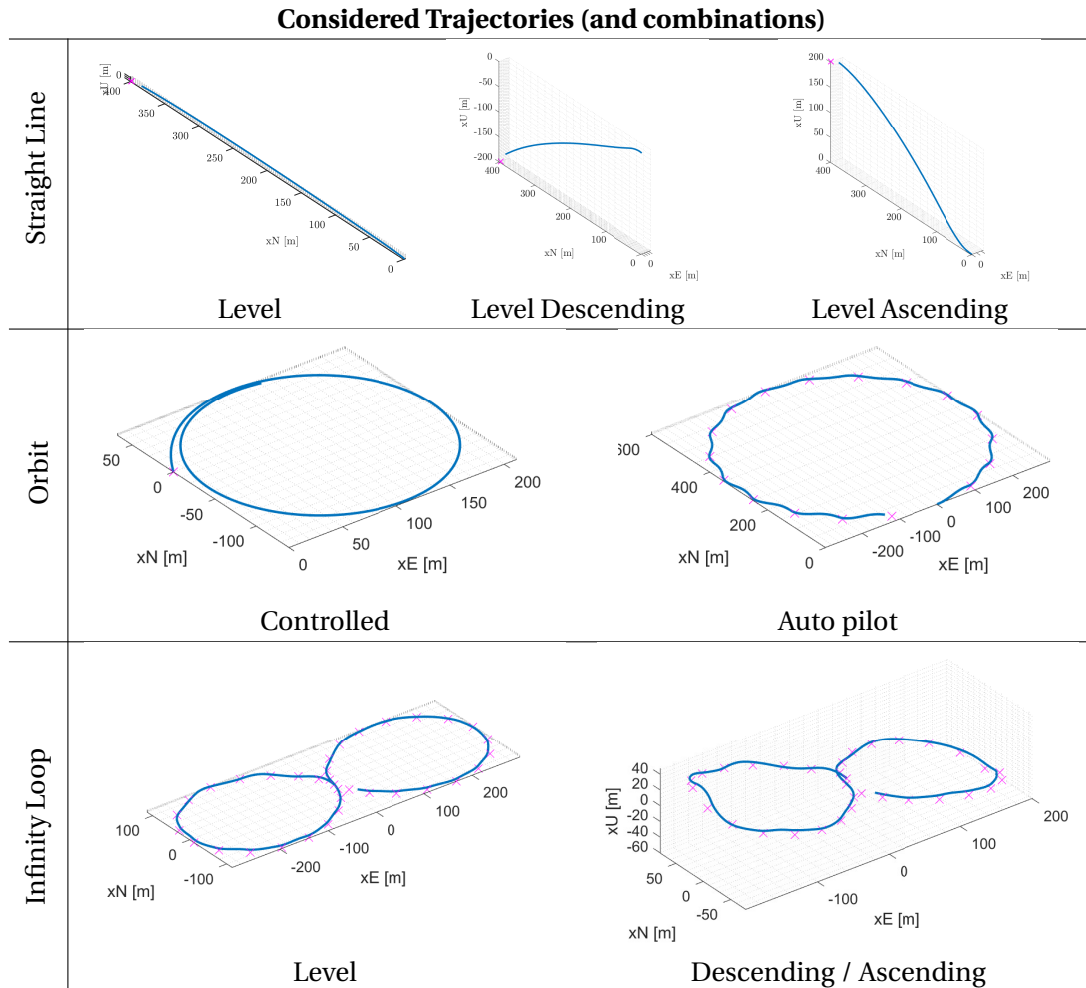


Table 8.2: Categorization of simulated trajectories into segments (7 types)

Two categories of orbits/loiters are considered as shown in the second row of Tab. 8.2: manual and autopilot controlled. The first is performed by setting the control surfaces to a constant value, while the second is executed by autopilot guidance following waypoints distributed along a circle. The infinity loop concatenates two complete orbits controlled by the autopilot in the opposite direction, with or without forced changes in altitude. These two segments are presented at the bottom of Tab. 8.2. The last type of trajectory is a concatenation of the segments mentioned above with the addition of velocity changes. The user-defined waypoints are represented with pink crosses on the different figures, and the axes (xE , xN and xU) belong to a local-level frame (ENU).

8.1.2 Flight Simulation

Once the shape of the trajectory is defined, the corresponding dynamic follows from the guidance and control that act on a particular platform. The implementation here is inspired by the *TP2* model presented in [51].

Within the simulation, the platform is initialized airborne with chosen heading and velocity at the first waypoint. The guidance uses waypoint coordinates expressed in a local-level frame relative to the first waypoint together with a desired velocity and attitude of the platform at each subsequent waypoint.

Waypoints are considered to be reached (cleared) when the platform is within a radius of 15 meters, and the next waypoint is activated. The guidance dictates the action of the actuators to be taken to direct the aircraft to the next waypoint. The controller generates the actuator states (i.e., the elevator, aileron and rudder angles, and propeller speed) and together with ideal VDM parameters, they define the nominal forces and moments following the model described in Sec. 3.5. The ideal reference trajectory follows from the rigid-body motion. Simultaneously, the output of the ideal sensors is generated at a desired frequency, that is, for the IMU, from the force and moment model given in Tab. 3.1 and from the derived states \mathbf{r}_e^l and \mathbf{v}_e^l as presented in Eq. 3.70 for the position and velocity of the GNSS.

The ideal trajectory is saved as a reference, while the generated sensors and actuator commands are used in the following steps. The generated control commands are delay-free, implying the absence of time-stamped data errors or delays of actuators when reaching the desired state. The influence of time-delay errors is presented in 9.1.1.

8.1.3 Sensor and VDM Parameters Errors

At this stage, the software has all the information to simulate the behavior of a platform following a defined trajectory and, most importantly, the sensor and actuator outputs at each discrete step.

To obtain a realistic stochastic model for the IMU errors, an internal identification is performed on one of the MEMS-IMU that is used in *TP2*, using the approach of Generalized Method of Wavelet Moments (GMWM) [104]. A summary of IMU error parameters from the GMWM analysis is provided in Tab. C.3.

GNSS position and velocity errors are considered to have a Gaussian distribution with $\sigma = 1\text{ m}$ and $\sigma = 0.03\text{ m/s}$ for each horizontal channel, respectively, and $\sigma = 2\text{ m}$ and $\sigma = 0.1\text{ m/s}$ for the vertical channel. The error in barometric altitude data is also considered white noise with $\sigma = 0.5\text{ m}$. This noise level is determined based on experimental data when the barometer output is compared with the post-processed (cm-level) GNSS position as a reference.

In the simulation environment, no residual imperfections related to the knowledge of sensor

position and alignment with respect to the body frame are assumed.

The initial uncertainty for the VDM parameters are fixed to **20%** of their reference values (1σ). Such uncertainty is reflected in the initial covariance matrix, while 100 Monte Carlo runs are simulated for each trajectory to diversify the initial error in parameter values. Similarly, the realization of stochastic processes in the simulated sensors (IMU, GNSS) is part of Monte Carlo simulation. Initial uncertainty for navigation states corresponds to the simulated sensor quality and is summarized in Tab. 8.3.

Table 8.3: Initial navigation state uncertainties

Navigation States		Values	Units
Position	- all axes	1	[m]
Velocity	- horizontal	1	[m]
	- vertical	0.5	[m]
Attitude	- roll/pitch	3	[deg]
	- yaw	5	[deg]
Angular rate	- all axes	1	[deg/s]

In the simulation environment, guidance and control are considered independent of navigation. In other words, the guidance is based on “error-free” sensors and VDM parameters, the reason for which the realized trajectory for each simulation may differ more from the ideal-desired trajectory. However, this fact is less important than the ability to examine the evolution of the parameter estimation compared to real model values. (The same simulated environment will be used in Sec. 9.1.1).

The flight dynamic influence on the estimation quality of VDM-parameters (aerodynamic coefficients) can be analyzed in terms of i) the remaining parameter errors and the reduction of the estimated parameter uncertainty, and ii) the residual correlations among the parameters among themselves and the other states.

8.1.4 Parameters Estimation per Segment

Dynamics in the trajectory increase the observability of different groups of parameters and improve their estimation by i) decreasing the variance and ii) decreasing their dependence on the auxiliary states. Tab. 8.4 accentuates this fact by showing the % of the remaining error in groups of VDM parameters after different types of maneuvers. The different groups are: thrust force F_T , forces along the x, y, and z body axes given by F_x , F_y and F_z , respectively, and the three moments M_x , M_y and M_z around the three body axes (as defined in Tab. 3.1). It is easily identifiable that better estimation is achieved with rotational dynamic, but each maneuver type influences different (groups of) coefficients. The lack of velocity changes in the suggested trajectories explains the relatively poor estimate of the thrust force group (F_T). Therefore, velocity variations should be considered during each maneuver.

Table 8.4: Group of VDM parameter error per trajectory

Trajectory	Average parameter error [in %] at the end of a maneuver per VDM category (lowest to largest)			Average
Desc. Straight L.	$F_y : 15.3 \%$	$< M_y < M_z < F_T < F_x < F_z <$	$M_x : 26.4 \%$	18.4 %
Straight line	$M_y : 10.0 \%$	$< F_y < F_T < F_z < M_x < M_z <$	$F_x : 17.6 \%$	13.8 %
Climb. Straight L.	$M_y : 9.1 \%$	$< F_y < F_T < F_z < M_x < M_z <$	$F_x : 17.4 \%$	13.7 %
Controlled Orbit	$M_y : 11.2 \%$	$< F_y < M_z < M_x < F_z < F_x <$	$F_T : 20.0 \%$	13.5 %
Orbit w. AutoPilot	$M_y : 10.8 \%$	$< F_y < M_x < M_z < F_z < F_x <$	$F_T : 17.9 \%$	13.2 %
8 Loop w. Alt. fix	$M_z : 8.3 \%$	$< M_x < F_y < M_y < F_z < F_x <$	$F_T : 17.5 \%$	11.1 %
8 Loop w. Alt. var.	$M_z : 8.3 \%$	$< M_x < M_y < F_y < F_z < F_x <$	$F_T : 16.8 \%$	10.9 %

Parameter Correlation

For a subset of trajectories, the correlation between all estimated states is shown on a gray scale of the correlation matrix in Fig. 8.2 at the end of the simulated segments.

In these figures, the diagonal elements represent the normalized variances of the estimated parameters, i.e., these always have value 1 and white color. The varying gray scale depicts the correlation coefficients (i.e., off-diagonal elements) from 0 to 1. As presented in Sec. 3.5.2 the state-vector \mathbf{x} is categorized into different groups, including: the 13 states of navigation error \mathbf{x}_n , the 26 states VDM parameters \mathbf{x}_p presented in Sec. 3.5 including the parameters $\mathbf{S}, \bar{c}, b, D$ and n which are considered a priori unknown, the time-correlated sensor errors \mathbf{x}_e that are the accelerometer and gyroscope biases for each axis (6 states). The estimation of time-correlated errors is the dominant part, even though other types of noise are added to the observations, and the 3 wind states \mathbf{x}_w are set to zero as no wind condition is assumed. The distinguishable “large square” in the middle of the correlation matrix corresponds to the VDM parameter cross-correlation among themselves.

The correlation matrices show that the directional dynamics of the trajectory increase the correlation among VDM parameters themselves and with the navigation states (initially set to zero). This is important as VDM parameters become observable through such a relationship when corrections to navigation states are made via GNSS and IMU update. However, at the same time, certain dynamics decrease the correlation of VDM parameters with respect to auxiliary states (i.e., other states than VDM coefficients). In other words, the group of VDM parameters stays correlated among themselves¹ but become less dependent to auxiliary quantities (e.g., the sensor errors) in the state vector. This is an essential prerequisite for their employment outside the calibration scheme. Tab. A.6 in the Appendix presents the VDM highly correlated (>90%) parameter pairs for a subset of trajectories. The correlation among VDM parameters will be further analyzed with real flight data and the use of an optimal smoother in Sec. 8.3.2

¹The correlation among them is related to model definition. In this particular case, the chosen polynomials are non-orthogonal, therefore, their terms remain correlated implicitly.

8.1 Effect of Flight Dynamics and Initialization

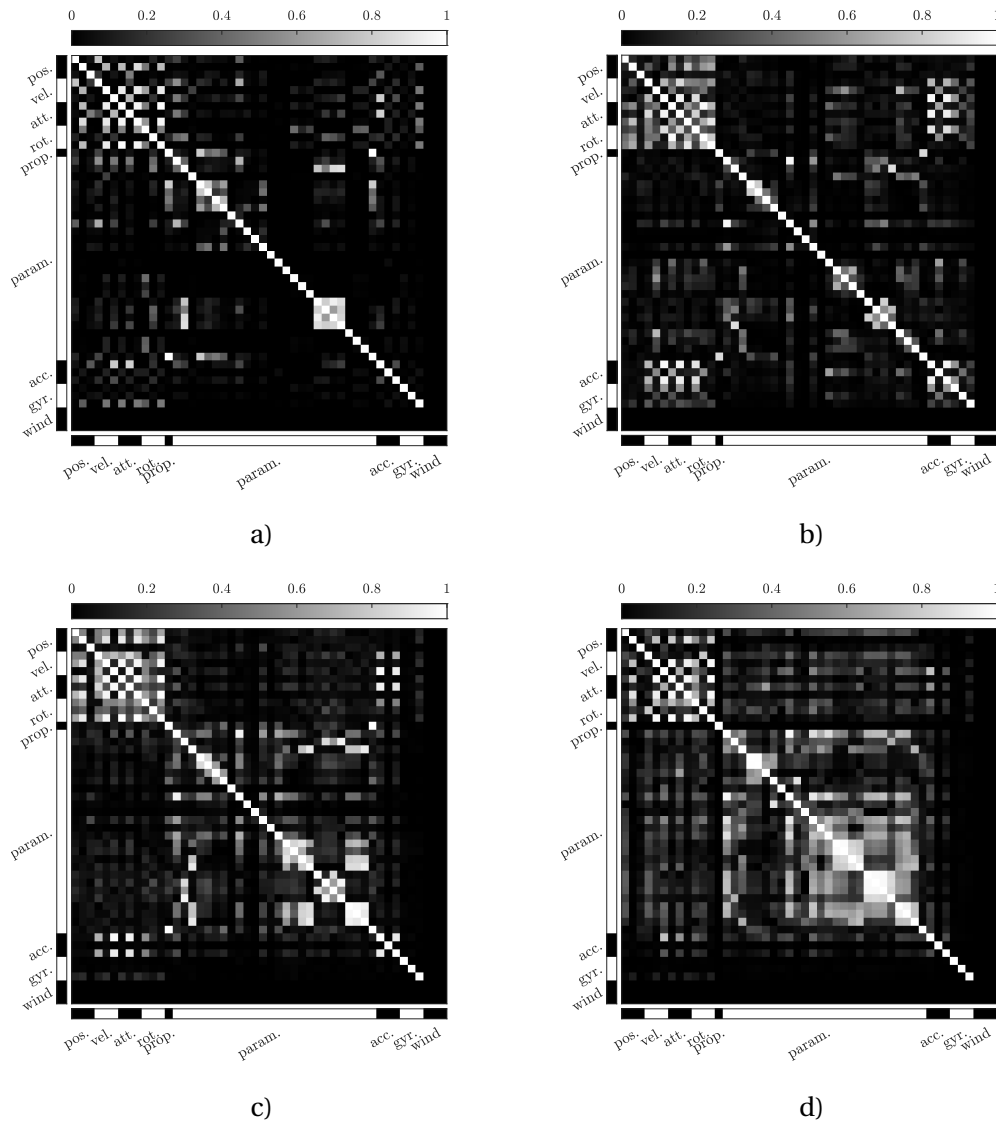


Figure 8.2: Correlation matrix P between state-vector elements at the end of a maneuver: a) - Level flight, b) - Controlled orbit c) - Infinity loop, d) - Combination of segments. The wind is not estimated and set to zero (no correlation)

8.1.5 Sequence of Maneuvers

The previous analysis indicated how different maneuvers contribute to the improved estimation of subgroups of the VDM parameters. The idea is therefore to combine the segments of different shapes into a “global calibration maneuver”. The proposed trajectory combines the trajectories above with accelerations and decelerations, as well as upward and downward sections. The trajectory is depicted in Fig. 8.3. The beginning of the trajectory, denoted with a red circle, starts with a straight climbing line, and the total flight time is around 3 minutes.

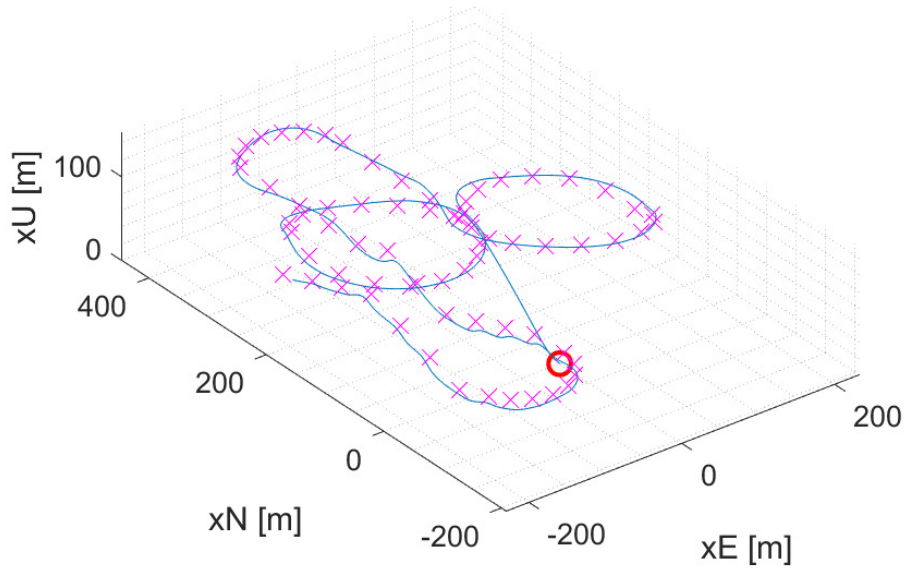


Figure 8.3: Combination of maneuvers. The resulting trajectory (blue) is designed with waypoints (pink crosses) and starts at the red circle

The effects of such compound maneuvers on the estimation of the VDM parameter $C_{F_x\alpha}$ are shown in Fig. 8.4. The uncertainty of this parameter represented by the estimated standard

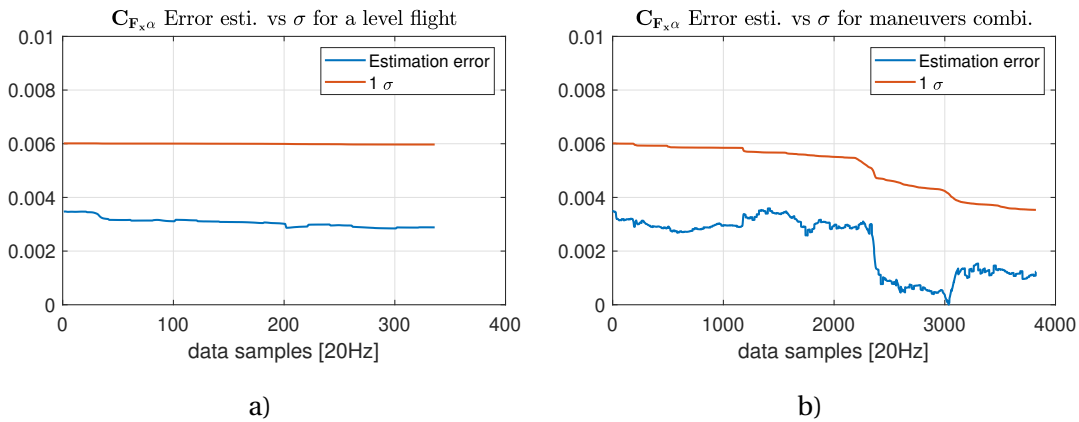


Figure 8.4: Error estimation evolution versus 1σ during a specific trajectory: a) - Level flight, b) - Combination of maneuvers

deviation (upper curve on both plots) is unchanged on the straight line (a), while reduced to 40% after maneuver sequencing (b). The latter case also removes about 50% of the parameter error (lower line), while the improvement of the former remains marginal. These investigations are generalized for every VDM parameter in Fig. 8.5. In this figure, the error for each VDM parameter is compared with the calibration result after a level flight in a constant direction for two types of sequences: an “infinity loop” at a constant altitude and the previously described global calibration maneuver. The latter decreases the uncertainties in all VDM coefficients.

8.1 Effect of Flight Dynamics and Initialization

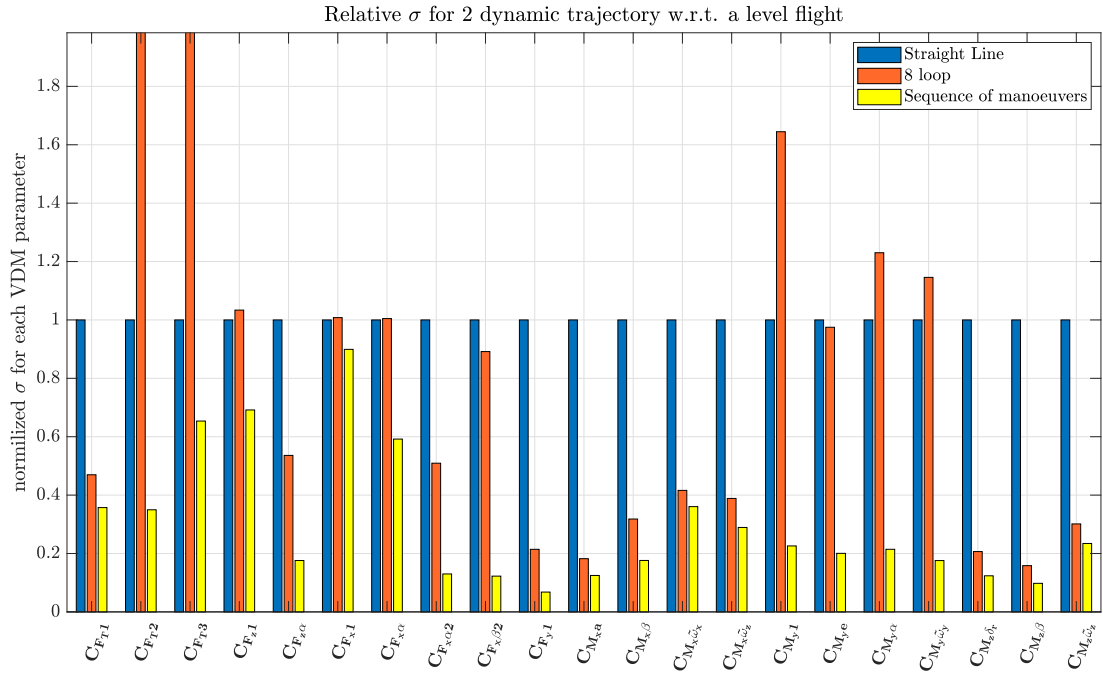


Figure 8.5: Estimation error of the 21 VDM parameters at the end of 3 calibration maneuvers

8.1.6 Effect of Initial VDM Parameter Values

To investigate the influence of incorrect initial VDM parameters, the simulated trajectories are used a second time with variation of the initial conditions. The methodology is shown in Fig. 8.6 where the first three steps are equivalent to those presented previously. Before starting

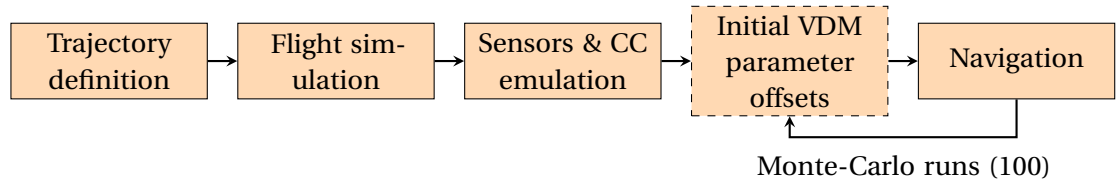


Figure 8.6: Methodology to test the influence of incorrect initial VDM parameters

navigation and estimation of the auxiliary states, the initial values of \mathbf{x}_p are randomly corrupted with a variable percentage of their correct values. The different “corruption” percentages are 10, 20, 30, 40, 50, 75 and 100% (1σ). Using Monte Carlo runs (100 for each percentage of corruption and each trajectory), each initial VDM parameter is corrupted with a different random value within the modified $\sigma\mathbf{x}_p$.

At the end of the 100 Monte Carlo runs, the average (residual) error in the estimation of the VDM parameters is inspected with respect to their correct value and the filter divergence based on successful runs can be observed as a function of initial errors. Individual cases are

Chapter 8. Determination of Coefficients

depicted in Fig. 8.7(a) for the complete range of σ and in Fig. 8.7(b) in detail on the first half.

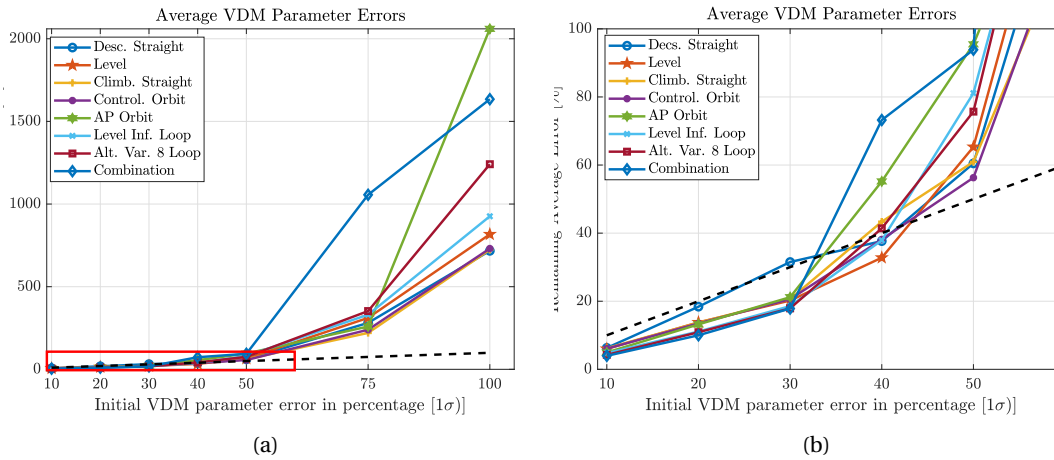


Figure 8.7: Average VDM parameter errors at the end of different trajectories after their varying initial corruptions (a) from 10 to 100% (b) zoomed from 10 to 50%

The dashed black line traces the situation where the initial and final % of errors remain the same to mark a threshold between improvement and divergence. When the initial error in \mathbf{x}_p is less than 40% of the parameter value, the descending straight line maneuver (light blue with circle markers) reveals low parameter observability due to the lack of dynamic, as the trend almost follows the dashed black line. For all the other trajectories, the dynamics allow reducing the VDM parameters error with respect to the initial values. It should be noted that the simulated trajectories are of a short duration and a further improvement in VDM parameter determination is expected with longer flight segments. For initial errors exceeding 40% of parameter values, the estimation starts to diverge. Above 50% the estimation diverges for all trajectories. The consequences of the wrong VDM parameters are multiple. Incorrect VDM parameters that produce inappropriate forces and moments are compensated with other auxiliary states, such as IMU bias and wrong navigation states (or wind when estimated) as presented in Sec. 8.3.1. In the event of GNSS outage, the solution becomes rapidly unusable and potentially worse than INS dead reckoning. Another issue with incorrect VDM parameters is the possibility of the numerical instability of the filter. Indeed, during Monte Carlo simulations, some of the realizations are affected by incorrect filter conditioning (*NaN* or *Inf* values in the states \mathbf{x} and/or in the covariance matrix \mathbf{P}). Tab. 8.5 summarizes the number of successful Monte Carlo runs.

Although 100 Monte Carlo runs is a rather small number to generalize the findings, a trend can be observed from Tab. 8.5. The segment resulting in the largest number of successful runs is colored green and the smallest is colored red for each percentage. Already from 20% of the initial error (1σ), the stability of the filter cannot always be guaranteed. The "controlled orbit" maneuver (light green) seems to be the most robust to initial error. A possible explanation is that during this maneuver, the control inputs (actuators) are fixed and thus avoid instability

8.2 Identification of Coefficients with Observability Criteria

Table 8.5: Number of successful navigation runs per initial error in VDM parameters per trajectories

Trajectory	Number of successful navigation runs (Monte-Carlo 100)						
	Initial VDM parameter error 1σ						
	10%	20%	30%	40%	50%	75%	100%
Descend. Straight L.	100	92	74	59	55	36	28
Straight line	100	98	71	56	53	36	26
Climbing Straight L.	100	98	87	70	60	40	31
Controlled Orbit	100	100	100	95	83	61	40
Orbit w. AutoPilot	100	99	95	79	57	31	19
8 Loop w. Alt. fix	100	100	99	86	73	43	31
8 Loop w. Alt. var.	100	100	98	79	66	41	21
Combinations	100	91	74	56	43	22	13

due to a filter incoherence between the flight control inputs and the moments produced using the corresponding actuator states \mathbf{x}_a .

These simulations indicate that it is not recommended to rely on a VDM-based navigation system with incorrect aerodynamic coefficients. The threshold at which the filter starts to diverge appears to be around 40% of VDM parameter values (for the *TP2* drone).

Flight dynamics (mainly rotational) is found to be essential for the observability of aerodynamic coefficients, that is, reducing their uncertainty, while alleviating the formation of correlation within groups of VDM parameters and accentuating their relationship with navigation error states. A trajectory that combines different maneuvers emphasizes dynamic variation, which helps to estimate aerodynamic coefficients. This is even more important if the aerodynamic coefficients are not known at all and are identified with a post-processing trajectory as proposed in Sec. 5.2 and practically confirmed hereafter.

8.2 Identification of Coefficients with Observability Criteria

Sec. 5.2 presented the methodology to identify the aerodynamic coefficients by decoupling the wind, the moments, and the forces estimation from the inertial and INS/GNSS solution of a calibration flight. To implement this proposal in practice, the payload "SODA-STIM" (Sec 6.2.3) is employed in *TP2* and the flights *STIM5*, *STIM6* and *STIM7* are used. For *eBeeX*, the payload "eBee-GECKO" and the flights *eBee_652* and *eBee_756* and for *eBeeX* are used. The experimental workflow is shown in Fig. 8.8 and the calibration details were explained in Sec. 5.2 and the defined terminology is used hereafter. Following the experimental workflow for *TP2*, flight data is collected, and decoupled aerodynamic estimation is performed on calibration flights. The proposed heuristic uniqueness of observability is presented. The quality metric associated with the calibration results is presented. The practical usage of these calibrated parameters in application flights is demonstrated. The entire methodology is repeated for a

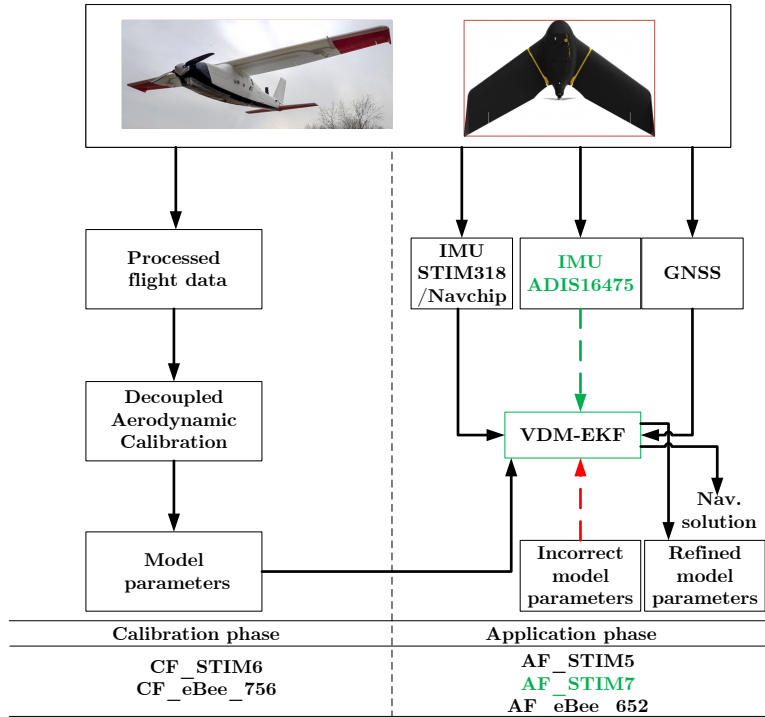


Figure 8.8: Experimental pipeline

delta-wing platform *eBeeX*.

8.2.1 Observability Grammian

Due to its simplicity, the empirical and graphical evolution of the observation Grammian in an experimental setting is exemplified in the wind estimator. The procedure detailed in Sec. 5.2.1 is implemented for the other two estimators (for moments and forces).

For the wind estimator, the dimensionality of the state space is four ($j = 4$) and $\mathbf{H}_k \in \mathbb{R}^{1 \times 4}$, whereas observability Grammian $\mathbf{W}_0(k) \in \mathbb{R}^{4 \times 4}$. Upon calculating the Eigen Values (EVA) and Eigen Vectors (EVe) of $\mathbf{W}_0(k)$ after all observations $k \in \{1, 2, \dots, K\}$, EVe of $\mathbf{W}_0(K)$ are chosen as the basis of a new state space. It should be noted that $\mathbf{W}_0(k)$ is symmetric and the EVe matrix \mathbf{V}_k is orthonormal, making it a rotation matrix in \mathbb{R}^4 . Subsequently, a closeness matrix (Γ_k) is computed as a dot product between \mathbf{V}_k and the base (\mathbf{V}_K). This is followed by the $\max(\Gamma_k)$ operation. This operation finds for each eigen vector of $\mathbf{W}_0(k)$ that is most closely aligned with each of the basis vectors. For the wind estimator, this closest Eigen vector is reported in Fig. 8.9 and the corresponding numerical value evolution of $\max(\Gamma_k)$ is shown in Fig. 8.10. As $\mathbf{V}_k^T \mathbf{V}_K = \mathbf{I}$ for $k = K$, the value of $\max(\Gamma_K) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$. This fact can be observed in Fig. 8.10, wherein all the trends eventually converge to unity. This means that at $k = K$, the first (second, etc.) basis vector is the same as the first (second, etc.) eigen vector of $\mathbf{W}_0(k)$. This can be seen in Fig. 8.9. The plot in Fig. 8.9 is a very special graphical representation where i) the

8.2 Identification of Coefficients with Observability Criteria

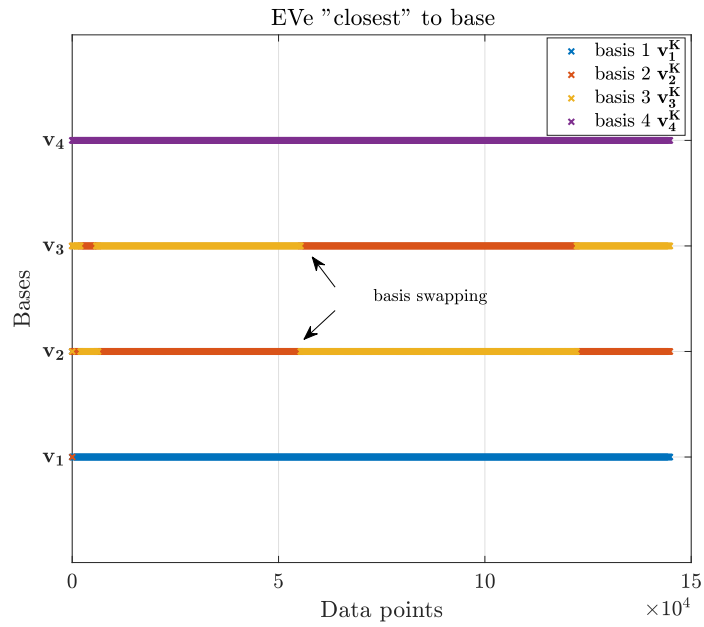


Figure 8.9: Tracking Eigen vector EVE of observability grammian

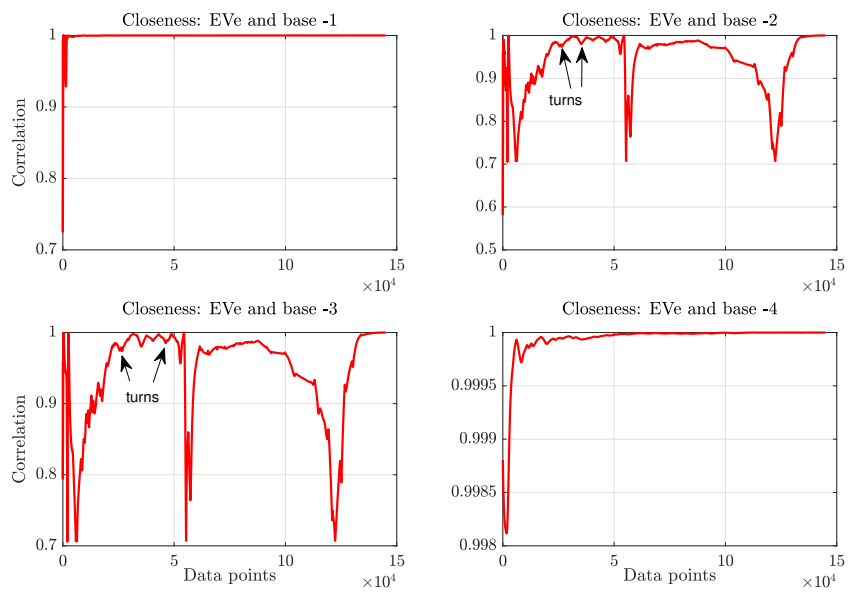


Figure 8.10: Maximum closeness in wind observability

Chapter 8. Determination of Coefficients

x axis $\in \{1, 2, \dots, K\}$ accounts for the discrete-time events at which airspeed measurements are made, and ii) the y axis $\in \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ represents the eigen vectors of $\mathbf{W}_0(k)$, each made up of one of the four colors (blue, red, yellow, purple). As discussed earlier, EVa and EVE of \mathbf{W}_0 are calculated at each discrete-time event, followed by establishing their association to the basis (using a closeness metric). In this graph, each basis vector is color-coded, and each \mathbf{v}_j is represented as a *quantized* time series of these colors. For example, the time series \mathbf{v}_1 mainly comprises basis vector 1 (blue) except for a few samples of basis vector 2 (red) at the beginning.

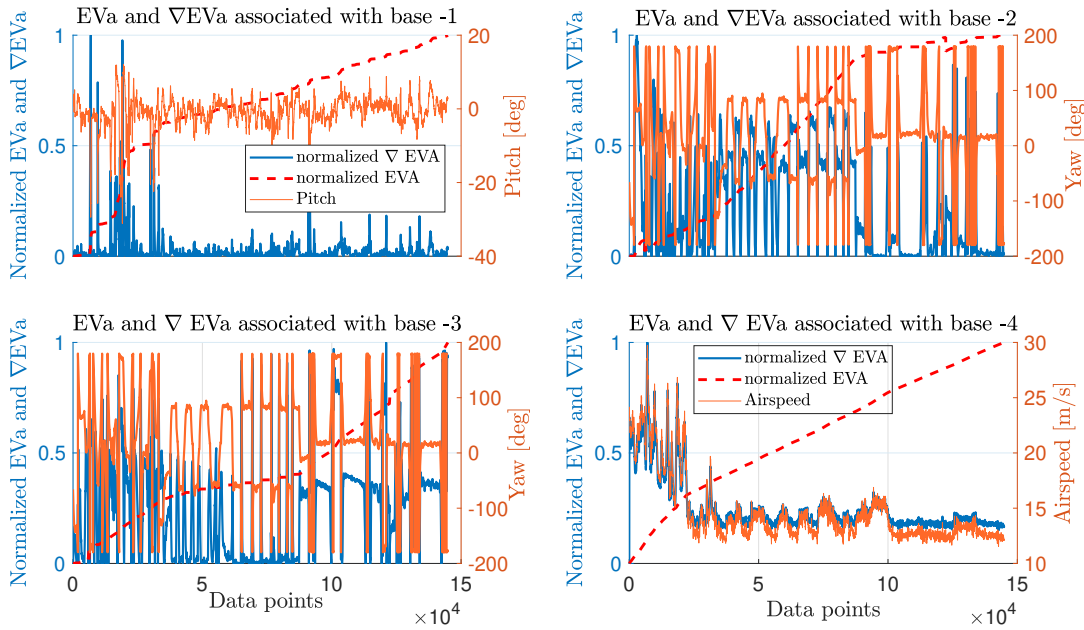


Figure 8.11: Evolution of Eigen Values associated to the basis

In this way, the eigen values of $\mathbf{W}_0(k) \forall k$ are approximately associated with the basis vectors. The evolution of these EVA (red dashed line) can be seen in Fig. 8.11, where they show a generally growing trend, except for certain minor variations in the basis vectors 2 and 3, corresponding to basis swapping.

In Fig. 8.11 (dashed red), the normalized incremental growth of EVA (Eq. 5.15) is presented. This incremental growth (IG) is then compared to drone's attitude and Pitot measurement, collectively shown using the right-hand axis of Fig. 8.11.

It can be observed that (i) the IG along basis vector 1 is correlated to pitch: The peaks (both positive and negative) in pitch lead to a non-zero IG along basis vector 1. However, this IG is mostly close to zero as the UAV flies in constant attitude; (ii) IG along basis vectors 2 and 3 are correlated to the UAV heading: The IG peaks along these basis vectors alternate with a change in heading. Moreover, these IGs show complementary trends, that is, a peak in one corresponds to a sharp valley in the other; (iii) IG along basis vector 4 is correlated to

8.2 Identification of Coefficients with Observability Criteria

Pitot measurements: this IG is always greater than zero. These observations are very much consistent with the findings of previous work [52], which state that i) UAV must change its heading for the horizontal wind to be observable, ii) pitching maneuvers are essential for better observability of down wind, and iii) the pitot scale factor is always observable as long as the UAV is airborne.

Therefore, following this methodology, it is approximately possible to track the growth of eigen values of observability Grammian, rather than evaluating these values only after the last observation as in [52]. It is the IG along basis vectors that is primarily responsible for observability. Therefore, a change of state space basis is implemented. For the wind estimator, it seems that i) basis vector 1 corresponds to down wind, ii) basis vectors 2 and 3 correspond to horizontal wind, and iii) basis vector 4 corresponds to pitot scale factor. Indeed, this interpretation is intuitive, thanks to the low-dimensionality of the state space; however, making such one-to-one correspondences for a high-dimensional state space may not always be possible (as in the case of moment and force parameters). Nevertheless, after each observation, the mathematics of the methodology handles the task of discerning most observable states from the rest. The same methodology is used to calculate the parameters of the aerodynamic model associated with moments and forces. An experimental comparison of the wind estimation using the mentioned method with respect to the local weather station is given in Appendix A.8.

Remarks: There exist certain observations that are forced to be rejected by the estimator due to their incompatibility. These observations arise due to: i) nonuniqueness of $\max(\Gamma_k) \rightarrow$; it is found that in some situations, two EVe are close to the same basis vector, leading to an eventual reduction in the dimensionality of tracked EVe space. ii) Transitioning from one closest Eigen vector to the other \rightarrow causes sharp peaks in IG, which do not necessarily correspond to observability. The cause of both these situations seems to be due to the empirical nature of the algorithm and presence of noise. However, such cases are easy to detect, from a software point of view, and are completely rejected by the estimator. Moreover, they collectively form less than 0.5% of the total number of observations for all three estimators.

8.2.2 Aerodynamic Model-Parameters

Following the partial update methodology presented in Sec. 4.2, wind, moment and force parameters using sensor data from the calibration flight are sequentially estimated. A smoother Infinite Impulse Response (IIR) with a cutoff frequency of $f_c = 30$ and 40 Hz (from visual data inspection using Fast Fourier Transform (FFT)) for gyroscopes and accelerometers is used, respectively, to eliminate high-frequency noise from inertial sensors caused by residual vibration in the UAV. The quality of the IMU data onboard (STIM318) is far superior to that of the autopilot (Tab. C.1).

IMU observations are compared with those obtained using calibrated VDM parameters. For this comparison, the focus is on forces, as it is implicitly dependent on moments and wind as a result of the cascaded estimation and computation of rigid body motion. Calculating forces

using accelerometer measurements is a straightforward operation that involves multiplication by mass of the drone. One of the estimated quantities, body force along the y axis is compared with that computed using IMU on Fig. 8.12(a). The figure shows that the force estimated by the model parameters follows trends to the one computed directly from IMU. Furthermore, the estimated forces appear to be a low-pass filtered version of the IMU data.

The other force and moment residuals for *TP2* and *eBeeX* are presented in Appendix A.9.

The entire calibration and application procedure is repeated for a delta-wing UAV *eBeeX*. However, repetitive details are skipped. Flight *eBeeX_757* is used to determine the coefficient with the payload presented in Appendix C.4, and flight *eBeeX_652* is used for validation.

The estimated body force along the x-axis is compared with the one computed using IMU. This comparison is presented in Fig. 8.12(b). As the force model of *eBeeX* contains a large number of parameters compared to *TP2*, the two signals match better and the residual is lower, confirming the correctness of model identification with respect to observations. However, this does not necessarily guarantee better navigation performance. This aspect will be discussed in more detail later (Sec. 10.1.2).

The determined coefficients are first validated in Sec. 10.1 using a validation flight (with GNSS outages) and then extensively used in Sec. 10.3 for performance comparison with inertial coasting for *TP2* and *eBeeX*.

8.2.3 Parameter Initialization and Uncertainty

All the estimation equations presented are purely recursive, needing an initial guess that can be computed with the observations. This is done by making use of observability Grammian and linear regression. The developed calibration chooses certain observations, equaling the number of states (10/11 for *TP2* for instance), corresponding to the highest incremental growth of the eigenvalue along each basis vector for running classical (weighted) linear regression:

$$\mathbf{x}_0 = \left[\underbrace{\mathcal{H}^T \mathbf{R}^{-1} \mathcal{H}}_{\mathbf{P}_0} \right]^{-1} \mathcal{H}^T \mathbf{R}^{-1} \mathcal{Z} \text{ with, } \mathcal{H} = \begin{bmatrix} \mathbf{H}_{e_1} \\ \vdots \\ \mathbf{H}_{e_n} \end{bmatrix}, \mathcal{Z} = \begin{bmatrix} \mathbf{z}_{e_1} \\ \vdots \\ \mathbf{z}_{e_n} \end{bmatrix}, \quad (8.1)$$

where the cardinality of $\{e_1, e_2, \dots, e_n\}$ is equal to the dimension of the state space. The subscript e_j denotes most of *exciting entity* for the basis vector j . An example of the points chosen for the moment estimator is shown in Fig. 8.13. The selected points are, for most, located at the beginning of a curve, which empirically validates the selection where the excitation of the system occurs during maneuvers. This provides an initial guess as to how to start the estimators. It should be noted that if the first n temporal observations are selected (assuming that n is sufficiently small as above), there is no guarantee that the regressor matrix will be full rank, thereby causing the entire framework to break. The calculation of \mathbf{R} is given

8.2 Identification of Coefficients with Observability Criteria

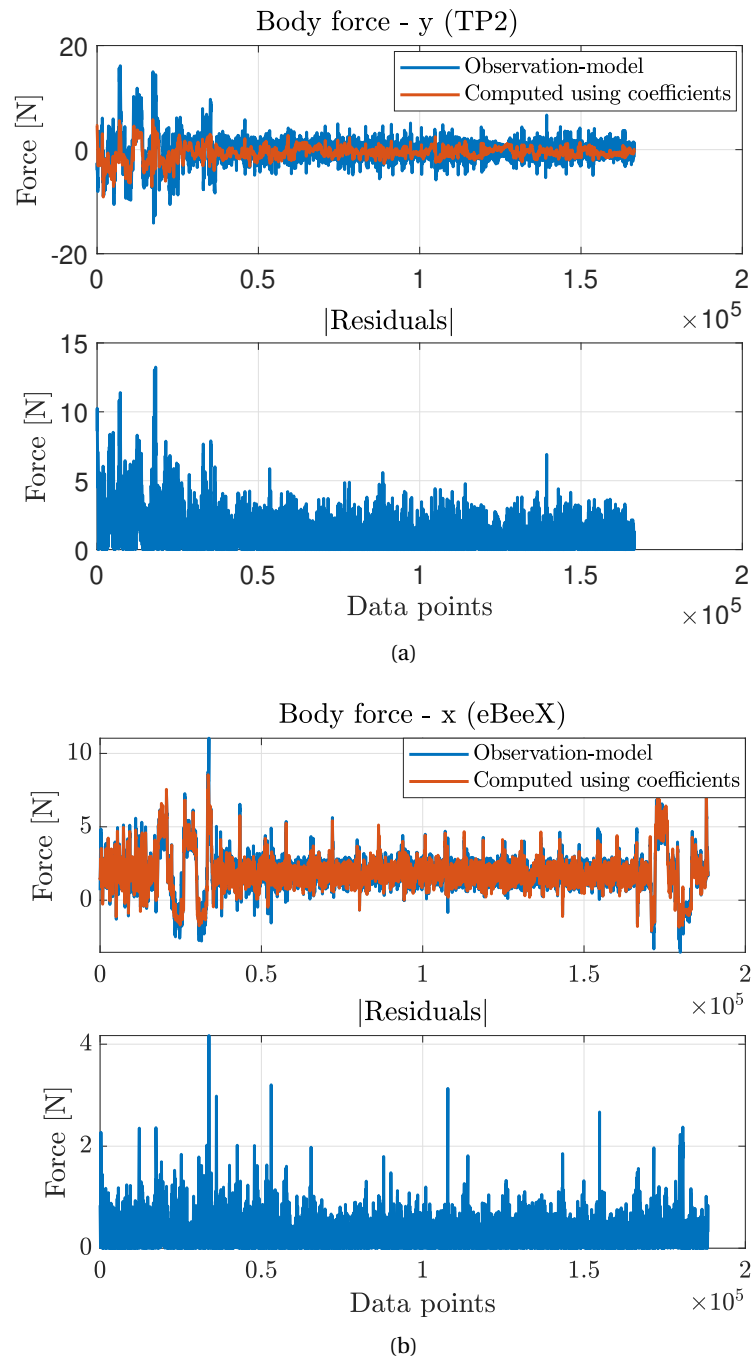


Figure 8.12: (top) Juxtaposition of force-y computed using IMU and calibrated model and their residuals (bottom) for *TP2* (a), and force-x for *eBeeX* (b)

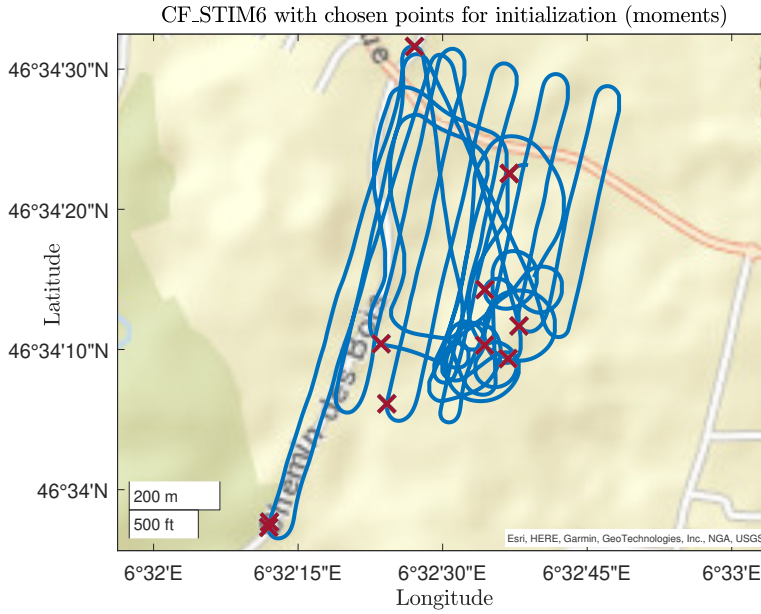


Figure 8.13: Initial chosen points \mathbf{x}_0 for the moment estimator

in the following.

The initial uncertainty \mathbf{P}_0 (1σ) for the wind is set to 0.3 m/s and 0.1 m/s for the horizontal and vertical components. The process noise \mathbf{Q} for the wind is experimentally set with relatively small values to limit the range of variations in the wind directions. For the scale factor γ , the process noise did not affect the convergence value, but only the fluctuations during the estimation and, therefore, is also set with a small value.

For the force RLS estimator, the initial uncertainty $\mathbf{P}_0 = \mathcal{H}^T \mathbf{R}^{-1} \mathcal{H}$. \mathcal{H} was defined in Eq. 8.1 and \mathbf{R} is the covariance of the accelerometer measurement, the strength assumed to be the same for each axis.

For the moment estimator, the initial covariance matrix \mathbf{P}_0 is calculated with respect to the gyroscope measurement noise (angular velocity) and the differentiated value (angular acceleration) $\dot{\boldsymbol{\omega}}_{ib}^b = (\boldsymbol{\omega}_{ib}^b(k) - \boldsymbol{\omega}_{ib}^b(k-1)) \frac{1}{\Delta t}$. Assuming that consecutive gyroscope measurement noises are *iid*, the uncertainty of $\dot{\boldsymbol{\omega}}_{ib}^b$ is given by uncertainty propagation $\sigma_{\dot{\boldsymbol{\omega}}}^2 = 2\sigma_{\boldsymbol{\omega}}^2 \frac{1}{\Delta t}^2 = 2\sigma_{\boldsymbol{\omega}}^2 f_s^2$ where f_s is the sampling frequency and $\sigma_{\boldsymbol{\omega}}^2$ the noise variance of the gyroscope measurement.

From a measurement moment \mathbf{z}_m given in Eq. 5.7 the Jacobian for uncertainty propagation

can be expressed as:

$$\partial \mathbf{z}_m = \frac{1}{\bar{q}S} \left\{ \mathbf{I}^b \partial \dot{\boldsymbol{\omega}}_{ib}^b + \partial \boldsymbol{\Omega}_{ib}^b \mathbf{I}^b \boldsymbol{\omega}_{ib}^b + \boldsymbol{\Omega}_{ib}^b \mathbf{I}^b \partial \boldsymbol{\omega}_{ib}^b \right\} \quad (8.2)$$

$$\partial \mathbf{z}_m = \frac{1}{\bar{q}S} \left\{ \mathbf{I}^b \partial \boldsymbol{\omega}_{ib}^b - [\mathbf{I}^b \boldsymbol{\omega}_{ib}^b]_{\times} \partial \boldsymbol{\omega}_{ib}^b + \boldsymbol{\Omega}_{ib}^b \mathbf{I}^b \partial \boldsymbol{\omega}_{ib}^b \right\} \quad (8.3)$$

$$\partial \mathbf{z}_m = \frac{1}{\bar{q}S} \left[\mathbf{I}^b \quad \boldsymbol{\Omega}_{ib}^b \mathbf{I}^b - [\mathbf{I}^b \boldsymbol{\omega}_{ib}^b]_{\times} \right] \cdot \begin{bmatrix} \partial \dot{\boldsymbol{\omega}}_{ib}^b \\ \partial \boldsymbol{\omega}_{ib}^b \end{bmatrix} = \mathbf{F} \begin{bmatrix} \partial \dot{\boldsymbol{\omega}}_{ib}^b \\ \partial \boldsymbol{\omega}_{ib}^b \end{bmatrix} \quad (8.4)$$

where $[\]_{\times}$ is the skew matrix operator (Eq. 3.13) and \mathbf{F} 's arguments are the different observations (wind from the previous estimator and gyroscope measurements). \mathbf{R} is the uncertainty of \mathbf{z}_m , and for each selected point e_i with $i \in [1, n]$

$$\mathbf{R}_s = \begin{bmatrix} \mathbf{R}(e_1) & 0 & \dots & 0 \\ 0 & \mathbf{R}(e_2) & \dots & 0 \\ 0 & 0 & \dots & \mathbf{R}(e_n) \end{bmatrix} \quad (8.5)$$

where each $R(e_i)$ is given by

$$\mathbf{R}(e_i) = \mathbf{F}^T(e_i) \begin{bmatrix} \sigma_{\dot{\boldsymbol{\omega}}}^2 & 0 \\ 0 & \sigma_{\boldsymbol{\omega}}^2 \end{bmatrix} \mathbf{F}(e_i) \quad (8.6)$$

This gives $\mathbf{P0} = \mathcal{H}^T \mathbf{R}_s^{-1} \mathcal{H}$ and \mathbf{x}_0 is computed as in Eq. 8.1.

With these initial states and uncertainty based on the observations, the three estimators are properly initialized, and the three cascaded estimators can be performed. Once a set of approximated coefficients is obtained, these can be further refined with the two complementary methods (Sec 8.3).

8.3 Refinement Methods

8.3.1 Attitude Update via Photogrammetry

By adding the aerodynamic coefficients as auxiliary states as presented in Sec. 3.5.2, one can take advantage of self-calibration by state space augmentation using precise observations. However, the initial states (VDM parameters) cannot be "completely" incorrect, otherwise the filter can diverge as presented in Sec. 8.1.6. Therefore, this method can only be applied when the aerodynamic coefficients are approximately known (either from a platform of a similar shape or using the method proposed in Sec. 8.2).

The flight *CF_i8* (with the payload "IGN-GECKO" Sec. 6.2.1) is exemplified by the use of photogrammetry to first derive a sporadic but precise attitude of the plane that is then applied to improve the estimation of the aerodynamic coefficients, as presented in Sec. 5.3.2. The flight characteristics are detailed in Tab. 8.6 while the trajectory is depicted in Fig. 6.9(a). In

Chapter 8. Determination of Coefficients

total, 440 images acquired during 26 flight lines at 2 flight levels (120 m and 150 m AGL) in a block geometry are considered. The longitudinal overlap is approximately 65% while the lateral overlap is roughly 45%. Bundle adjustment is performed with the Metashape software (previously PhotoScan) from AgiSoft ^{II}, using precise aerocontrol from INS/GNSS and about 20 GCPs. The image coordinates of signalized GCPs are obtained automatically by mask fitting with an accuracy of about 0.1 pixel.

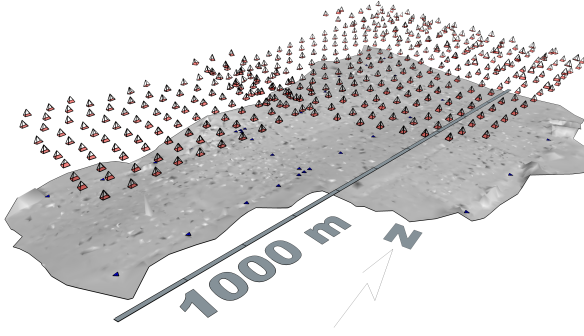


Figure 8.14: Camera orientation (triangles) and terrain. After [105]

Table 8.6: *CF_i8* flight details.

Flight name	<i>CF_i8</i>
Geometry	Block
# Images	440
# Flight lines	26
# Flight levels	2
GSD at 120/150m [mm]	24/30
Long. overlap [%]	65
Lat. overlap [%]	45
External control	GCP
No.	21
1σ (xy,z) [mm]	10, 15

Fig. 8.14 shows the camera orientations in red triangles at the time the image is taken, and the GCPs in blue triangles on the ground. The mean camera orientation uncertainties over all images after bundle adjustment are (3, 3, 1.4) *arcsec* (i.e., ≈ 0.002 deg) for the angle around the x_c (ω), y_c (ϕ), and z_c (κ) axis, respectively. In total, more than 400 oriented images were used to obtain absolute attitude measurements. The INS/GNSS integrated trajectory is used as an approximate pose for the camera before the bundle adjustment. These are also used to initialize the VDM-based estimator within the calibration flight.

Discussion

The initial uncertainties of the VDM parameters \mathbf{x}_p are set to 2% of their values. They are listed in the 1st and 5th columns of Tab. 8.7 for forces C_F and moments C_M , respectively. While highlighting the largest changes in color within Tab. 8.7 (sign changes in green and large differences in magnitude in red), the difference between the pairs of estimated VDM parameters is further expressed in percentage $\Delta_{\%}$ as

$$\Delta_{\%} = \frac{|\mathbf{X}_{w_c} - \mathbf{X}_{w/o_c}|}{\min(\mathbf{X}_{w_c}, \mathbf{X}_{w/o_c})} \times 100 \quad (8.7)$$

where \mathbf{X}_{w_c} and \mathbf{X}_{w/o_c} are the states estimated with and without the addition of camera attitude reference, respectively. These relative discrepancies are depicted in Fig. 8.15. Although some coefficients change little after convergence for both methods, others change remarkably.

^{II}AgiSoft PhotoScan Professional (Version 1.2.6) (Software). (2016 *) Retained from <http://www.agisoft.com/downloads/installer/>

Table 8.7: Initial and estimated (filtered) VDM parameters without (GNSS) and with photogrammetry (GNSS + Att)

Forces	Init.	GNSS	GNSS+Att	Moments	Init.	GNSS	GNSS+Att
C_{FT_1}	0.0026	0.0030	0.0035	C_{Mx_α}	-0.00234	-0.012	-0.012
C_{FT_2}	-0.05	-0.0326	-0.0601	C_{Mx_β}	0.0025	0.0058	0.0061
C_{FT_3}	2.23	4.37	4.27	$C_{Mx_{\omega_x}}$	-0.0465	-0.161	-0.1536
C_{Fz_1}	-0.125	-0.049	-0.091	$C_{Mx_{\omega_z}}$	-0.0219	0.0195	0.0161
C_{Fz_α}	-4.76	-18.7	-18.7	C_{My_1}	0.0215	-0.026	-0.026
C_{Fx_1}	-0.205	-0.485	-0.386	C_{My_e}	0.27	0.370	0.369
C_{Fx_α}	-0.728	0.507	1.451	C_{My_α}	-0.657	-1.150	-1.121
$C_{Fx_{\alpha_2}}$	-0.0538	-0.0551	-0.0537	$C_{My_{\omega_y}}$	-9.16	-17.93	-18.96
$C_{Fx_{\beta_2}}$	0.568	1.06	0.89	C_{Mz_r}	-0.0137	0.0007	0.0008
C_{Fy_1}	-0.127	-0.321	-0.272	C_{Mz_β}	0.0002	0.0012	0.0011
				$C_{Mz_{\omega_z}}$	-0.128	-0.0192	-0.0191

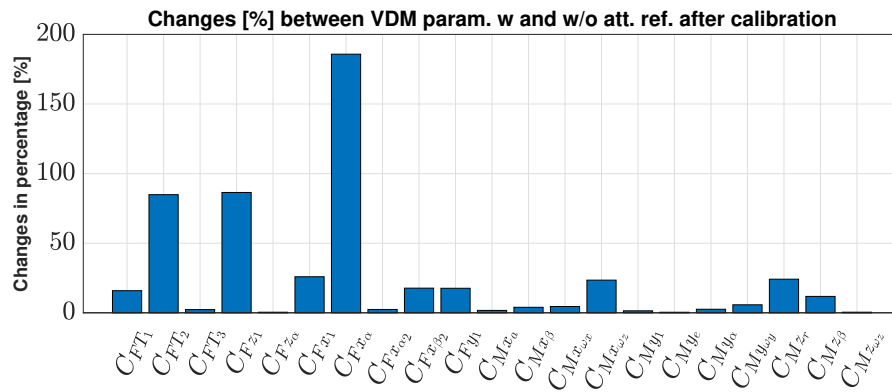


Figure 8.15: Percentage of change for each VDM parameter with and without the use of attitude reference measurements

In particular, C_{FT_2} , C_{Fz_1} and C_{Fx_α} converge to considerably different values (in magnitude) when attitude references are used (represented in red in Tab. 8.7). Furthermore, the sign of estimated C_{Fx_α} is changed. Some other parameters also change their signs. These are printed in green within Tab. 8.7. Note that, for some parameters, a sign change occurred after convergence irrespective of the use of attitude updates, which highlights the coarse approximation of their initial values. In contrast, the addition of precise attitude observations influences the estimation of the force coefficients more. This observation is counter-intuitive but is not explored further as the coefficients are evaluated via navigation and they are not evaluated individually.

The discussion of the influence of photogrammetry on the estimation of the errors (\mathbf{x}_e) and the wind (\mathbf{x}_w) states is discussed in [28]. It is shown that due to the constant airspeed velocity during a photo-flight, attitude aiding helps to decorrelate the inertial errors from the wind estimation, which improves the determination of the VDM parameters \mathbf{x}_p .

Chapter 8. Determination of Coefficients

The improvement in autonomous navigation using precise attitude updates is presented in Sec. A.4.

8.3.2 Optimal Smoother

Similarly to Sec 8.3.1, the flight CF_i8 , is used as the calibration flight with the differential carrier-phase GNSS (PPK). The accuracy of position and velocity updates is essential for estimating auxiliary states related to aerodynamic parameters. Considered time-invariant within the flight, their best estimate is obtained via an optimal (forward-backward) smoother.

It should be stressed again that the observability of parameters depends on maneuvers (as was explored in Sec. 8.1) and some dynamic maneuvers are part of this flight. The use of the optimal smoother further accentuates the existing structural correlations between the aerodynamic coefficients due to the model (Sec. 3.5) while de-correlating them with other states as depicted in Fig. 8.16(b) and 8.16(c).

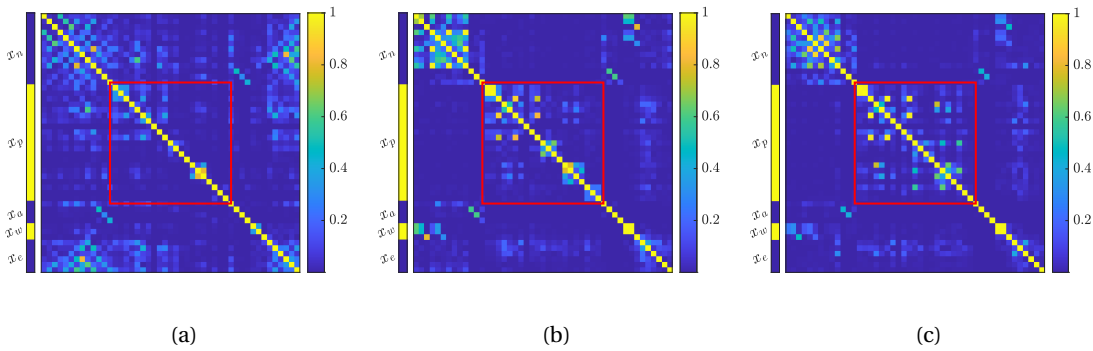


Figure 8.16: State correlation matrix (CF_i8 with highlighted \mathbf{x}_p after a) 30 s of filtering; b) at the end of forward-filtering; and c) optimal smoothing

The coefficients obtained by optimal smoother are given in Tab. 8.9 and compared with those obtained by the methods WMF and with attitude observation from photogrammetry (filter).

8.3.3 Parameter Reduction

The proposed approach described in Sec. 4.3 is to reduce the number of states by finding a linear relation between highly correlated aerodynamic coefficients. Their relations are obtained by analyzing the corresponding sub-block of the covariance matrix after smoothing (\mathbf{P}_{sm}) the calibration flight CF_i8 . As depicted in Fig. 8.16(c), the parameters outside the main diagonal in yellow are correlated by more than 90 %. Five highly correlated pairs are selected for regression analysis. The resulting linear relations between the selected pairs are detailed in Tab. 8.8. As the force parameter C_{F_y1} is correlated to C_{F_T3} as well as to C_{F_x1} , the model is reduced by four coefficients.

Table 8.8: Proposed correlated pairs for model reduction and their linear relations

Param. pair	Correlation	C_j	s_{ij}	C_i	o_{ij}
$C_{FT1} - C_{FT2}$	0.97	$C_{FT2} =$	$-40.9154 \times$	C_{FT1}	-0.202
$C_{FT3} - C_{Fx1}$	0.98	$C_{Fx1} =$	$0.0564 \times$	C_{FT3}	-0.412
$C_{Fy1} - C_{FT3}$	0.98	$C_{Fy1} =$	$-0.0006 \times$	C_{FT3}	-0.247
$C_{Fx1} - C_{Fy1}$	0.99	$C_{Fx1} =$	$0.0336 \times$	C_{Fy1}	-0.259
$C_{My1} - C_{My\alpha}$	0.85	$C_{My\alpha} =$	$0.820 \times$	C_{My1}	-1.552

The correlation pairs are quite different from those obtained from the simulations of Sec. 8.1.4 and summarized in Tab. A.6 where the correlations are more accentuated for moment-related parameters. The remaining correlation using real flight data is more pronounced for the force-related coefficients. The pair $C_{My1} - C_{My\alpha}$ is obtained in both scenarios (with a correlation of 97.5% in the simulation). Geometric parameters are not included in the auxiliary state vectors \mathbf{x}_p , which removes highly correlated relations with the coefficients. The use of the experimental refinement via optimal smoother indicates a low correlation among the coefficients compared to those obtained with the simulations (forward filter). The performance of autonomous navigation with the reduced model is described in Sec. 10.4.

8.4 Numerical Comparison of Coefficients

The different sets of coefficients obtained with the WMF method and the refinement using augmented state space with smoother and attitude observations from photogrammetry for *TP2* are summarized in Tab. 8.9. The coefficients for the *eBeeX* are given in Tab. C.5 in the Appendix.

 Table 8.9: Coefficients obtained from WMF (Sec. 8.2), attitude observation (Sec. 8.3.1) and smoother (Sec. 8.3.2) methods for the flight *CF_i8*

Forces	Refined			Moments	Refined		
	Approx. WMF	Att. Obs.	smoother		Approx. WMF	Att. Obs.	smoother
C_{FT1}	0.0002	0.0035	0.0163	$C_{Mx\alpha}$	-0.0053	-0.012	-0.015
C_{FT2}	-0.023	-0.0601	-0.867	$C_{Mx\beta}$	-0.0042	0.0061	-0.0113
C_{FT3}	58.61	4.27	10.4	$C_{Mx\omega_x}$	0.0312	-0.1536	-0.132
C_{Fz1}	-0.424	-0.091	0.2476	$C_{Mx\omega_z}$	0.0016	0.0161	0.0446
$C_{Fz\alpha}$	-6.149	-18.7	-23.99	C_{My1}	-0.0005	-0.026	-0.0012
C_{Fx1}	4.44	-0.386	-0.179	C_{Mye}	0.0601	0.369	0.321
$C_{Fx\alpha}$	0.0306	1.451	-2.703	$C_{My\alpha}$	-0.0372	-1.121	-1.53
$C_{Fx\alpha_2}$	-3.120	-0.0537	-0.0553	$C_{My\omega_y}$	-0.3829	-18.96	-7.713
$C_{Fx\beta_2}$	0.4542	0.89	0.258	C_{Mzr}	0.0050	0.0008	0.0019
C_{Fy1}	-4.728	-0.272	-0.251	$C_{Mz\beta}$	0.0094	0.0011	0.0024
				$C_{Mz\omega_z}$	0.0021	-0.0191	-0.0192

The coefficients obtained with the two refinement methods agree more in terms of sign and

Chapter 8. Determination of Coefficients

magnitude together than with the WMF method, and similarities can be observed. However, it is complex to evaluate each coefficient individually, as the polynomial character of the models implicitly correlates the group of coefficients together.

The **order of magnitude** of the uncertainty (σ^2) of the coefficients obtained with the refinement and the WMF methods is $< 1e^{-5}$ than their respective convergence value, for all coefficients related to moment and force. As summarized previously in Sec. 8.3.1, the compensation of the erroneous coefficients with the wind and IMU error estimation highlights that the computed uncertainties represent the accuracy of the whole state space estimation and not the uncertainty of the coefficients only [76]. In addition, high-frequency IMU observations that update the coefficients lead to a rapid steady state and decrease in uncertainty, even though they are potentially in a local minimum due to a lack of observability. The partial-Schmid update slows this convergence and might avoid setting a local minimum. This aspect at initialization is presented in Sec. 9.3.2.

The application of these coefficients is presented in Chp. 10 where they are evaluated using autonomous navigation and simulated GNSS outage. Although the coefficients obtained after refinement should theoretically better match the real values, (i) the payload "IGN-GECKO" is used for photogrammetry, (ii) *TP2* underwent some changes for the "SODA-STIM" flights (WMF and real-time experiments), and (iii) the models involving flight control commands have slightly evolved^{III}. Therefore, the set of coefficients obtained with the WMF method with the "SODA-STIM" payload is used as initial values for the real-time implementation.

Summary

This chapter has presented the results of the coefficient estimation/calibration methodology. At first, the importance of flight dynamics for the correct calibration of the aerodynamic coefficients has been demonstrated. Thereafter, using real flight data, two sets of coefficients have been given for both platforms using the WMF technique. This step has been shown to be essential because very inaccurate initial coefficients led to the impossibility of using state space augmentation for online refinement of the coefficients. Moreover, navigation with such incorrect coefficients has been shown to lead to filter divergence. On the other hand, when an approximate set of coefficients is available, e.g. adapted from a known model, it could be improved with offline methods, benefiting from precise observations and smoothing. With correct coefficients, the VDM-based navigation system is operational and the next chapter will present some of its real-time analyses.

^{III}The initial models[11] used [*rad*] for control surface deflections and [*rad/s*] for propeller speed, while the current model directly uses encoded PWM for the actuators (normalized) and propeller speed

9 Real-Time Aspects

Overview

One of the main objectives of this thesis is the design of a real-time prototype of VDM-based navigation system. So far, most of the navigation performance results and investigations have been obtained by simulations in a controlled setting. This new navigation approach needs validation with a real-time environment to confirm its feasibility on a low-cost/low-power embedded system on a small UAV. As presented in Chp. 7, such design requires specific hardware and software to solve the challenges brought about by an online system. This chapter covers some of the results attributed to the real-time aspects of the VDM-based navigation implementation.

First, the influence of incorrect time-tagging of flight control commands is analyzed with simulations to appreciate the impact on navigation performances in a real scenario. The internal clock of the AP has a drift of more than 100 *ms* per hour. Therefore, time-tagged sensor data, and in particular flight control commands, have to be continuously corrected using a GPS-AP time regression thanks to GPS's pulse-per-second (PPS). Then, the setup to test the real-time system is described. This setup comprises of the *TP2* aircraft, the modified ground control station to monitor the UAV online, and the complete embedded VDM-based navigation system using the ROS environment. The computational load of the navigation software is analyzed with different filter sizes for the specific embedded computer. Afterward, the initialization of the filter is discussed. The advantages of the partial update during the first seconds of the filter initialization are put forward to reduce the error in position. Finally, the initial wind conditions and the estimation are discussed. The real-time wind estimation during two demonstration flights without the use of an airspeed sensor is compared with the offline method used in Chp. 5 for coefficients calibration. The results come from publications and projects [95]. The overview of the real-time results is summarized in Tab. 8.1 with the corresponding application and sections.

Table 9.1: List of real-time-related results

Investigations	Simulation	TP2	Section
Time tagging			
Simulated delays	✓		9.1.1
PX4 vs GPS clock		✓	9.1.2
On-line demonstrator			
Components		✓	9.2.1
Computation load		✓	9.2.2
Initialization			
Incorrect initial VDM parameters		✓	9.3.1
Partial update		✓	9.3.2
Wind			
Wind initialization	✓		9.4.1
Wind estimation		✓	9.4.2

9.1 Sensors Time Delay and Influence of Time-Tagging errors

9.1.1 Requirements

VDM requires that the input of the flight control command be related to the same absolute time frame as the other observations. The (non-) tolerances of possible delays in control input command with respect to navigation performance on a fixed-wing UAV are analyzed.

Methodology

The control input (U) coming from the autopilot and presented in Sec. 3.5.1 is fed to VDM. In an ideal real-time implementation, the control input data should be read directly from the microcontroller that commands these inputs. For the *TP2* platform, the actuators include two servos for the ailerons, one servo for the rudder, one servo for the elevator, and one motor to spin the propeller. In most systems, the control commands can only be accessed from the onboard computer that hosts the autopilot. The latter time tags the data with its own system time. The data time tagging operation requires its own processing time, and depending on the computational load of the system, some delays d_{cc} can be expected between the time a control command is emitted t_0^{emi} and the time when it is time-tagged t_0^{tag} . This is illustrated in Fig. 9.1. Furthermore, the low quality of the system time clock will cause unavoidable drifts, adding other timing errors (Sec. 9.1.2).

To assess the influence of imperfections in the data time label on the performance of VDM-based navigation, a similar simulation framework as in Sec. 8.1 is used. The steps are shown in Fig. 9.2.

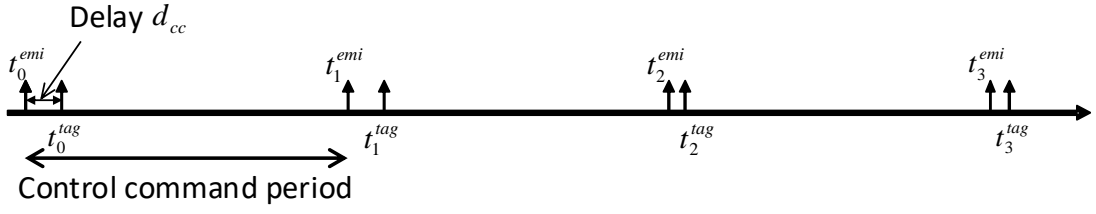


Figure 9.1: Time-tagging of the emitted control commands with delays

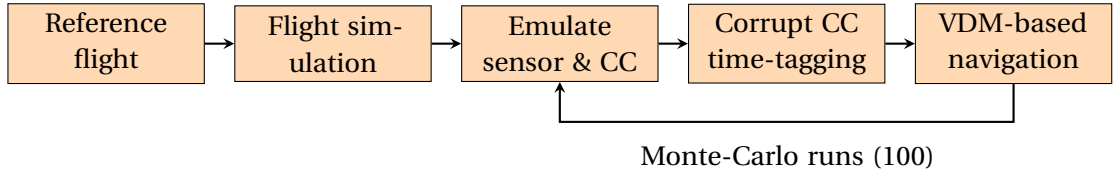


Figure 9.2: Flowchart of simulations to assess the effect of time-tagging errors

Reference, Flight Simulation and Sensor Data

Two flights are considered, the reference trajectories shown in Fig. 9.3 and introduced in Sec. 6.4.1. The first flight, denoted as *20170821_nx5id1*, lasts 425 seconds; while the second flight, denoted as *20170822_nx5id1*, lasts 272 seconds. Both flights are carried out using the TP2. PPK GNSS solution with cm-level accuracy is used to generate way points to emulate flights in the simulations.

Once the way points are generated from real flight data, the corresponding sensor data (with errors) are simulated for IMU, GNSS, barometer, and the control commands similarly as presented in Sec. 8.1. As the investigations aim to assess the effect of time-tagging error in the control input, emulated sensor readings with correct time tags are used to avoid mixing effects.

Time-Tagging Errors

At this step, the time-tagging of control input data is corrupted by adding stochastic delays to simulate a realistic case. The nature of such delay in the real case depends on the internal properties of the system (clock frequency, thread priority, system load) and is unknown. In this study, the time delays of the control command d_{cc} are modeled as the sum of a constant bias b and a positive random delay w as

$$d_{cc}(n) = b + |w(n)| \quad (9.1)$$

where $w \sim \mathcal{N}(0, \sigma^2)$. The impact of time-tagging errors in VDM-based navigation performance is studied for different values of b and σ .

Control commands are separated into two groups: those related to the engine/propeller and those related to the servos (for the aileron, elevator, and rudder). The servo and motor Rotation Per Minute (RPM) commands are assumed to be accessible at 10 Hz, as in the employed UAV. The ideal time tag is corrupted by delay d_{cc} separately for each data. The meaning of the emitted and tagged times is illustrated in Fig. 9.1.

When the motor and servo time tags are corrupted for a particular fixed delay and absolute noise error level, VDM-based navigation is performed. For both flights, the 3 min long GNSS outage is considered before the trajectory ends. There are 15 runs for every combination of fixed and random delays with different noise realizations, and the maximum position errors are saved for each run for statistics.

Navigation and filtering in the simulation are set to 100 Hz, but it also runs every time a new control input (with corrupted time-tagging) is available. This guarantees that the processing scheme does not introduce an additional delay due to buffering.

Results and Discussion

VDM-based navigation with 3 min of GNSS outage without any time-tagging error is performed for reference and is presented in Fig. 9.3. The corresponding maximum position error at the end of 3 min-long autonomous navigation is 83 m for 20170821_nx5id1 and 42 m for 20170822_nx5id1. It can be observed that the autonomous VDM-based navigation solutions drift in both flights but stay quite close to the references while the INS-based navigation drifts considerably faster.

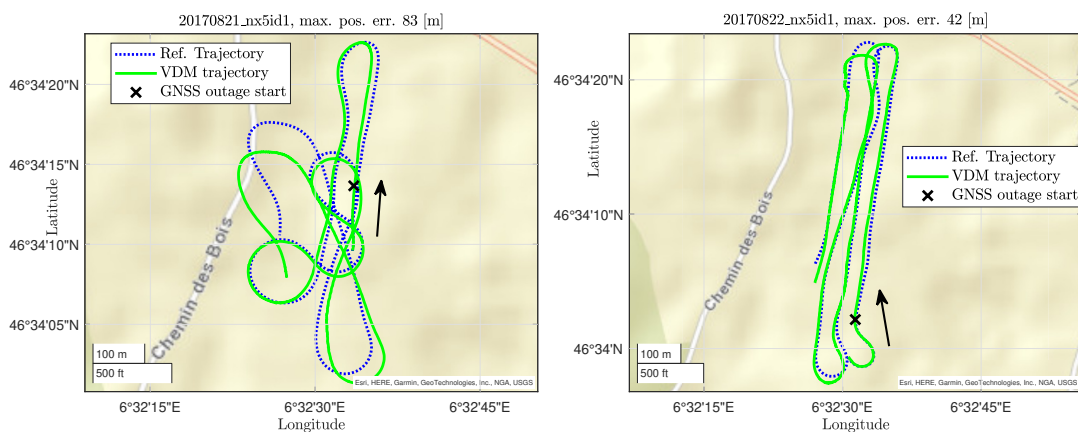


Figure 9.3: Reference trajectory (dashed-blue) and VDM-based navigation solution (green) with correct control command time-tagging during 3 minutes of GNSS outage

The cases with different levels of delays are simulated separately for motor and servo commands and are presented below.

Motor Data Time-Tagging Errors

Investigated motor data time-tagging errors range from 0 to 300 *ms* of fixed delay plus a random noise delay 1σ from 0 to 100 *ms*. The values considered are reported in Tab. 9.2. Fig. 9.4 shows the maximum position error for all runs (15 for each corruption scenario) that

Table 9.2: Fixed and random delays for motor data

Fixed Delays [ms]	Random Delays 1σ [ms]
0	0
10	10
50	20
100	50
200	100
300	\emptyset

combine random and fixed delays. The maximum position error for the five random delays combined with the more considerable fixed delay (300 *ms*) is depicted with boxplots (Fig. 9.4). Motor time-tagging errors do not appear to introduce a noticeable deterioration of VDM-based navigation performance compared to the case with perfect time-tagging. A possible reason is the slow dynamic response of UAV to the speed of the propeller, which causes relatively small time-tagging errors (with respect to the associated dynamic modes of the UAV) that have minimal impact on navigation. Indeed, the absence of wind in the simulation and the nominal and almost constant velocity of the platform during both trajectories imply almost no change in propeller speed, simplifying the model. More significant errors in position are expected if those two conditions are different, i.e. the wind is present or added in the simulation and the platform changes its velocity during the flight.

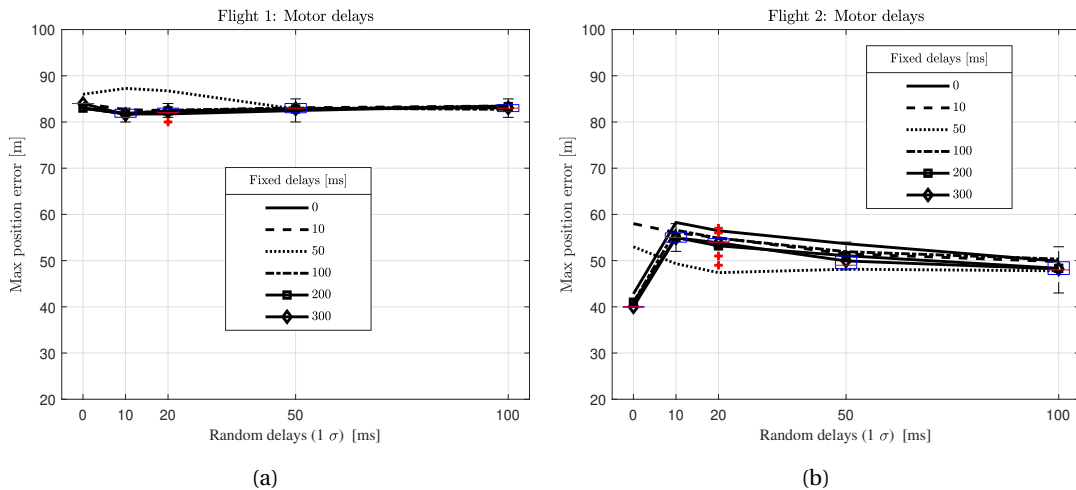


Figure 9.4: Maximum position error for all runs with motor time-tagging errors

Servo Data Time-Tagging Errors

Considering the high sensitivity of UAV dynamic response to servo data time tagging errors, the chosen fixed delay, and random noise standard deviation are limited within the range from 0 to 20 ms as shown in Tab. 9.3. The effect of those errors on max. error in position is plotted in Fig. 9.5.

Table 9.3: Fixed and random delays for servos data

Fixed Delays [ms]	Random Delays 1σ [ms]
0	0
5	5
10	10
15	15
20	20

The growth in position error is noticeable when the time-tagging error of the servo data increases. It is interesting to notice that the evolution of the errors with random delay follows a different trend per fixed delay. This may be due to a relatively low number of runs for each

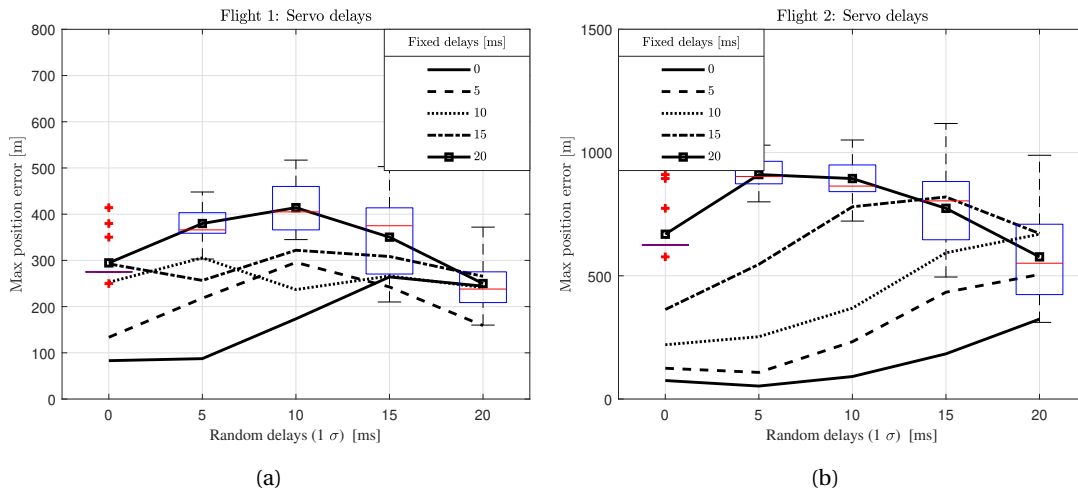


Figure 9.5: Maximum position error for all runs with servo time-tagging errors

(of many) combination. Although longer delays have been tested, these are not presented to stay with the primary objective: identify the central tendency of position error growth. A fixed delay of only 10 ms increases the maximum error in position more than three times, even without random delay. This reveals the importance of proper time-tagging control input data for VDM-based navigation.

Fig. 9.6 shows VDM-based navigation solution with an introduced fixed and random delay of both 10 ms in the servo data for a particular run. The maximum position error during

9.1 Sensors Time Delay and Influence of Time-Tagging errors

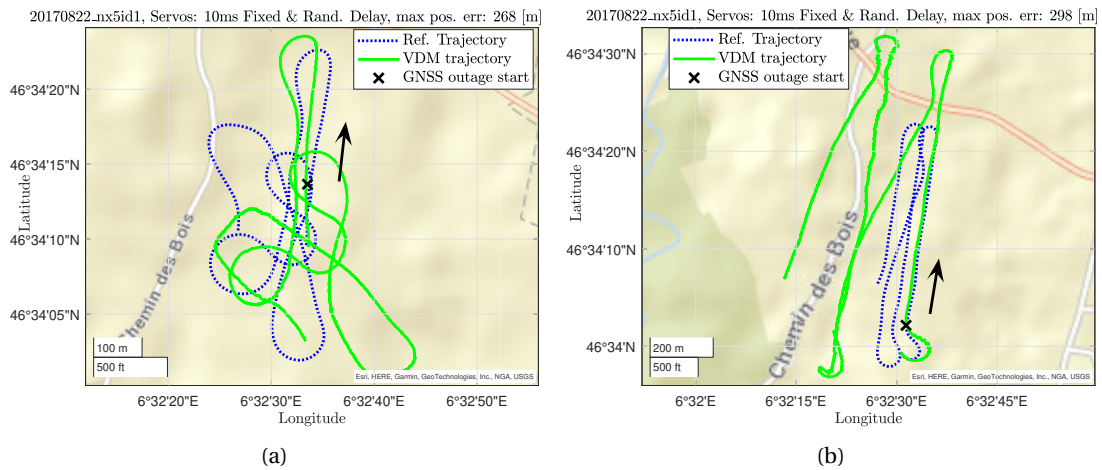


Figure 9.6: Reference trajectory (dashed-blue) and VDM navigation solution (green) over 3 minutes of GNSS outage with 10 [ms] of fixed and random delay error for the servo time-tagging

a GNSS outage for both flights is around 300 m, i.e., about 3 times larger than the nominal scenario. However, the trajectory dependency of dead reckoning methods makes the direct comparison between trajectories less relevant. Note that the magnitudes of vertical errors are limited by the barometer measurements. There is an in-flight refinement of the VDM parameters when GNSS data are available. *20170821_nx5id1* lasts 152 seconds longer than *20170822_nx5id1* and experiences more dynamic maneuvers, therefore, it is reasonable to expect a finer estimation of the VDM parameters resulting in slightly better performance in autonomous navigation.

Simulations are also performed by combining time-tagging errors for motor and servo data. However, compared to servo-error only, the difference with the addition of time-tagging errors for the motor input is almost negligible. Therefore, these results are not presented.

Discussion

For both trajectories, it is observed that the time-tagging error in the motor data has negligible influence on the maximum position error for all fixed and random delays considered. However, the impact of time-tagging errors on the servo data is incontestably greater. Rapid growth in position error is observed even for delays as small as ten milliseconds. Therefore, the correct time tag of the control command should be implemented with a global time reference at 1ms-level or better as presented in Sec.9.1.2.

On a side note, the results for both flights reveal a correlation with the trajectory characteristics, which is generally relevant for both kinematic and dynamic-based modeling. Therefore, the maximum position error is not necessarily the most pertinent criterion to assess the performance of the VDM-based navigation solution. However, it provides a bulk estimate of

the engineering requirements for implementing time-tagging in fixed-wing UAVs.

9.1.2 PX4 Control Command Time-Tagging

To relate the potential degradation of navigation performance due to incorrect control command time-tagging, the temporal stability of the autopilot oscillator is evaluated in terms of scale variation to GPS time. The demonstration of this synchronization is carried out on a sample flight (*AF_i7*) of about 30 minutes. A regression fit over the entire period translated to a drift of 0.12 s/hour between the PX4 system time and the GPS time. To capture the possible variation of frequency in the autopilot oscillator during flight, the real-time scale computation uses a buffer limited to 30 seconds, performing a regression over this period (30-second windowing). The updated scale s and bias b (taken at the beginning of the regression) is outputted as ROS topic `clock_synck` for further GPS time-tagging at the node `TimeSync` (Sec. 7.2) of the control commands and other information coming from the autopilot such as wind estimation, static and airspeed data.

If these offset changes are not compensated for in a flight of around 30 minutes, an incorrect time-tagging to the filter could potentially reach 50ms. As presented in Sec. 9.1.1 an incorrect time-tagging of the actuator of 10ms is responsible for a drift in the solution of 200 m (Fig. 9.6). Therefore, the correct time-tagging of the autopilot control commands is crucial. The proposed time tag in a global reference time avoids navigation errors as presented in the previous section due to incorrect time-tagging from a potential drifting clock.

9.2 VDM Online Demonstrator

The VDM online demonstrator aims to present the feasibility of VDM-based real-time navigation by merging theoretical and engineering concepts developed during this research into a prototype. Numerous flights have been performed to test the required hardware and software over several months, and the results presented come from multiple ones.

9.2.1 Components

The target demonstrator is based principally on the following components:

1. The *TP2* aircraft presented in Sec. 6.1 with the "STIM-SODA" payload (Sec. 6.2)
2. The Ground Control Station introduced in Sec. 6.3.3
3. The embedded VDM-based navigation software (brief description in Sec. 7.3, detailed implementation in Sec. 7.3.2, the additional required ROS nodes in Sec. 7.2)

The description of points 1. and 3. have already been detailed in the related sections. The modifications made to QGC for the online demonstrator requirements are presented below.

Modified QGroundControl

In its standard implementation, only the position and attitude of the UAV, calculated by PX4 during a mission, is displayed on the map. This is represented by the red line in Fig.9.7(a). QGC can be customized to add other tracks (e.g., blue track in Fig.9.7(a)) in parallel to the native track. This allows comparing the navigation solution of *Gi iNav* or *VDMc* (both possibly without GNSS) to the one calculated by the PX4 autopilot (with GNSS) when they are transmitted by the UAV embedded computer to the autopilot and received by the GCS software via MAVLink messages. This will allow us to observe the evolution of the different navigation solutions in real time and compare their performance.

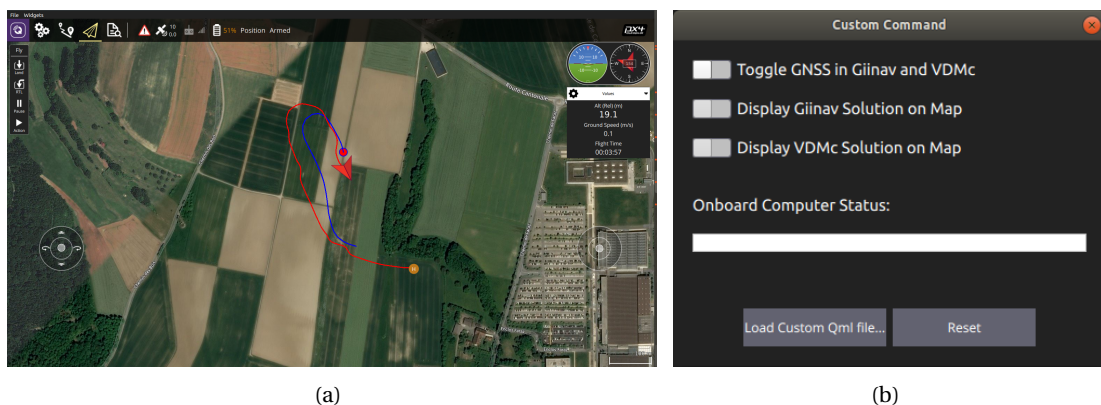


Figure 9.7: (a) Screen capture of QGC during a mission, displaying two tracks on map: navigation solution computed by PX4 autopilot (in red), navigation solution computed by INS/GNSS *Gi iNav* (in blue). *VDMc* trajectory not displayed. (b) Widget to send custom command to the UAV embedded computer and display received telemetry messages

Custom Commands

The GUI of QGC is modified to have bilateral communication with the companion computer to transmit custom commands and messages. For this purpose, a custom widget containing switches is designed and added to the graphical user interface, as shown in Fig. 9.7(b).

When activated by another selector, the pose of *Gi iNav* and *VDMc* can also be displayed in real time on GUI. One of the switches (the top one) sends a MAVLink message to the autopilot, which is relayed to the companion computer to block access to the GNSS data for *Gi iNav* and *VDMc*. This allows us to artificially simulate a GNSS outage during flight. To monitor the status of the different navigation nodes present in the UAVs, a display area is also added to the widget; however, message parsing is not yet implemented. For safety reasons, the PX4 autopilot is not affected by the software emulated GNSS-signal outage and its solution is used as reference. The three positioning solutions (i.e. the reference, INS and VDM) are communicated back to QGC and displayed on its screen.

The presentation of the online demonstrator follows the flight steps presented in Sec. 7.4 where some details have already been covered.

9.2.2 Computational Load

The implementation of the VDMc software on the companion computer supports varying filter sizes depending on the chosen submodels. The calculation of navigation solutions requires performing a large number of matrix operations, which consumes most of the computational power of the Central Processing Unit (CPU). The embedded computer introduced in Sec. 6.3.2 is relatively powerful. However, the target unit is likely to be a microcontroller. To have a first insight of the time needed to perform the different steps in the EKF, three different filter configurations for the TP2 platform, including with 68, 47 and 27 states, as recalled in Tab. 9.4 using the principle presented in Sec. 7.3.5. In the following, the three different

Table 9.4: Three different systems with various sub-models tested with the VDMc software on the embedded computer

Number of states	Sub-models
68	$\mathbf{X}_n, \mathbf{X}_p, \mathbf{X}_a, \mathbf{X}_{a_d}, \mathbf{X}_w, \mathbf{X}_e, \mathbf{X}_{L_{imu}}, \mathbf{X}_{B_{imu}}, \mathbf{X}_{L_{gps}}$
47	$\mathbf{X}_n, \mathbf{X}_p, \mathbf{X}_a, \mathbf{X}_w, \mathbf{X}_e$
27	$\mathbf{X}_n, \rho, \mathbf{X}_a, \mathbf{X}_w, \mathbf{X}_e$

filter sizes will be referred as "68-states", "47-states" and "27-states". The two main steps in a KF are state prediction and update. The execution time of these steps is monitored¹ and are shown in Tab. 9.5. The filter frequency step is set to 100 Hz. When using the filter

Table 9.5: Execution time of the prediction and update steps for the different filter size

System	functions	Average exec. time [μs]	CPU load reduction [scale]
"68-states"	pred.	5355	ref. value
	update-imu	1079	ref. value
	update-gps (pos. + vel.)	2029	ref. value
"47-states"	pred.	2490	2.2
	update-imu	763	1.4
	update-gps (pos. + vel.)	1545	1.3
"27-states"	pred.	762	7.0
	update-imu	414	2.6
	update-gps (pos. + vel.)	623	3.3

with 68 states (using extra auxiliary states such as sensor misalignment, actuator dynamic parameters, and more complex IMU error model), the time required for a prediction step is around 5.0 ms on the current embedded computer. Therefore, the filter can run without

¹These investigations were performed on a previous version (2021) of the VDMc software with the payload "SODA-GECKO". These average execution times are expected to be shorter in the current version on a common scale, but the relative load reduction should follow the same trend

overloading the CPU with this configuration, which is the most "computationally expensive". However, the reduced version of the software when using the 47- or 27-state configuration requires considerably less execution time ($\sim 2x$ and $\sim 6x$), which is potentially interesting for a microcontroller host. Additionally, code optimizations and KF algorithms can be used to further reduce the computational load of the VDMc software.

9.3 Initialization

9.3.1 Incorrect Model-Parameter Initialization

To emphasize the importance of correct initialization of aerodynamic parameters in a real scenarios, different initial sets of parameters are tested in the VDM-based navigation system. These tests are carried out on data from the application flight *AF_STIM7* with a 47-dimensional state vector. The initialization sets used in this test are as follows: (i) all ones, (ii) all zeros (approximated to 10^{-7}), and (iii) all random with $N \sim (0, 1)$. The effect of these incorrect initial parameters is evaluated during a simulated GNSS outage of 2 minutes after four minutes of flight. The scenario with the VDM parameters initialized with all zeros results in a straight-line trajectory during the 2 minutes of the outage because the model does not produce tangible moments or forces. The second scenario with initial values set to ones is completely erratic with a final position error of ~ 600 m. The last scenario reaches a numerical instability within the first second (and crashes). The results obtained with incorrect initialization are consistent with the simulations presented in Sec. 8.1.

9.3.2 Initialization Improvement with Partial-Schmidt Update

The benefit of the partial update (PSKF) implementation detailed in Sec. 4.2 with the *TP2* is tested with the payload "IGN-GECKO" (Sec. 6.2.1) on four flights.

The *CF_i8* flight is used for calibration. Three other flights from the same campaign (*IGN6x*, *IGN6U*, *IGN7*) are used as application flights. They are for these results called *AF_i7*, *AF_i6x*, and *AF_i6u* and are used to test the modifications suggested to the VDM-based navigation system. As presented in Fig. 6.9, the flights are dissimilar in their geometry, combining dynamics and a block pattern for *CF_i8*, different dynamics and block for *AF_i7*, a long straight corridor for *AF_i6x* and a u-shaped corridor for *AF_i6u*.

Methodology

The coefficients \mathbf{x}_p obtained from the flight *CF_i8*, after refinement as presented in Sec. 8.3.2 with the corresponding block-covariance matrix \mathbf{P}_p , are used as priors for all other validation flights, while increasing the uncertainty of 1σ by a factor 1.02 (2%). During the three validation flights, the VDM parameters are fine-tuned (when GNSS observations are present) to allow small possible refinements due to changes in the platform configuration and its environment

related to battery position, reassembled parts with slightly different orientation between flights, small modification on the payload, or changes in weather conditions. Therefore, the initial error state of the model parameters $\mathbf{x}_p(0)$ is zero. Initial values of navigation parameters and IMU sensor errors are obtained from conventional INS/GNSS together with their confidence levels (Sec. 6.2.1). Therefore, the initial values of the error states $\mathbf{x}_n(0)$ and $\mathbf{x}_e(0)$ are zero, as are those of the actuator errors $\mathbf{x}_a(0)$. The position of the actuators is obtained from the autopilot. The initial wind is set to zero with an uncertainty of 0.5 m/s and 0.1 m/s in the horizontal and vertical directions, respectively (1σ).

Initialization periods of different durations ($T_{ini} \in [50, 100, 300]$) are tested on the three application flights to observe the influence of partial updates (Sec. 4.2) on the fluctuation of position errors. Their maximum horizontal position error during the first 200 seconds after initialization is shown in Fig. 9.8 for the three application flights. For all cases with $T_{ini} \geq 50\text{ s}$,

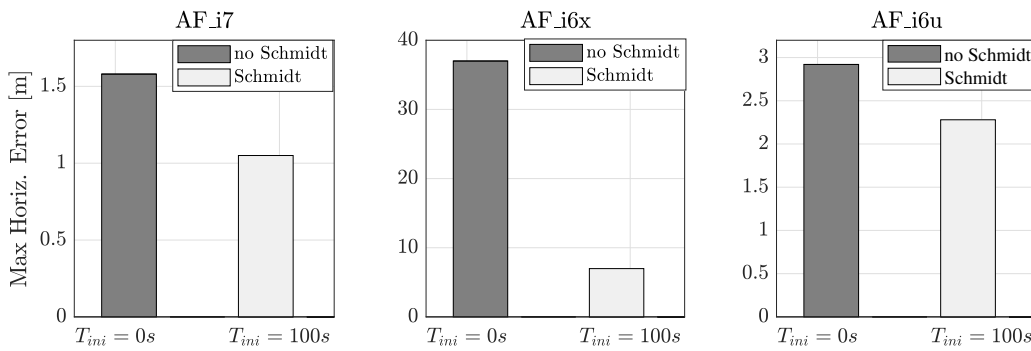


Figure 9.8: Max. position error during the first 200 s after initialization without and with partial-Schmidt ($T_{ini} = 100\text{ s}$) for the three application flights

the maximum position error is reduced, improving the estimation with respect to the reference trajectories. In a similar trend, the respective norms of velocity and attitude errors also decrease. These improvements are only marginal for T_{ini} to be longer than 1 or 2 minutes, which is why $T_{ini} = 100\text{ s}$ is chosen as the default value. Fig. 9.9 further shows the detailed evolution of the horizontal error (magnitude) within flights *AF_i7* and *AF_i6x* without ($T_{ini} = 0\text{ s}$) and with partial updates ($T_{ini} = 100\text{ s}$), respectively. For *AF_i6x*, the benefit of an “initialization” phase is substantial as the position error without partial update is quite large. This is likely due to the instability of the filter caused by incorrect initial values of some parameters. Applying the partial-Schmidt filter reduced the maximum error in position by a factor of 6.

Generally, within the three application flights, all navigation state errors decrease when an initialization time of close to 1 minute or longer is selected. For initialization periods lasting longer than 5 minutes, there seems to be a higher dependence on the initial state values, which reduces the rate of convergence. However, a longer initialization time (more than 500 seconds, for example) may be considered on some flights with limited dynamics, such as those flown for mapping missions [28]. These types of flight are monotonous with repetitive patterns that are flown at constant height and constant velocity. There, the benefit of partial Schmidt in the

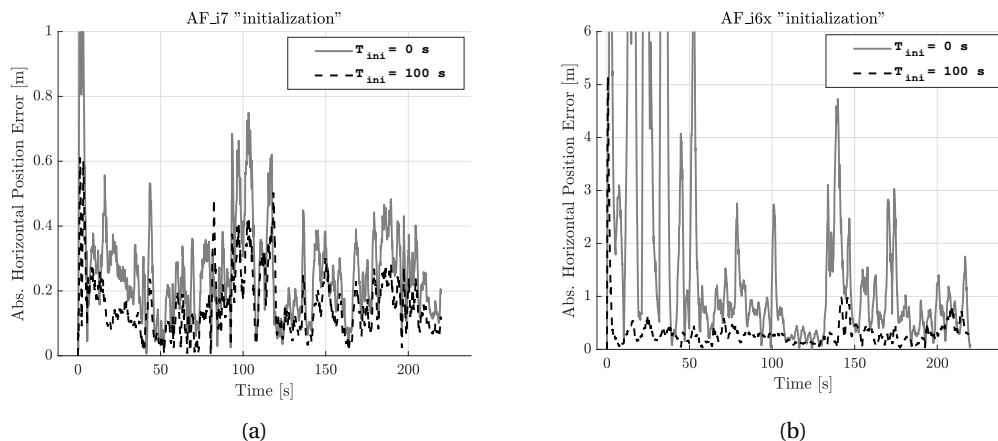


Figure 9.9: Evolution of horizontal position error (magnitude) after initialization for $T_{ini} = [0, 100]$ s in flights *AF_i7* (left) and *AF_i6x* (right)

initialization phase of VDM-based navigation is less certain, because the dynamics during such missions is low, and, in turn, the criteria for obtaining sufficient observability to refine the aerodynamic coefficients may not be achieved [52]. Wind estimation can benefit from such a partial update method, and its initialization is evaluated hereafter.

9.4 Wind

The correct wind estimation is of primary importance for accurate VDM-based navigation. First, a simulation with real wind is investigated to understand the influence of initial conditions (Sec. 9.4.1). Then, the real-time wind estimate for flight *STIM_12* and *STIM_13* is presented later in Sec. 9.4.2.

9.4.1 Wind Estimation in Simulation

The *TP2* model with the coefficients obtained after calibration (Sec. 5.3.2) is used with a simulated flight of about 15 minutes to test the influence of the wind velocity values in the initial state for its estimation during the simulated flight depicted in Fig. 9.10. The trajectory is designed to have high dynamics with heading modifications to ease the wind estimation (Sec. 5.2) and the refinement of the coefficient (8.1). The generation of sensor data and flight control commands is similar to the methods presented in Sec. 8.1. The initial uncertainty of the coefficients \mathbf{P}_0 is set to 1% (1σ) of the calibrated values. The real wind record taken at 60m AGL provided by *Wind Engineering and Renewable Energy at EPFL (WIRE)* is added to the simulated environment presented in Sec. 8.1 and Sec. 9.1.1. Two different initial wind velocities are used: (i) zero wind for the three axes and reference as scenario *init_0*, and (ii) 1.5m/s, 3m/s and 0m/s for the North, East, and West wind velocity, respectively, and referred to as *init_non0*. The initial wind uncertainties \mathbf{P}_0 (1σ) are 0.5m/s for North and East, and

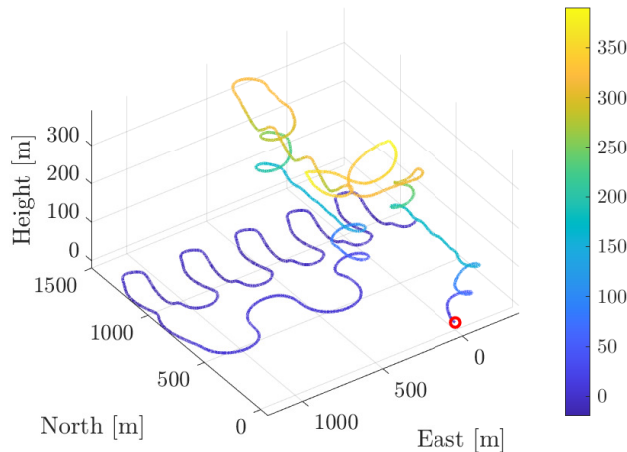


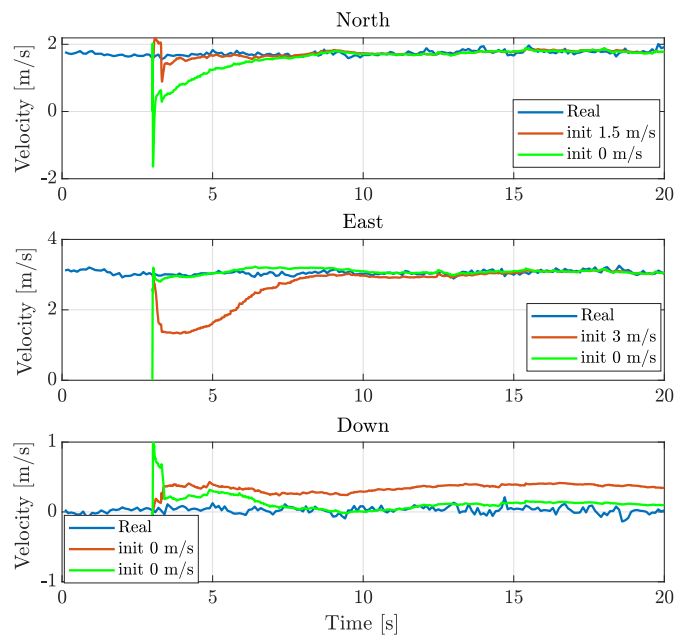
Figure 9.10: Simulated 15-minutes trajectory

0.1 m/s for the down component for both initialization scenarios. The wind estimate for the first 20 seconds (Fig. 9.11(a)) and the remaining flight (Fig. 9.11(b)) with respect to the wind reference is presented for each axis. For all three directions, when the initial wind is set to zero, the wind estimate tends to make large corrections before converging to the real value. The difference in wind estimation between the two initialization scenarios seems to last less than 10 seconds. The difference in estimate after that is only marginal and cannot even be distinguished between the two scenarios after a few minutes. When calibrated aerodynamic coefficients are used with the simulated trajectory, the estimated RMSE of the wind relative to the real horizontal and vertical is 0.11 m/s and 0.05 m/s , respectively, and 0.10 m/s and 0.07 m/s .

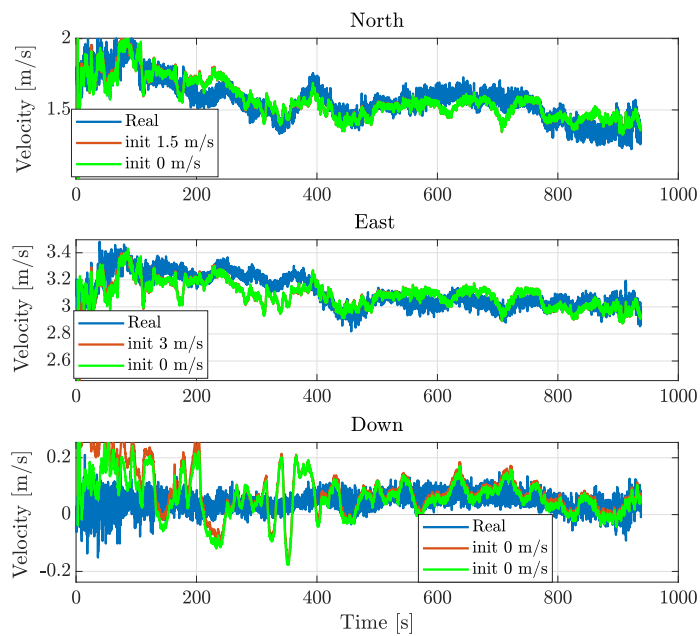
For the defined research objectives, these findings are sufficient and have not been further investigated: with correct aerodynamic coefficients, the in-flight wind estimate is “correct”. However, more tests could be performed to investigate the accuracy of wind estimation in other scenarios (i.e., with gusts). This is now part of the research of an additional thesis using the VDM-based navigation scheme conducted in *TOPO-EPFL*.

Recommendations

To mitigate the large oscillations in wind estimation during initialization, a partial update is recommended as presented in Sec. 4.2. However, suppose the local wind is not approximately known a priori. In that case, the partial update on the wind velocity states should be used with care: initializing the wind with zero values with a slow update convergence might corrupt the remaining auxiliary states by compensating for the incorrect wind velocities before their convergence to the (a priori unknown) actual value. The evaluation of the wind in real-time is presented below.



(a)



(b)

Figure 9.11: Wind estimation during the (a) first 20 seconds and (b) the whole trajectory of a simulated trajectory for two different initialization scenarios: (i) approximated initial values in red, and (ii) with zeros in green with respect to the reference wind (blue)

9.4.2 Wind estimation in real-time

The horizontal wind estimation using WMF, the PX4 autopilot, and the VDM filter for the online demonstrator flight *STIM_12* is graphically compared in Fig. 9.12. Recall that the two former

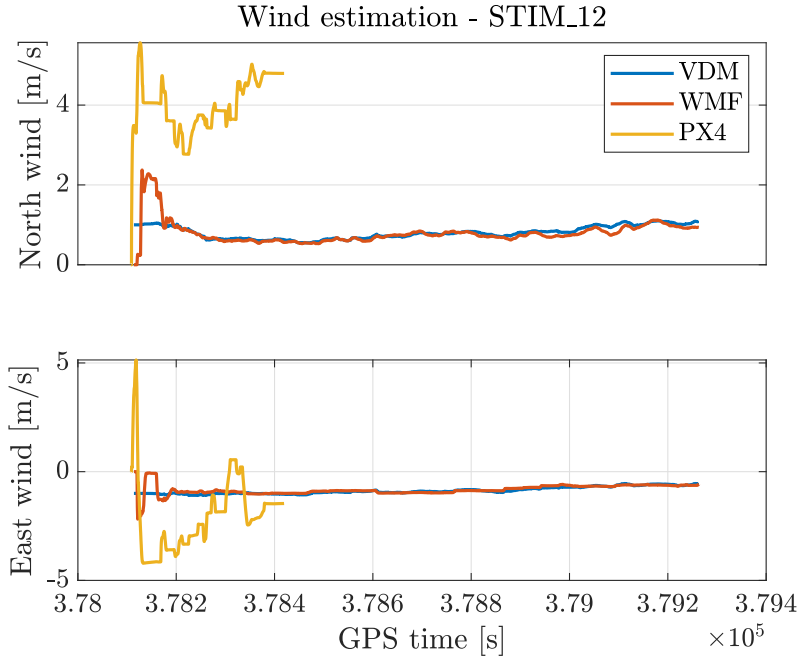


Figure 9.12: Wind estimation for the flight *STIM_12* using WMF (red), the VDM filter (blue) and PX4 (yellow)

estimators (WMF, PX4) use airspeed measurements, whereas the latter (VDM) do not. Wind estimated with autopilot PX4 does not match the other two, probably due to a high residual bias in dynamic pressure measurements^{II}. It can be observed that for the WMF wind, some jumps are presented at the beginning of the trajectory. In contrast, these jumps are absent for the VDM online estimation due to the partial Schmidt update applied at initialization. The initial wind velocities are taken from the weather station readings. Tab. 9.6 summarizes the average direction and velocity of the wind for flights *STIM_12* and *STIM_13* using the two different methods. The wind estimated with the VDM estimator corresponds quite well to the

Table 9.6: Comparison of estimated horizontal wind for *STIM_12* and *STIM_13* using the WMF method and VDM-EKF

Flights	WMF		VDM	
	magnitude [m/s]	direction [deg]	magnitude [m/s]	direction [deg]
<i>STIM_12</i>	1.18	136.4	1.20	137.1
<i>STIM_13</i>	1.89	116.2	2.02	117.9

one estimated with WMF in terms of magnitude (~ 2 [cm/s]) and direction (0.5 [deg]) for the

^{II}Only 5 minutes were saved due to (unknown) autopilot internal logging issue

flight *STIM_12*. The wind was relatively calm and constant. The weather station measured a wind velocity of $1.14[m/s]$ on the ground.

During the second flight *STIM_13*, the wind rose with sporadic gusts, resulting in more heretic wind velocity which is captured with the VDM filter because of continuous estimation. Whereas the WMF method blocks (partial-update) the estimation depending on the direction of the UAV as described in Sec. 5.2.1. This results in a smoother wind velocity estimation. The observed difference in wind magnitude is $\sim 13 [cm/s]$ and the heading is impacted in consequence with a difference of $1.7[deg]$. No result could be obtained from the weather station for this flight.

Summary

This chapter has presented several details and constraints of the real-time VDM-based navigation system. First, the correct real-time time-tagging of the flight control commands has been shown in simulation to be crucial to avoid the deterioration of autonomous navigation. Second, with the correct time-tagging of all subsystems, the three components to test the real-time setup have been briefly discussed: the UAV, the GCS software, and the VDMc navigation filter. Subsequently, the addition of partial updates in the navigation filter during initialization has shown a reduction in errors and removed erratic jumps in the position solution. Finally, during the remaining duration of the mission, real-time wind estimation has been verified using the proposed design. The next chapter will validate the theoretical and engineering contributions by evaluating the real-time system under different autonomous navigation scenarios.

10 Autonomous Navigation

Overview

The presented research strives to improve autonomous navigation of small UAVs, during GNSS-outages, compared to traditional methods. VDM-driven autonomous navigation performance is directly related to the quality of aerodynamic model parameters, the settings (tuning) of the navigation filter, and the proper real-time system design. In this chapter, a two-fold performance analysis of VDM-based navigation system is performed: first, validation of the overall real-time setup, covering the calibration of aerodynamic coefficients, the hardware and software setup, and correct configuration. Second, evaluating the benefits of navigation based on VDM in challenging environments.

Following this idea, this chapter exploits the performance of autonomous navigation during a GNSS outage to (i) validate the coarse estimation methods WMF for the *TP2* and *eBeeX*; (ii) demonstrate the recommended partial update methodology by not updating some states when no observations from the GNSS receiver is available; (iii) compare the VDM-based navigation with traditional INS coasting for *TP2* with the real-time setup during the online demonstrator flight campaigns; and (iv) investigate the reduced model (lumping highly correlated coefficients). The dedicated section to these investigations is summarized in Tab. 10.1 and some of the results come from [28].

Table 10.1: List of autonomous navigation results

Investigations	Payloads			Section
	IGN-GECKO	SODA-STIM	eBee-GECKO	
WMF obtained coefficients validation		✓	✓	10.1
Partial Schmidt improved stability	✓			10.2
VDM vs INS coasting		✓		10.3
VDM-based with reduced model	✓			10.4

10.1 Coarse Coefficient Estimation

The “correctness” of the approximate parameters estimated using the method WMF (Sec. 8.2) for the two platforms (*TP2* and *eBeeX*) is tested by inspecting the performance of autonomous VDM-based navigation under multiple emulated GNSS outages. Furthermore, two different filters are used: (i) the coefficients are kept fixed throughout the flight, and a new state is introduced, known as the aerodynamic scale factor and denoted by s , to compensate for changes in atmospheric conditions and weather between calibration and application flights; (ii) they are used as an initial guess and are re-estimated in the VDM-EKF framework.

10.1.1 TP2

Once the set of force and moment aerodynamic parameters is obtained from the calibration flight *CF_STIM6*, the coefficients are used for validation on the two application flights (*AF_STIM5* and *AF_STIM7*).

Both filters produce encouraging results and mitigate navigation drift during the 2 minute GNSS outage by a very significant margin. This is shown in Fig. 10.1(a) for the application flight *AF_STIM5* and Fig. 10.1(b) for the application flight *AF_STIM7*.

The navigation solution obtained from the fusion of identified aerodynamic parameters is also compared with i) *STIM-318* and ii) *ADIS-16475* during a GNSS outage. Data from the application flight *AF_STIM7* are used using both architectures mentioned above (47/27 state estimator). The result, presented in Fig. 10.1(c), shows that the navigation solution for the two IMUs is quite similar and that the dynamics identified from a high-grade IMU, using the proposed methodology, could be fused with measurements from a lower grade inertial sensor to obtain similar results. On the other hand, if the *ADIS-16475* is used in an INS-driven navigation system, the navigation performance during the outage is poor, and the result is shown in Fig. 10.1(b).

It should also be noted that the UAV for the application flight - *AP_STIM5* is somewhat different compared to the calibration flight and the application flight - *AP_STIM7* due to hardware modifications.

10.1.2 eBeeX

The application flight *eBeeX_652* is first processed with simulated GNSS outages after 23 and 28 minutes, respectively.

The model parameters are included in the state, yielding a 69-dimensional state vector (as discussed in Sec. 5.2). Navigation results are shown in Fig. 10.2(a) and 10.2(b) along with the INS solution. The IMU data are precalibrated: removal of the deterministic noise error thanks to [53] offering the best possible INS solution with embedded IMU (NavChip).

10.1 Coarse Coefficient Estimation

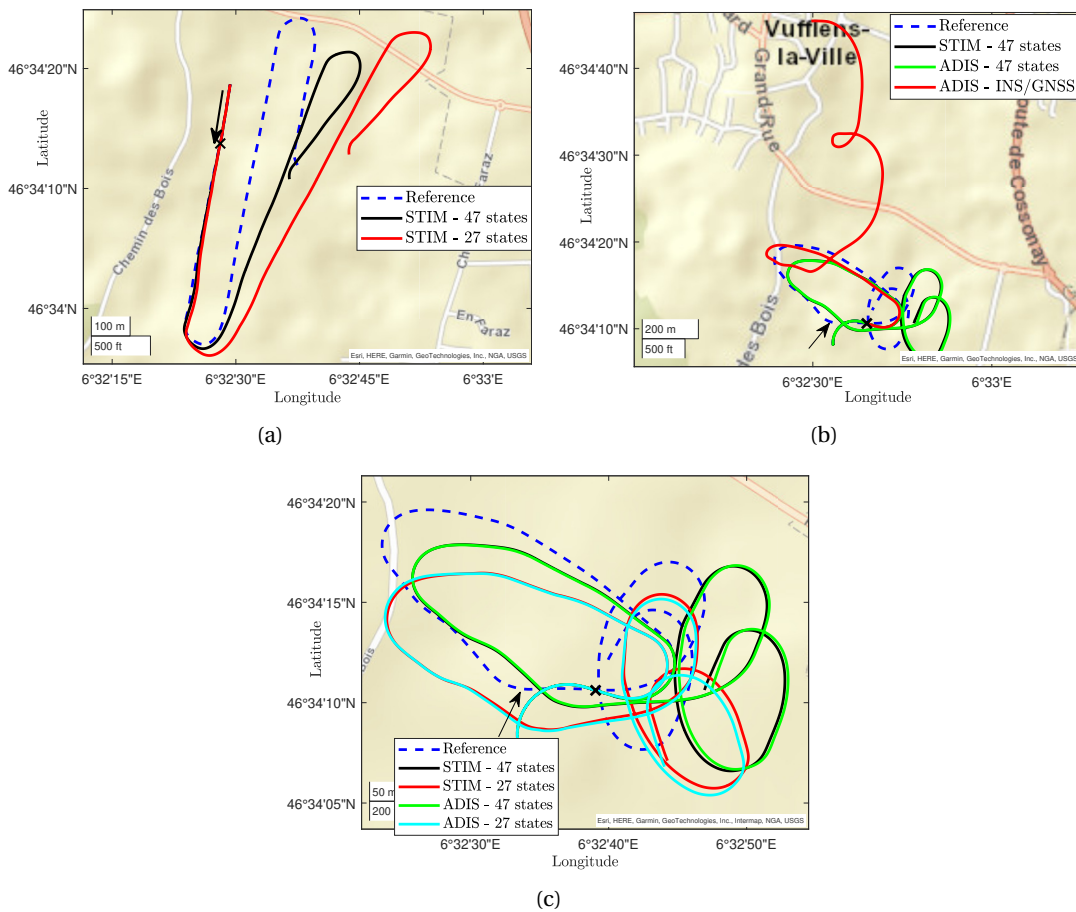


Figure 10.1: TP2 flight *AF_STIM5* (a) with a 2-minute outage after 15 minute, and flight *AF_STIM7* (b) with outage after 4 minutes and (c) invariance of calibrated VDM to different IMU. GNSS outage starts at the black cross

The results are shown in Fig. 10.2(a) and 10.2(b). However, the 26-dimensional estimator, which considers the model parameters constant, did not produce encouraging results. The plausible reason for such a result, based on the following evidence, seems to be overfitting: (i) low residual error during calibration, (ii) large number of unknown parameters, and (iii) significant navigation drift/numerical instability when model parameters are kept fixed during the application phase.

From these two investigations, it can be observed that (i) the position error during an outage when using VDM-based navigation with an uncalibrated IMU (*ADIS-16475*) is reduced compared to inertial coasting (ii), similar VDM-based navigation performance can be obtained with different grade MEMs-IMU, and (iii) better navigation performance is expected with the complete model.

In Sec 10.3, for both platforms, multiple additional outages are emulated to confirm these findings.

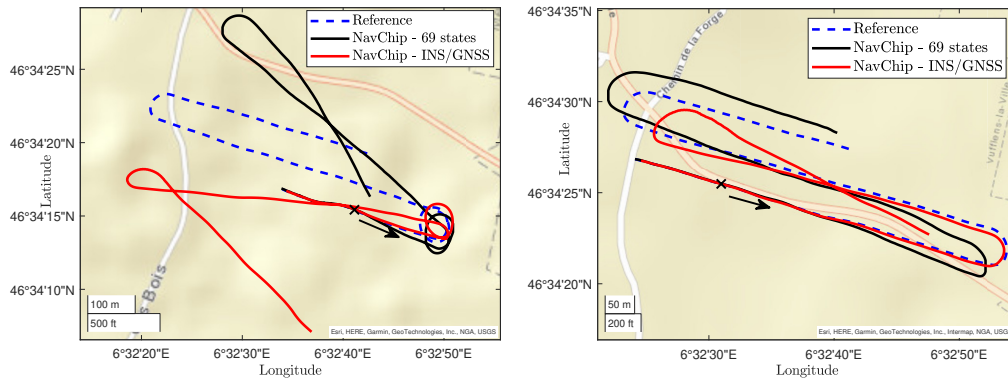


Figure 10.2: eBeeX application flight *AF_eBeeX_652* with 2 minutes outage after (a) 23 and (b) 28 minutes for VDM (black) and INS (red) solution

10.2 Partial Correction of Navigation States

When a GNSS outage occurs, the IMU measurements and barometric-derived height are the only observations available (airspeed observations are used in different investigations). As described in Sec. 4.2, the filter is modified so that all error states related to VDM parameters and that of position are placed in the “considered” mode during a GNSS outage. The effects of this approach are compared to the full state update.

For the application flight, Fig. 10.3 details the maximum and median errors on horizontal position during two GNSS outages (each 2 minutes) with partial (light gray) and full (dark gray) updates of the state vector. Apart from one minor exception, the position errors (as well

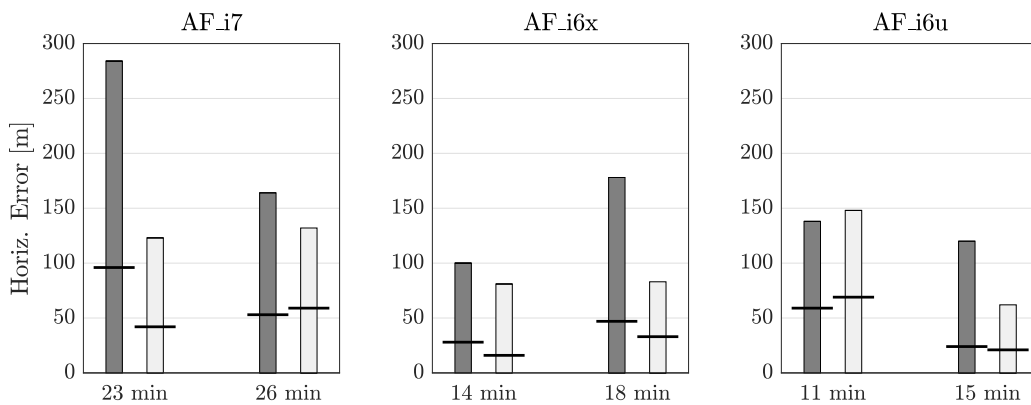


Figure 10.3: Max. and median horizontal errors without (dark grey) and with (light grey) partial updates during 2-min GNSS outages

as the velocity and attitude) are lower in all cases when partial (rather than full) updates are applied. Fig. 10.4 shows the estimated position during some of the GNSS outages described above in the application flights *AF_i7* and *AF_i6u* without (dashed red) and with (green) partial

filtering. The reference trajectory is depicted as a dotted blue line. The minor exception of

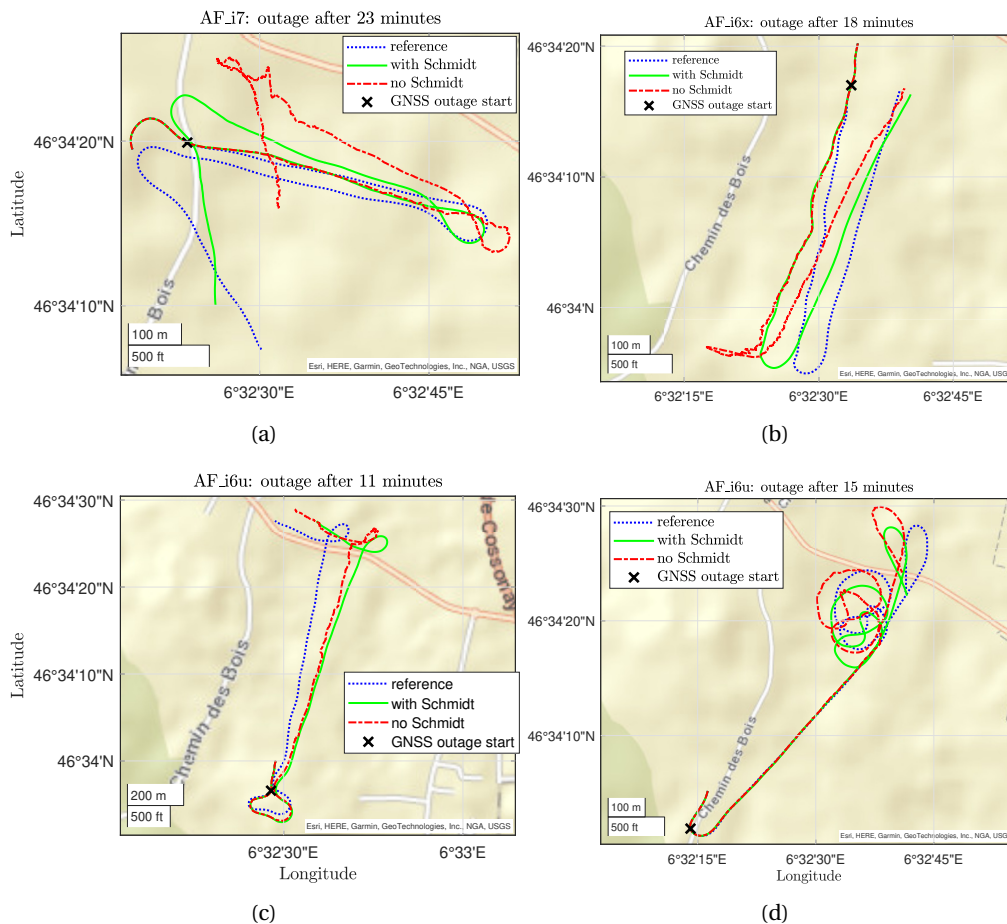


Figure 10.4: 2 minute GNSS outage without (red dashes), with (solid green) partial updates, on (a) *AF_i7*, (b) *AF_i6x*, (c) and (d) *AF_i6u* with the reference trajectory (dotted blue)

a slightly higher positioning error with partial filtering is related to the first simulated GNSS outage in the flight *AF_i6u*. There, the error in the heading is higher with the partial-Schmidt implementation, causing a slightly larger deviation in the horizontal position after the nearly 1 km long straight line, as shown in Fig. 10.4(c). In all cases, the trajectory with partial updates is smoother than the trajectory with updates in position. Such differences intensify toward the end of the outage period when the confidence in position is lower. A smooth and continuous estimate of position with a higher confidence level is more suitable for guidance and control algorithms within the autopilot [106], especially when executing a fail-safe action, such as return-to-land.

10.3 Comparison to Inertial Coasting

This section compares VDM and INS-based navigation that employs 16 states including position (3), velocity (3), attitude (4) and IMU biases (6) for accelerometers and gyroscopes.

Motivation

Recall that the "SODA-STIM" contains, apart from the bias-temperature calibrated IMU (STIM318), another small IMU with low-noise level (*ADIS-16475*). Both IMUs are used in a INS/GNSS navigation setup using flight *AF_STIM7*. Four GNSS outages of 2 minutes are simulated starting at 9, 11, 12 and 14 minutes after takeoff. The horizontal error during inertial coasting for both scenarios is shown in Fig. 10.5. The maximum horizontal error when using

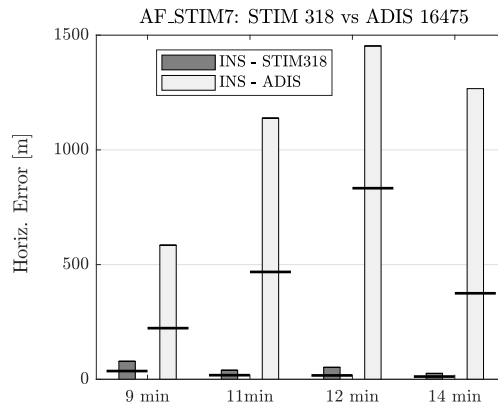


Figure 10.5: Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for INS only with *STIM-318* versus *ADIS-16475* for flight *AF_STIM7*

the uncalibrated *ADIS-16475* IMU reaches more than 1 km compared to the $< 100\text{ m}$ when the *STIM318* is used. However, with reasonably “good” aerodynamic coefficients, VDM-based navigation performance with IMU observations from this lower grade sensor is reported to be similar to when using the higher grade IMU *STIM318* (Sec. A.3.3). This motivates to investigate the comparison in performance under GNSS outages between the INS-based navigation with respect to the VDM version on different platforms and payloads.

Unless otherwise specified in the related section, the initial state uncertainties for both methods are summarized in the Tab. 10.2 with some initial value (IV) specified.

For the different payloads, the IMU measurement noise and error state \mathbf{x}_e follow the characteristic described in Appendix C.2.1. For the first three flight campaigns (eBeeX+GECKO, "IGN-GECKO", WMF - STIM5, 6 and 7) the GNSS mode is PPK, and for the online demonstrator campaign (*STIM12* and *STIM13*), the GNSS mode is SPP with noise characteristic given in Tab. C.4. Extensive GNSS outages are simulated to compare the performance of VDM-based navigation with respect to inertial coasting for the first three campaign flights. They are pre-

States	1σ	Comment
\mathbf{x}_n		Considering "PPK" precision
pos. [m]	0.03 / 0.08	hor. / ver.
vel. [m/s]	0.04 / 0.05	hor. / ver.
att. [deg]	0.5 / 1	$r, p / y$
ω [deg/s]	1 / 2	$\dot{r}, \dot{p} / \dot{y}$
\mathbf{x}_p	1%	IV from diff. cal. method
\mathbf{x}_a		IV from AP
acc. [deg]	1	
n [rad/s]	20	
\mathbf{x}_e		From Appendix C.2.1
\mathbf{x}_w [cm/s]		IV set to zero
hor.	0.5	
ver.	0.1	

Table 10.2: Initial state uncertainty

sented in Appendix A.3. However, the simulated outages for the online demonstrator flights *STIM_12* and *STIM_13* (online demonstrator campaign 1), and the real-time outages for the flights *STIM_14* and *STIM_15* (online demonstrator campaign 2) are presented below for consistency.

10.3.1 Online Demonstrator Campaign

For the generality with respect to the different payloads and IMU qualities, GNSS outages are simulated for some flights with the payload "SODA-STIM". The two online demonstrator flight campaigns are considered. The initial coefficients are obtained thanks to the WMF method with *CF_STIM6*. The GNSS outage start times are summarized in Tab. 10.3 for the flights considered: 2 minute duration (simulated) for campaign 1 and varying for campaign 2 (in real-time).

Table 10.3: Start times (in minutes) of simulated GNSS outage after takeoff for *STIM_12* and *STIM_13* flights (online demonstrator campaign 1) and GNSS outages performed in real-time for the flights *STIM_14* and *STIM_15* (online demonstrator campaign 2)

Flights	GNSS outage start time [min]			
Campaign 1 (simulated)				
<i>STIM_12</i>	10	12	14	16
<i>STIM_13</i>	14	16	18	20
Campaign 2 (real-time)				
<i>STIM_14</i>	9	18		
<i>STIM_15</i>	6			

Online Demonstrator Campaign 1

Due to operational reasons, flight data of *STIM_12* and *STIM_13* are replayed under the same conditions as the real-time system configuration (VDMc) but with multiple GNSS outages for comparing the VDMc implementation with respect to INS coasting. The initial coefficients \mathbf{x}_p for both flights are taken from the calibration flight *CF_STIM6* using the WMF method (Sec. 8.2) and their initial uncertainty set to 2% (1σ). During the first 10 minutes of both flights, the aerodynamic coefficients are refined, the values presented in Tab. 10.4 and then fixed, resulting in a 26-state filter. This time, the aerodynamic scale factor s is not added to the state space (the results with the addition of the scale factor s are presented in Appendix A.3.3).

Table 10.4: Initial coefficient and after 10 minutes of refinement for flights *STIM_12* and *STIM_13*

Force	Initial	<i>STIM_12</i>	<i>STIM_13</i>	Moment	Initial	<i>STIM_12</i>	<i>STIM_13</i>
C_{FT_1}	0.0002	0.0009	0.002	C_{Mx_α}	-0.0053	-0.015	-0.003
C_{FT_2}	-0.023	0.043	-0.039	C_{Mx_β}	-0.0042	-0.014	0.0005
C_{FT_3}	58.61	11.064	33.25	$C_{Mx_{\omega_x}}$	0.0312	-0.121	-0.234
C_{Fz_1}	-0.424	-0.082	-0.561	$C_{Mx_{\omega_z}}$	0.0016	0.0017	0.007
C_{Fz_α}	-6.149	-13.83	-0.256	C_{My_1}	-0.0005	-0.0003	-0.0007
C_{Fx_1}	4.44	-0.254	-2.599	C_{My_e}	0.0601	0.195	0.079
C_{Fx_α}	0.0306	0.017	-0.256	C_{My_α}	-0.0372	-0.074	-0.017
$C_{Fx_{\alpha_2}}$	-3.120	2.251	0.646	$C_{My_{\omega_y}}$	-0.3829	-0.633	-1.794
$C_{Fx_{\beta_2}}$	0.4542	0.229	1.157	C_{Mz_r}	0.0050	0.012	0.005
C_{Fy_1}	-4.728	-0.306	2.15	C_{Mz_β}	0.0094	0.027	0.010
				$C_{Mz_{\omega_z}}$	0.0021	0.0015	0.0020

From Tab. 10.4, we can observe that some coefficients remain pretty similar to the initial value for both flights after the 10-min refinement, whereas others change in sign and magnitude. The direct interpretation of these changes is not straightforward. It is currently attributed to a change in wind conditions, a different trajectory dynamic at the beginning of the flights, and a change of battery of different capacities (different weights, implying modification of the total mass and possibly its center).

Fig. 10.6 presents the horizontal errors after the simulated GNSS outage of 2 minutes for the online demonstrator campaign flights *STIM_12* and *STIM_13* using VDM and INS. For comparison, the flights are run another time without reducing the state space after 10 minutes, keeping the VDM parameters to be estimated throughout the trajectory. For 75% of the scenarios, the full filter (47 states) performs slightly better than when the coefficients are fixed after 10 minutes. For both flights, in all cases except one, the navigation performance based on VDM is similar to or better (up to 5×) than the navigation based on INS only for the two types of filter (26 and 47 states). In all cases, the maximum navigation errors is lower than 75 meters for the VDM-based navigation with *STIM318* (and the full model), while the median error is maximum ~25 meters. These values are up to 3 times higher for inertial coasting with

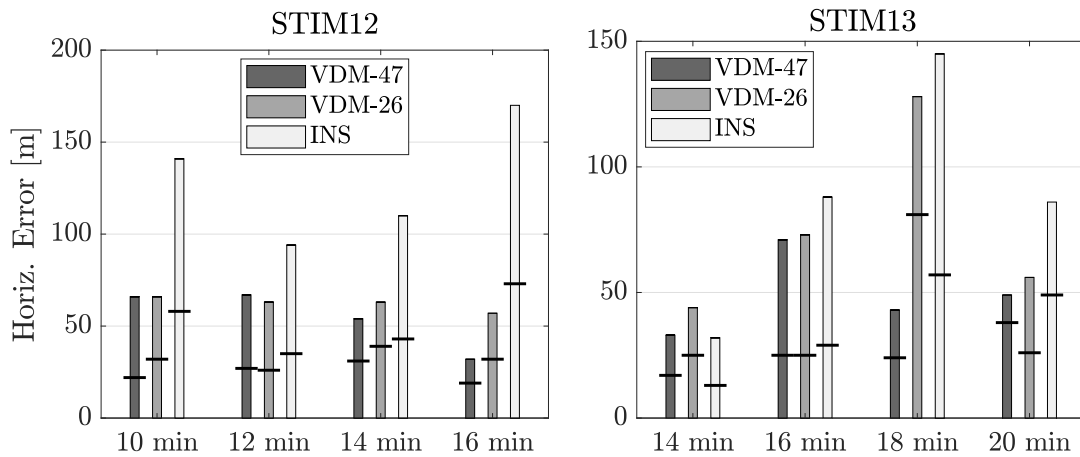


Figure 10.6: Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for flights *STIM_12* and *STIM_13* with 47 and 26 states, and INS

this good quality IMU.

Similar results with fixed parameters (however fixed from takeoff) are given in the Appendix A.5 for the "IGN-GECKO" payloads and flight *AF_i7*. A general conclusion can be formulated: with a correct set of pre-calibrated aerodynamic coefficients, VDM-based navigation gives improved results with respect to INS in case of outages. However, between flights, the battery and payload may be placed in a different location/orientation, the platform may undergo modifications (sensors, mounting), and the weather conditions may differ. Thus, it is best to let the coefficients change slightly between flights, even after pre-calibration.

As the last investigation, instead of having the full model (47) or completely removing the VDM parameters from the states (26 or 27 with scale state *s*), a proposed alternative is to reduce the model by lumping some states together. This is explored in the following.

Online Demonstrator Campaign 2

During this campaign, the GNSS outages trigger from the QGC was correctly configured and VDM-based navigation performance was monitored in real-time versus the INS-based solution. For the flight *STIM_14*, the first GNSS outage after 9 minutes of flights lasts 309 *sec*. The error in position at the end of the outage for the VDM-based navigation solution is ~570 *m* and ~2600 *m* for the INS (Gi iNAV). After the outage, the UAV continues its mission. The second outage starts after 18 minutes and lasts 310 *sec*. The error in position at the end of the outage for the VDM and INS-based navigation solution are ~590 *m* and ~2380 *m*, respectively. The two outages can be seen in Fig. 10.7

For the flight *STIM_15*, an outage of 474 *sec* was triggered after only 6 minutes of flight. The error in position at the end of the outage for the VDM and INS-based navigation solution are ~600 *m* and ~20 *km*, respectively, and the solution is depicted in Fig. 10.8 with a zoomed

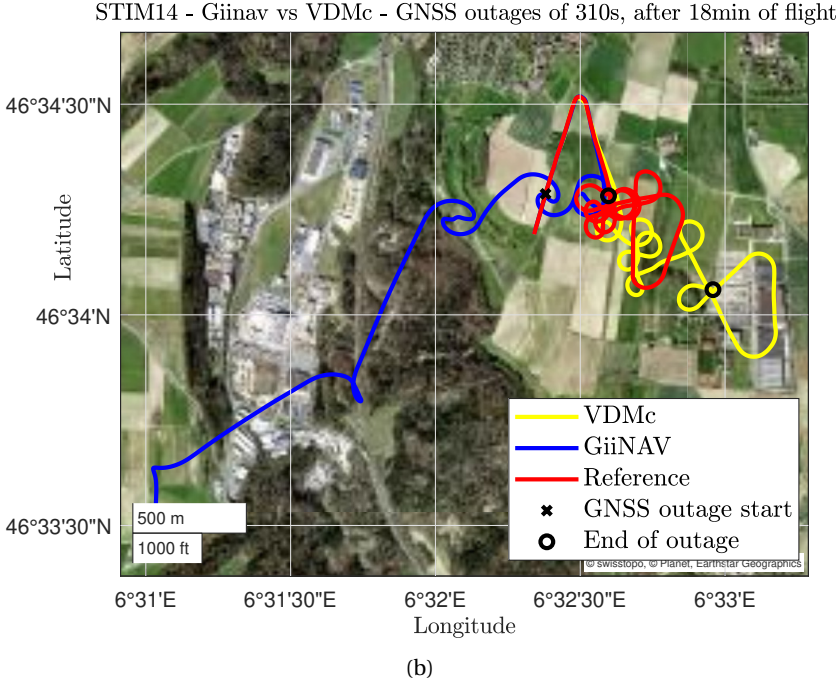
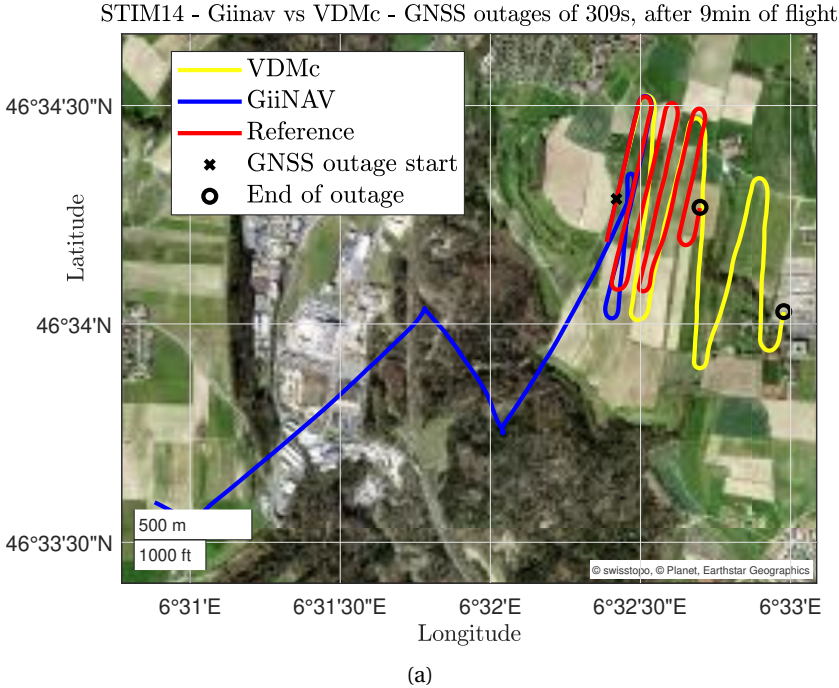


Figure 10.7: *STIM_14*: VDM (yellow) vs INS-based navigation solution (blue) compared with the reference (red) during GNSS outages of about 5 minutes after (a) 9 minutes and (b) 18 minutes

(right) near the reference. For both flights, the partial update during the GNSS outage is enabled. The smooth trajectories are easily distinguishable. The dynamic state reduction after 10 minutes was also enabled for the flights and did not produce any error. The improvement in autonomous navigation when using the VDMc in comparison with Gi iNAV is evident. It is the first time that this type of navigation has been demonstrated in real-time on a small UAV.

10.4 Reduced Model

The reduced model uses the coefficient pairs obtained from the calibration flight *CF_i8* via smoothing and not the ones obtained from simulation (Sec. 8.1, Tab. A.6).

The reduced model is compared first with the full model of *TP2* for the nominal case of GNSS signal reception (100 s after initialization) for the same flights presented in Sec. 9.3.2, including flight *AF_i7*, *AF_i6x* and *AF_i6u*. The differences in positions between both models are less than 0.2 m, which is practically negligible under regular (1 Hz) GNSS updates.

However, the effect of model reduction is more noticeable when the GNSS solution is absent. Taking into account the very same cases as in Fig. 10.3, the differences between both models are plotted in Fig. 10.9. The errors with the reduced filter are more significant in five out of six cases by a factor ranging from 1.2× to 2×. Compared to simulations where both filters performed practically identically, these differences are noteworthy. This may be due in part to a more significant error in attitude, particularly in determining the yaw angle. Although drone guidance aims to fly each line with constant speed and azimuth, the flying envelope of actual tests is undoubtedly more turbulent than that of simulations. Thus, the higher-order coefficients may account for (or absorb some) real or non-modeled effects. For example, the yaw angle is correlated with the coefficient $C_{F_{y,1}}$, “refined” value which may be influenced by the (less correctly) estimated side-slip angle and thus the real wind (\mathbf{x}_w).

From these findings, it can be concluded that the gain in computational efficiency brought by the reduced model comes at the price of slightly worse navigation accuracy in the case of a GNSS outage. The quality of autonomous navigation with the VDM reduced model is still significantly higher than that of the inertial coasting model. In contrast, within the nominal reception conditions of the GNSS signal, these differences are practically insignificant. To maintain the best possible navigation performance in a GNSS-denied environment, it is recommended to use the full model.

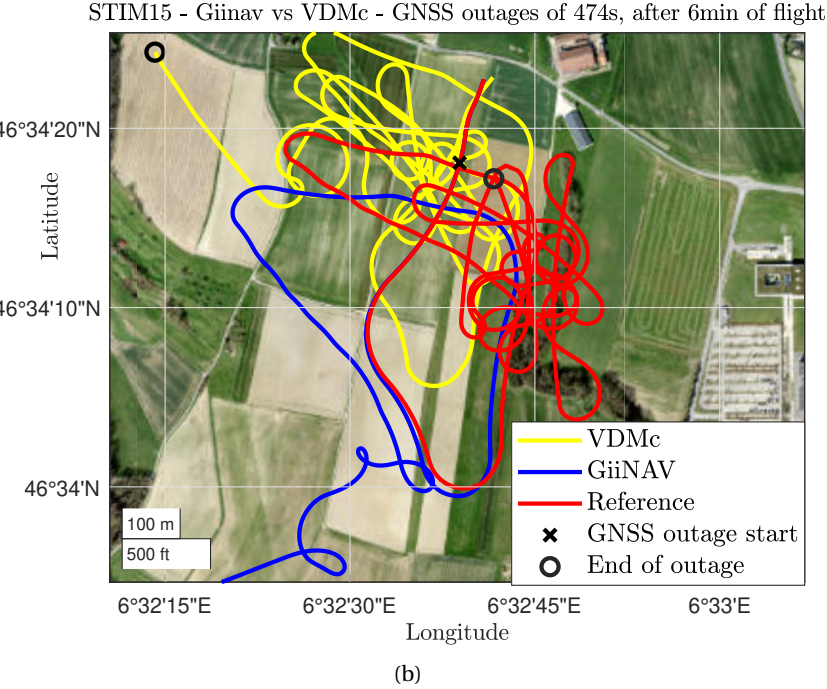
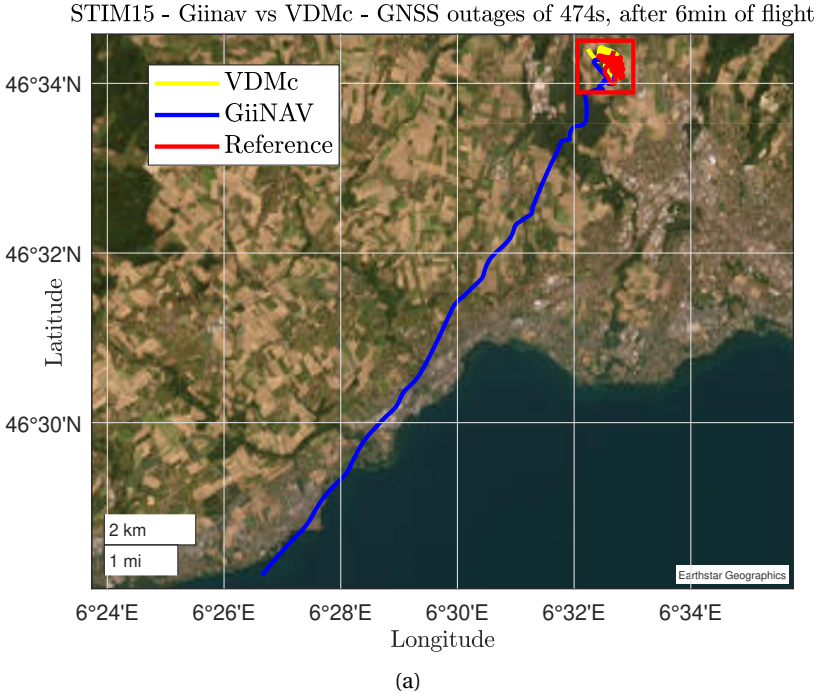


Figure 10.8: *STIM_15*: VDM (yellow) vs INS-based navigation solution (blue) compared with the reference (red) during GNSS outages of about 8 minutes after 6 minutes, zoomed version (b)

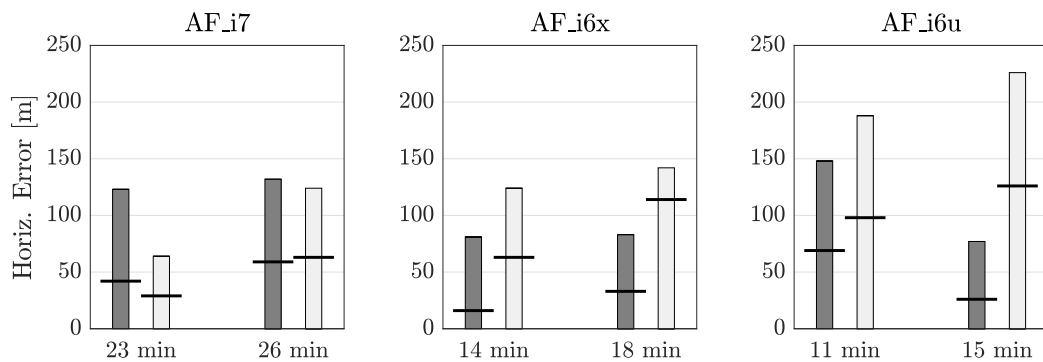


Figure 10.9: Max. and median horizontal errors during 2 minutes of GNSS outages with the full (dark grey) and reduced (light grey) models

Summary

This last chapter has validated theoretical and engineering contributions by evaluating the performance of autonomous VDM-based navigation in different scenarios. First, the WMF calibration methodology has been verified using the obtained coarse coefficients as initial values during validation flights for both platforms. Then, the benefit of the partial update was confirmed during the autonomous phase by reducing the errors and removing erratic jumps in position due to the high update rate of IMU. Subsequently, the real-time autonomous navigation during GNSS outages has been presented for the two online demonstrator campaigns. The outage durations were longer than 5 minutes for the three real-time autonomous tests. The improvement in position at the end of the outages compared to inertial coasting was greater than five times for all tests. Finally, the navigation performance of a proposed reduced model has been compared with the full model during a simulated GNSS outage. It has been revealed to still have a better performance than inertial coasting. In contrast, the full model provided slightly better results, confirming that the coefficients should be refined during the first minutes of a mission to adapt to the potential modifications of the platform geometry and payload.

Conclusion and Recommendations **Part V**

11 Conclusion

This chapter concludes the work presented in this thesis by summarizing the key research findings in relation to the stability and implementation of real-time, VDM-based navigation and discusses its contribution to the growing field of autonomous drone navigation. The limitations of the investigations conducted during the work are reviewed and recommendations are provided for future studies.

11.1 Contributions

11.1.1 Conceptual

Coarse calibration

A coarse estimate of the aerodynamic model parameters was demonstrated to be indispensable for model-based navigation to avoid divergence or catastrophic failure of the navigation filter. In this thesis, a methodology was proposed to **estimate an initial set of aerodynamic parameters from flight data** without the need for external tools or additional setup. The method is independent of platform geometry nor requires prior knowledge of the platform's aerodynamic parameters owing to the linearity of the estimation approach. The parameters obtained from the proposed calibration algorithm have been successfully tested and validated on two different drone geometries during flights that contained simulated GNSS outages.

Fine calibration

Attitude references derived using photogrammetric methods and **optimal smoothing** with high-precision GNSS aiding in the calibration of VDM parameters has been proposed. Coupling these observations with this estimation method enhances the separability of the force and moment parameters and their decorrelation with respect to sensor errors and wind. In turn, it leads to improved autonomous navigation performance. The importance of their joint-usage increases with the lower quality of on-board IMU.

Estimation stability

Steps were proposed to improve the **numerical stability of the VDM filter**. First, a method was developed to identify and re-scale a small subset of state variables. Second, implementing factorization together with partial updates of the filter parameter subset proved to stabilize and improve the phases of initialization and autonomous navigation.

Autonomous navigation

Navigation quality during **GNSS outages** from different platforms and IMU quality validated the benefit of the VDM-based navigation over the kinematic (inertial) approach. The drift of VDM/IMU-based navigation was confirmed to be significantly lower than that of inertial coasting. The maximum positioning error was maintained at 100 *m* or less over 2 minutes for most scenarios and was **less dependant on IMU quality**.

Wind estimation

Wind velocity using model-based navigation is shown to be correctly estimated in simulation and in two real flights where reference observations were available. This opens up new opportunities for in-flight wind estimation without the use of an airspeed sensor and/or to serve as a means of independent performance verification of such sensors.

11.1.2 Engineering

Real-time design

A **real-time VDM-based navigation filter** was implemented on an embedded computer. It was tested using an online demonstrator, and the computational load was analyzed for different filter configurations. A real-time dynamic state reduction mechanism was conceived to adapt the filter size based on the calibration results of the VDM coefficients or to optimize the computational load of the filter based on the available resources. Although navigation performance during simulated GNSS outages was not as efficient when state reduction was applied to the filter as compared to the full model, this solution may be interesting for μ -controller implementations where computational resources are limited.

Time-tagging quality

The sensitivity of quality of control command timestamps for VDM-based navigation was analyzed. While the time-tagging error in the motor data was found to have a negligible influence on navigation error, that associated to the servo control commands was significant. Rapid growth in position error was observed even for delays as small as 10 milliseconds. In this regard, a time-tagging mechanism was implemented to label the flight control input and all sensor observations with a global reference (GPS) to ensure consistency in the navigation filter and mitigate potential hundreds meters of position error in case of long GNSS outages. The remaining real-time errors in the time-tagging were verified to be on the order of 10^{-5} s.

11.2 Limitations

The quality of the estimated aerodynamic coefficients was inferred through VDM-based autonomous navigation with respect to inertial coasting. This test is general and global, yet limits the analysis of the individual coefficient influence on the navigation solution. Due to the non-linear relations between force and moment coefficients, such a test may be valid throughout the flying envelope allowed by the autopilot (maximum banking angles, descent angles).

The replay environment for testing the VDM-based navigation (MATLAB or ROS bags) removes some possible real-time system challenges such as missing or corrupted sensor data, unexpected hardware defects or operating system performance flaws. Although the validation of the real-time implementation was satisfactory, additional testing for robustness is recommended before applying its output to guidance and control inputs of an autopilot.

State-based estimation is highly dependent on the correct setting of stochastic models (initial states and their covariance, process, and measurement noise). Optimal settings may be invalidated due to hardware failure or its operating environment. Again, a fallback strategy for identification and adaptation to such situations should be implemented before closing the loop with the autopilot guidance and control.

11.3 Further work

With respect to the last limitation point, a thorough filter adjustment should be performed. It would involve, but is not limited to, actuator deflection dynamic delays, investigation of propeller speed during kinematics, extensive analysis of wind stochastic properties, the influence of battery voltage on propeller speed versus PWM, possibly employing motor-controller with RPM signal or implementing separate RPM sensor, studies on the IMU noise model on IMUs of lower quality.

The VDM for a particular platform is known, and does not change during a mission, and is implemented as a process model inside the navigation filter. The calibration method based on the observability Gramian presented a change in vector basis to isolated states that are observable. If the eigenvector basis is known a priori (i.e., after pre-calibration), an adequate partial update on selected states could potentially be applied in real-time. The candidate does not know whether such an implementation exists and thus suggests it as an interesting research investigation.

Considering the high frequency of the IMU measurements leading to rapid and constant updates of the states, an interesting investigation would be to leave the partial update in place throughout the flight for the IMU updates, or lower their frequency, and not only during initialization and during detected GNSS outages. The benefit of such an implementation was observed during these two phases; therefore, an additional analysis could be performed to

Chapter 11. Conclusion

determine if navigation performance gains could be realized when applying it under nominal GNSS reception conditions.

A large and general aerodynamic model for the delta wing drone was selected. The results of the coefficient calibration revealed that the model was overfitting the sensor observations. In future work, a modified/simplified model should be tested to better match the physical behavior of the platforms. This will likely improve the general applicability of the model between flights and potentially improve the performance of autonomous navigation.

In the absence of an air sensor (airspeed/pressure sensor) on the platform, real-time observations from the weather station (wind direction and magnitude, air temperature, humidity, and pressure) could be transmitted (via *WiFi* or other communication links) to the VDM system. The impact of weather observation updates on navigation performance could be investigated and interesting research avenues could be identified.

Numerous additional options and improvements to the real-time application VDMc are collected on the *gitlab* repository associated to the project. Future users of the software are welcome to implement these enhancements.

A Additional Analysis and results

A.1 Numerical Conditioning

The implementation follows the methodology suggested in Sec. 4.1. The condition number (eigen values of \mathbf{P}) is initially driven by the variances (diagonal elements) of \mathbf{P}_0 . The calibration flight *CF_i8* (presented in Sec. 8.3.2) is shown as an example to represent the unknown correlation between the states. Fig. A.1(a) shows the order of magnitude ($\in [2; -34]$) of each element of the covariance matrix \mathbf{P}_0 at the initialization of the filter without scaling. The initial

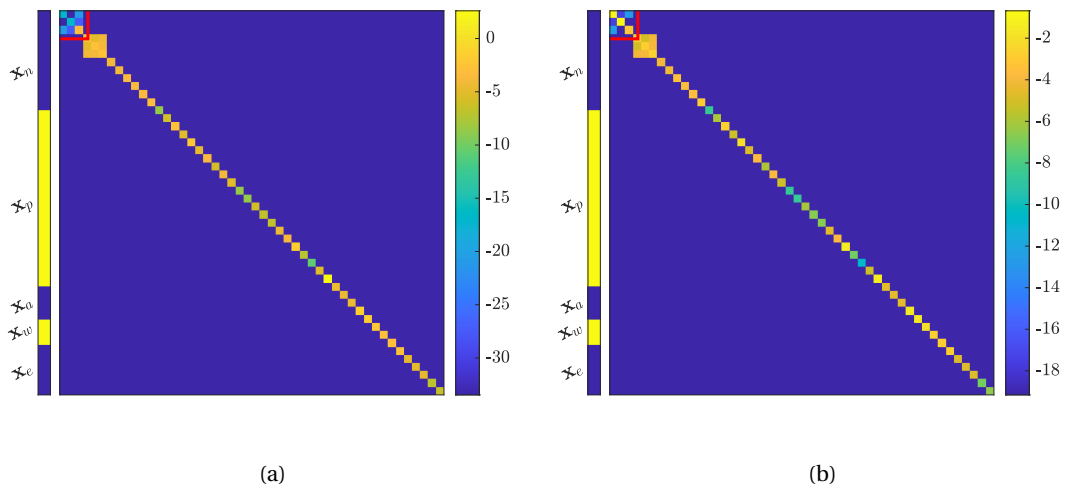


Figure A.1: Order of magnitude (right-axis) of the initial covariance matrix elements \mathbf{P}_0 of a state-vector components (left-axis): a) before, b) after scaling

covariance matrix is quasi-diagonal, with the exception of the position x and velocity v , which are propagated to their corresponding frame/space. The variances for the position in latitude and longitude are small $\approx 10^{-18}$, (inside the red square on Fig A.1(a)) and the covariance between latitude and longitude is even smaller $\approx 10^{-34}$. They are hardly distinguishable from the null elements present off-diagonal within a 64-bit precision architecture and can be wrongly

Appendix A. Additional Analysis and results

interpreted as zeros.

After scaling the states related to horizontal position errors and propeller speed n , the condition number of \mathbf{P}_0 decreases considerably (from 10^{24} to 10^{10}) and the range of the matrix elements is also largely reduced (Fig A.1(b) $\in [0; -18]$). With the previously described scaling of only three variables, the condition number of the matrix S used to compute the KF gain (Tab. 2.1) drops from 10^{13} to 10^2 . With that the GNSS velocity and IMU measurements are no longer potentially numerically problematic, as the related states are already correctly conditioned: GNSS velocity update values correspond to the magnitude of $\approx 10^0$ and IMU of $\approx 10^2$.

During the flight, the condition number of \mathbf{P} with or without scaling of the states stabilizes with a magnitude difference of approximately 10^5 as depicted in Fig. A.2 for a simulated flight of 15 minutes duration [107]

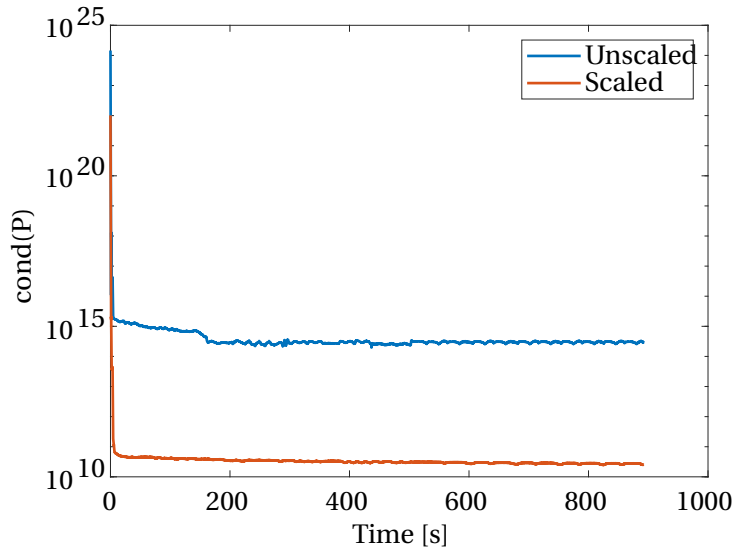


Figure A.2: Condition number of the covariance matrix with and without scaling of lat/long and propeller speed n

leading to less numerical errors during covariance update and propagation.

According to Eq. 4.3 the propagation of round-off errors is reduced by a factor of 10^{10} , the fact of which improves the numerical stability of the matrix inversion. Although a matrix inversion failure is not observed during the empirical tests without scaling in the experiments conducted, this does not prove that it cannot occur during longer flights or under different conditions. To paraphrase [67]: “Even when catastrophic illness does not occur, there is diminished health”. Therefore, even when the original filter performed adequately within the performed course, its scaled version is in much better shape and more stable for real-time application.

A.2 Numerical Integration

The implementation of the KF prediction steps include solving differential equations. There exist numerous numerical differential solvers of different orders. While the first order Euler's numerical method [108] may give good results in some cases, a higher order is desirable particularly for differentiating non-linear equations. The most well known member of the Runge Kutta is called the "the classic Runge-Kutta method of order 4" (Runge Kutta 4 (RK4)) [109]. It is a numerical technique to solve linear or non-linear differential equations reducing the error between the true solution and the numerical approximation. Its implementation in the context of the ESEKF with quaternion is discussed below, especially for attitude propagation.

A.2.1 Attitude with Quaternions

Let $\mathbf{x} \in \mathbb{R}^n$ be the state vector of a system governed by the following differential equation (we here omit the influence of the forcing input \mathbf{u} for simplification):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad (\text{A.1})$$

Let the system initial state at $t = t_0$ be $\mathbf{x}(t_0) = \mathbf{x}_n$, where $t \in \mathbb{R}$ denotes time. RK4 addresses the problem of computing $\mathbf{x}(t_0 + \Delta t)$, given $\mathbf{x}(t_0)$. The general implementation of RK4 is given by the following set of equations:

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_n) \quad (\text{A.2})$$

$$\mathbf{k}_2 = \mathbf{f}\left(\mathbf{x}_n + \mathbf{k}_1 \frac{\Delta t}{2}\right) \quad (\text{A.3})$$

$$\mathbf{k}_3 = \mathbf{f}\left(\mathbf{x}_n + \mathbf{k}_2 \frac{\Delta t}{2}\right) \quad (\text{A.4})$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{x}_n + \mathbf{k}_3 \Delta t) \quad (\text{A.5})$$

The final result is given by a weighted average of the aforementioned operations:

$$\mathbf{x}(t_0 + \Delta t) = \mathbf{x}_n + \left[\frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} \right] \Delta t \quad (\text{A.6})$$

The implement is straightforward in most linear systems. However, in the case of the attitude ODE in quaternion (Eq. 3.54), some extra steps need to be implemented and the operations listed from Eq. A.2 to A.6 need to be adapted. Recalling the ODE for the attitude in quaternion

$$\dot{\mathbf{q}} = \mathbf{q} \otimes \frac{\boldsymbol{\omega}_{lb}^b(\mathbf{x})}{2}, \quad (\text{A.7})$$

where $\mathbf{q} \in \mathbb{S}^4$ and $\boldsymbol{\omega}_{lb}^b \in \mathbb{R}^3$. \otimes represent quaternion product introduced in Sec 3.21.

Let the initial state be given by: $\mathbf{q}(t_0) = \mathbf{q}_n$, $\mathbf{x}(t_0) = \mathbf{x}_n$, therefore $\boldsymbol{\omega}_{lb}^b(t_0) = \boldsymbol{\omega}_{lb}^b(\mathbf{x}_n)$.

Appendix A. Additional Analysis and results

Extending the RK4 Eq. A.2 to A.6 leads to evaluating ω_{lb}^b at each new point $\mathbf{k}_i \in [1-4]$ which represent intermediate rotation rate quantities to integrate, new "dot quaternions". The "quaternion rate" \mathbf{k}_1 is

$$\mathbf{k}_1 = \dot{\mathbf{q}}_1 = \mathbf{f}(\mathbf{x}_n) = \mathbf{q}_n \otimes \frac{\boldsymbol{\omega}_{lb}^b(\mathbf{x}_n)}{2} \quad (\text{A.8})$$

Then, the \mathbf{k}_1 can be integrated for the period $\frac{\Delta t}{2}$ to be added to the initial attitude \mathbf{x}_n to obtain the intermediate state

$$\mathbf{x}_n + \mathbf{k}_1 \frac{\Delta t}{2} = \mathbf{q}_n \otimes \mathbf{q} \left[\frac{\boldsymbol{\omega}_{lb}^b(\mathbf{x}_n) \Delta t}{4} \right] = \mathbf{x}_{k1} \frac{\Delta}{2} \quad (\text{A.9})$$

where $\mathbf{q}[\cdot]$ is the operator that transforms the elements of tangent space $SO(3)$ (Euler angles) to the quaternion space in S^4 as seen in Eq. B.20, and $\mathbf{x}_{k1} \frac{\Delta}{2}$ is an intermediate state. The result will be then used as argument to compute \mathbf{k}_2 as

$$\mathbf{k}_2 = \dot{\mathbf{q}}_2 = \mathbf{f} \left(\mathbf{x}_n + \mathbf{k}_1 \frac{\Delta t}{2} \right) = \mathbf{q}_n \otimes \frac{\boldsymbol{\omega}_{lb}^b \left(\mathbf{x}_{k1} \frac{\Delta}{2} \right)}{2} \quad (\text{A.10})$$

\mathbf{k}_3 and \mathbf{k}_4 are then computed similarly. Finally, when the "quaternion rate" $\mathbf{k}_i \in [1,4]$ are computed, one can evaluate the equivalent rotation using the Eq. A.6 where each term contributes to rotation the states with his own perturbation. The first two quaternions $q_{k1} = \frac{\mathbf{k}_1}{6} \Delta t$ and $q_{k2} = \frac{\mathbf{k}_2}{3} \Delta t$ are computed as

$$\mathbf{q}_{k1} = \mathbf{q} \left[\frac{\boldsymbol{\omega}_{lb}^b(\mathbf{x}_n) \Delta t}{12} \right] \quad (\text{A.11})$$

$$\mathbf{q}_{k2} = \mathbf{q} \left[\frac{\boldsymbol{\omega}_{lb}^b \left(\mathbf{x}_{k1} \frac{\Delta}{2} \right) \Delta t}{6} \right] \quad (\text{A.12})$$

and this can be generalized for the two last intermediate quaternions q_{k3} and q_{k4} and the completed RK4 expression is given by the following relation:

$$\mathbf{q}(t_0 + \Delta t) = \mathbf{q}_n \otimes \mathbf{q}_{k1} \otimes \mathbf{q}_{k2} \otimes \mathbf{q}_{k3} \otimes \mathbf{q}_{k4} \quad (\text{A.13})$$

A.2.2 Small Integration Time

From [110], if the integration time is small enough ($> 50 [Hz]$), a RK4 implementation with a brute-forced quaternion normalization (Eq. A.14) after each time step propagation equals the performance of the implementation proposed previously with quaternion multiplication which guarantees the quaternion unity. Therefore, the different RK coefficients (\mathbf{k}_1 to \mathbf{k}_4) as presented in Eq. A.2-A.5 can be added together and for each evaluation of the propagation function $\mathbf{f}(\mathbf{x}, \mathbf{u})$, and a quaternion normalization is performed with the following quaternion

normalization

$$q_{k+1} \longrightarrow q_{k+1} / \|q_{k+1}\| \quad (\text{A.14})$$

Finally, the propagation of the state in Eq. A.6 requires a final quaternion normalization. The two methods has been tested on a VDM-based navigation system runing at 100 [Hz] using the MATLAB environment and post-processed data from a flight of 33-minutes (*CF_i8* Sec. 6.4) with a simulated GNSS outage of 2 minutes at the end of the trajectory. The difference in position at the end of the two runs is less than 1 [cm].

A.3 VDM vs Inertial Coasting for the Different Payloads

A.3.1 eBeeX with "GECKO" Payload

For the flights *eBeeX_652* and *eBeeX_756*, a total of eight additional GNSS interruptions of 2 minutes are introduced at different times, as summarized in Tab. A.1. The initial coefficients used are obtained with the WMF method with flight *eBeeX_756* and are given in Tab. C.5. The coefficients are kept in the state space, so their values may evolve during the trajectory.

Table A.1: Start times (in minutes) of 2 minute-long GNSS outage after takeoff for both calibration (*eBeeX_756*) and application (*eBeeX_652*) flights

Flights	GNSS outage start time [min]			
<i>eBeeX_652</i>	22	24	27	29
<i>eBeeX_756</i>	18	20	22	24

Horizontal errors (magnitudes) observed during autonomous navigation based on VDM (dark gray) versus INS (light gray) are shown in Fig. A.3. For each outage, the central mark indicates

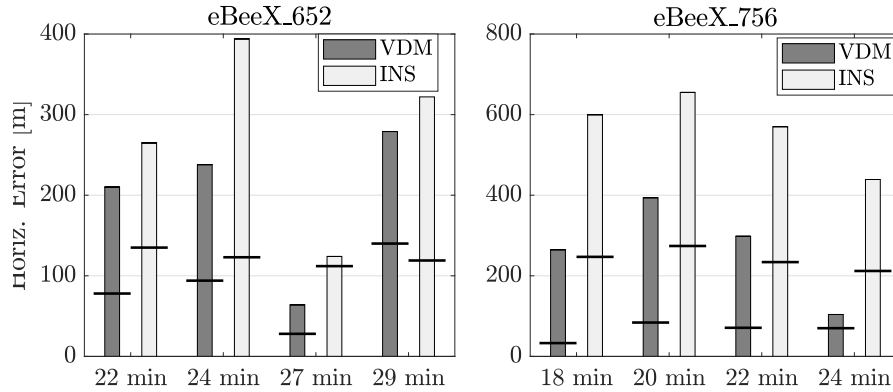


Figure A.3: Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for flights *eBeeX_756* *eBeeX_652* using VDM and INS

the median, and the bar indicates the maximum error. For comparison, the second evaluation is plotted on the same figure for inertial coasting (with barometer height aiding) using the

Appendix A. Additional Analysis and results

identical sensor error model. For both flights and for all GNSS outages, horizontal errors with VDM-based navigation are reduced with respect to free INS by a factor up to $3\times$. The reduction is more visible in the flight *eBeeX_756*, which also serves as the calibration flight.

The results are promising, and the coefficient calibration could be further improved following the methods proposed in Sec. 8.3. However, as mentioned in Sec. 10.1.2 and visible in Fig. A.3, the high number of aerodynamic parameters (44) can result in overfitting and better results could be obtained using model simplifications. However, this assumption could not be further explored due to the limited number of experiments (two flights).

A.3.2 TP2 with "IGN-GECKO" Payload

Similarly for the flights *AP_i7*, *AP_i6x* and *AP_i6u*, four GNSS outages with a duration of 2 minutes are simulated. Their starts are summarized in Tab. A.2. Recall that the parameters are calibrated with flight *CF_i8* and the smoothing method (Sec. 8.3.2 and initial coefficients: Tab. 8.9).

Table A.2: Start times of the 2 minutes GNSS outage within the application flights (in minutes after take-off).

Flights	GNSS outage start time [min]			
<i>AP_i7</i>	17	20	23	26
<i>AP_i6x</i>	12	14	16	18
<i>AP_i6u</i>	10	11	13	14

Fig. A.4 shows the horizontal error statistics observed during autonomous navigation based on VDM and INS. From the total of twelve cases, the reduction of maximum horizontal error

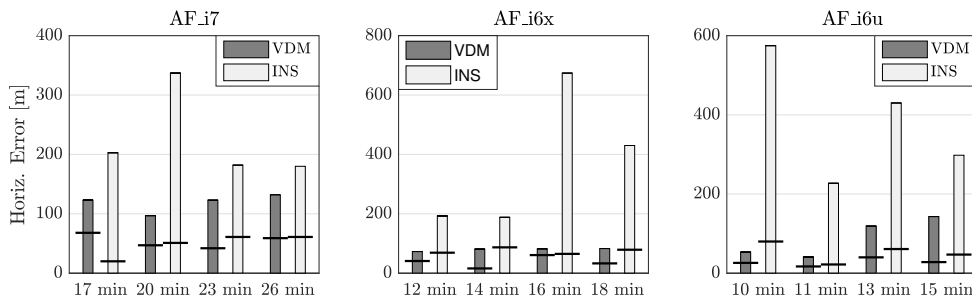


Figure A.4: Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for VDM and INS

for VDM with respect to inertial coasting is very significant on 3 occasions (more than $10\times$), and significant on 3 others (more than $5\times$). In the rest of 6 cases, the improvement varies from $1.5\times$ to $2.5\times$.

For visual comparison, another GNSS outage of 2 minutes is simulated at the very end of the

A.3 VDM vs Inertial Coasting for the Different Payloads

application flights AF_{i7} and AF_{i6x} for both VDM and INS-based navigation. The superior performance of VDM-based navigation with respect to inertial-only navigation in the GNSS-denied environment can be observed by comparing Fig. A.8 and Fig. A.5 for flight AF_{i6x} and AF_{i7}

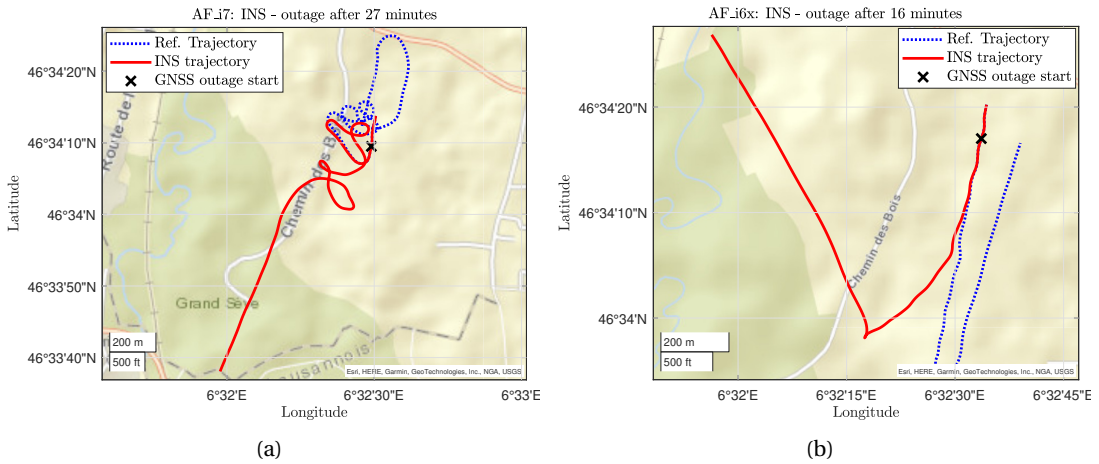


Figure A.5: INS/GNSS based navigation performance under 2-*min* GNSS outage for the flights (a) AF_{i7} , and (b) AF_{i6x}

Furthermore, to observe the improvement in navigation performance, the duration of the first GNSS outage in AF_{i6u} is increased to 6 minutes. Autonomous navigation during this period is detailed in Fig. A.6, for VDM (green), INS (red), and reference (blue). Although the maximum horizontal error in position is $\sim 250\text{ m}$ for VDM, it is $18\times$ larger ($\sim 4.5\text{ km}$) for the inertial coasting case.

A.3.3 TP2 with "SODA-STIM" Payload for the WMF Campaign

The initial coefficients are obtained thanks to the WMF method with CF_{STIM6} . The GNSS outage start times are summarized in Tab. A.3 for the flights considered.

Table A.3: Start times (in minutes) of 2 minute-long GNSS outage after takeoff for $STIM_5$ and $STIM_6$ flights (WMF campaign)

Flights	GNSS outage start time [<i>min</i>]			
	WMF campaign			
$STIM_5$	22	24	27	29
$STIM_6$	22	24	26	28

During the flight AF_{STIM7} , the control commands were only recorded for the first 6 minutes due to autopilot logging problems; therefore, this flight is not used for the following investiga-

Appendix A. Additional Analysis and results

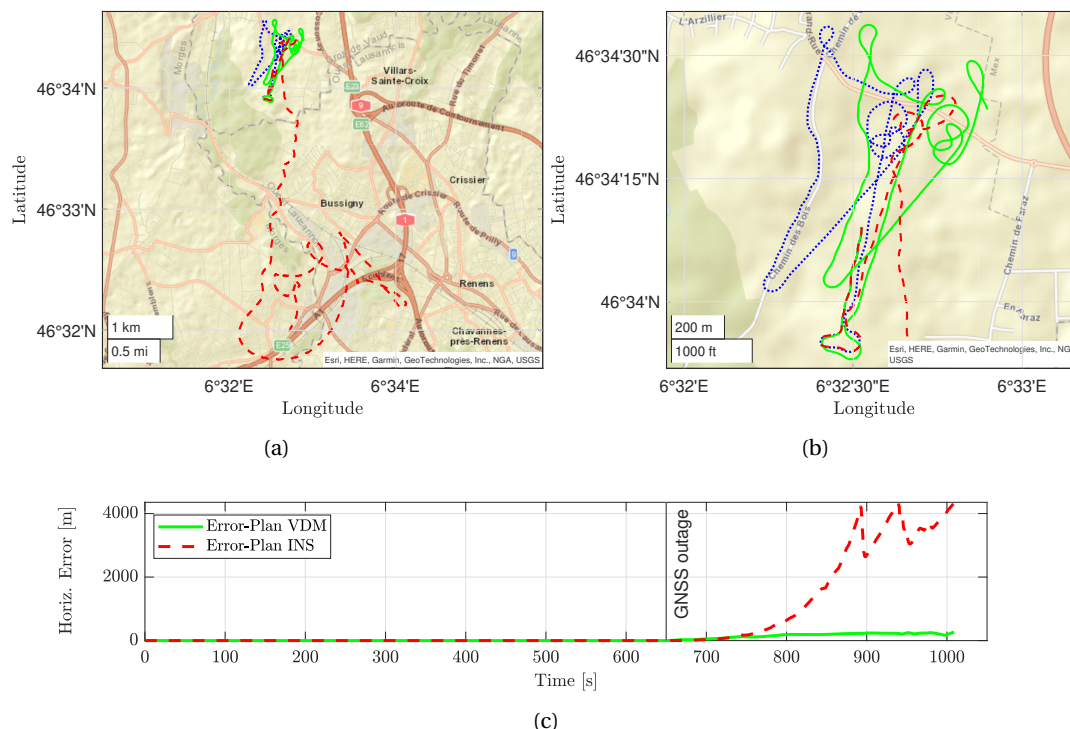


Figure A.6: For flight *AF_i6u*, (a) Last 400 seconds of VDM (green), INS (red), reference (blue) during a GNSS outage of 6 minutes: (a) 2D, (b) 2D-zoomed, (c) error(t)

tions. The horizontal error statistic shown in Fig. A.7 compares the flights of the calibration campaign (*STIM5* and *STIM6*). For flight *AF_STIM5* and *CF_STIM6*, the autonomous VDM-based navigation during the GNSS outage performs better for 75% scenarios than INS-only navigation. As the *STIM318* IMU is of relatively good industrial quality (Tab. C.1), inertial coasting gives reasonably satisfactory results (less $< 100m$ error after 2 minutes of dead reckoning).

The use of a IMU of higher quality (as *STIM-318*) can be useful to calibrate the aerodynamic coefficient of a particular UAV. Then a fleet of similar UAVs can fly with a VDM-based navigation system equipped with a lower-grade IMU while achieving relatively good autonomous navigation under GNSS outage.

A.4 Coefficients with Photogrammetry Aiding

The true values of the VDM parameters remain unknown. As mentioned in [28], some of the estimated aerodynamic parameters may be partially absorbed by random, yet time-correlated inertial errors. The question, therefore, remains how suitable the estimated VDM parameters are for autonomous navigation, especially those determined by a filter without coarse pre-estimation. This is indirectly evaluated by emulating GNSS 2-minute update outage at the end

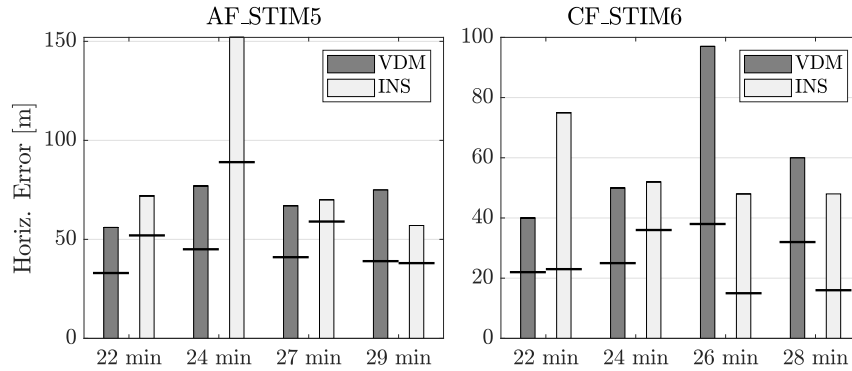


Figure A.7: Horizontal-position errors (max. & median) during repetitive GNSS outages of 2 minutes for VDM and INS for flights *AF_STIM5* and *CF_STIM6*

of the calibration (*CF_i8*) flight, as well as in the two application flights *AF_i7* and *AF_i6X*. The GNSS outages occur in their corresponding flight after 31, 27 and 18 minutes, respectively.

The experimental setup used in the application flights included exactly the same aircraft and payload as is used in the calibration phase (see Sec. 6.4.2). The application flights contain the following differences: a) although some photos are taken, they are no longer used, b) the flights are carried out in the same area but different flight plans are used, so the flight lines differ in length (*AF_i7* - shorter, *AF_i6X* - longer), c) the cm-level relative positioning (PPK) is used only as a reference; hence only stand-alone GNSS position and velocities are used for inertial-based before emulating their absence (the uncertainties for both GNSS modes are summarized in Tab. C.4). Furthermore, the VDM parameters \mathbf{x}_p at the end of the calibration phase in the *CF_i8* flight for both cases (without/with att. ref.) are saved along with their corresponding correlation matrices (\mathbf{P}). They are then used as initial states $\mathbf{x}_p(0)$ and covariance matrices $\mathbf{P}_p(0)$ for the two other application flights, *AF_i6X* and *AF_i7* respectively. The initial uncertainty related to \mathbf{x}_p is increased by 1%, while the initial systematic errors in IMU and the wind are reset to zero with the same uncertainty as in the simulation environment presented in Sec. 8.1.3. By doing so, new calibration phases (however, this time without attitude updates) can adapt the whole parameter set to a new set of sensor errors (inertial) and weather conditions (wind).

Fig. A.8 shows the last 130 seconds of the three trajectories with (i) the reference position drawn as a dashed blue line, the GNSS outage starts are marked with a black cross; (ii) autonomous navigation based on VDM using the priors of the aerodynamic coefficients of *CF_i8* derived without (red) and (iii) with (green) attitude updates. It is clearly evident in the figures that the complementary information from the attitude updates used in the calibration phase of *CF_i8* positively influenced the determination of the aerodynamic coefficients, so that the magnitude and direction of the position drift is mitigated in the application flights.

Tab. A.4 summarizes navigation performance during the emulated GNSS outage for the calibration and the two validation flights. The four elements presented are (i) the horizontal

Appendix A. Additional Analysis and results

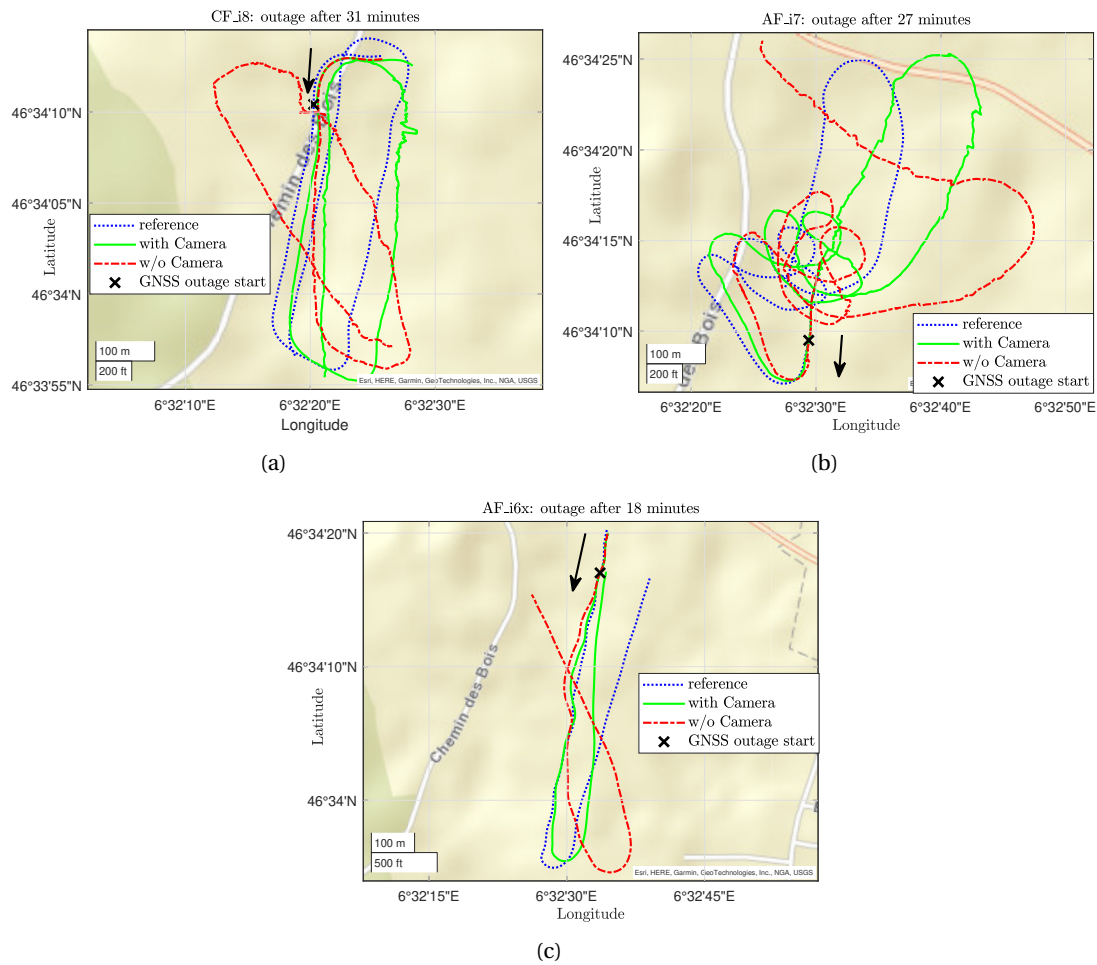


Figure A.8: Comparison during 2 minute GNSS outage of VDM-based navigation performance using (a) the *CF_i8* trained aerodynamic coefficients with camera attitude references (green) or without (red), and the application flights (b) *AF_i7* and (c) *AF_i6X*

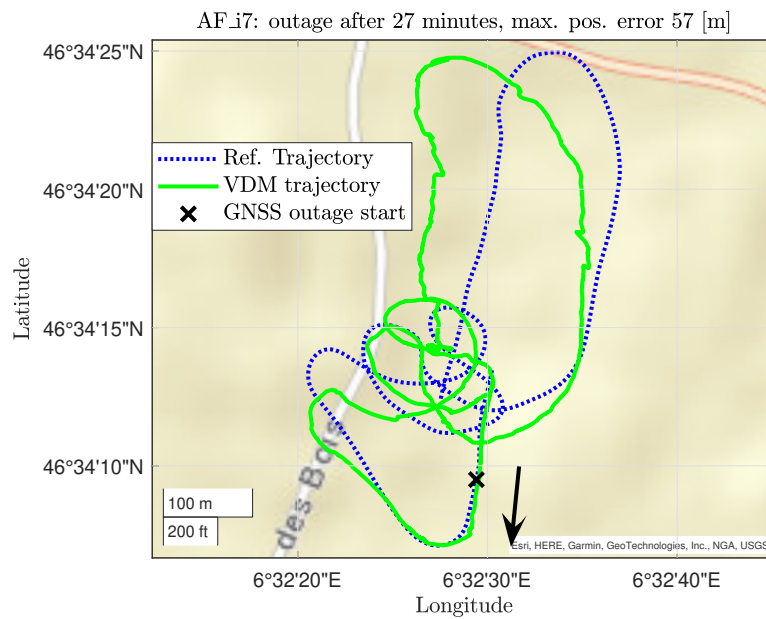
position error at the end of the 2 minutes of GNSS outage, (ii) the root mean square error for the 2D and (iii) 3D positions throughout the period, and (iv) the percentage of time the UAV stays within 150 m of the true trajectory. The latter criterion is useful for the UAV to return close to a safe (home) location where (i) the UAV can land or (ii) the operator can assume manual control. For the set of VDM-coefficients calibrated with the help of attitude aiding, this is achieved in all tested cases for the given prototype. Taking into account the nominal speed of 16 m/s for the given prototype, such UAV could return home within a radius of 150 m without GNSS from a distance of almost 2 km.

Table A.4: Position error at the end of the 2 minute GNSS outage, 2D and 3D position RMSE and % of time with hor. error under 150m

	IGN8		IGN7		IGN6	
	w/o cam	w. cam	w/o cam	w. cam	w/o cam	w. cam
Error end traj. [m]	167	108	421	60.7	275	102
RMSE 2D [m]	97.3	46.9	133.5	62.5	68.9	30.0
RMSE 3D [m]	98.6	47.2	138.14	67.0	79.78	53.3
Hor. error < 150 m [%]	77.7	100	68.1	98.8	88.7	100

A.5 Fixed VDM Parameters

The current model for *TP2* has 47 parameters, among them 21 representing the aerodynamic coefficients. After attitude-aided calibration, the coefficients in the investigation are removed from the system state from the very beginning of the flight, and only a scale factor s embedded in the change in weather conditions is added and estimated. The filter is thus reduced from 47 to 27 states, consisting of the following elements: navigation \mathbf{x}_n , actuator \mathbf{x}_a , scale factor s , wind velocity \mathbf{x}_w and IMU biases \mathbf{x}_e . Flight *AF_i7* is selected to determine the performance of



(a)

Figure A.9: VDM-based navigation with 27 states

autonomous navigation with the reduced filter model, where the aerodynamic coefficients are treated as constants. In the later state of this flight, a GNSS outage of two minutes duration is emulated. The deviation from the reference trajectory is shown in Fig. A.9. Additional tests with fixed parameters are presented for two additional flights *STIM_12* and *STIM_13* in Sec. 10.3.1. However, in these two flights, a 10 minutes refinement is performed before fixing

and removing the aerodynamic coefficients from the states. The dynamic state reduction is discussed in Sec. 9.2.

Note that the Schmidt partial-update for this experiment was not yet implemented and the “roughness” of the trajectory is observable during the outage.

A.6 Aerodynamic Coefficient Refinement Thanks to ‘pose’ Sensor

The following results are taken from [76] where two sets of coefficients are obtained with different methods. One employs the ‘pose’ sensor, and the other method uses regression of selected flight phases to estimate some force coefficients. The second method was implemented by another Ph.D. student from TOPO-EPFL, for which the obtained related results are not presented here. This experiment was carried out before the development of the WMF methodology presented in Sec. 8.2. The initial static aerodynamic coefficients were obtained with wind tunnel experiments, and the dynamic ones with *Tornado*¹. In addition, and even more importantly, it took about one year to have approximately working aerodynamic coefficients due to the complexity of the model and the communication with the industrial partner with whom a project on autonomous navigation was conducted. With the methodology presented in Sec. 5.2, it took two days to adapt the WMF software to *eBeeX* and obtain a set of (working) aerodynamic coefficients!

The VDM-based navigation filter implemented in MATLAB can employ observations coming from high fidelity PP INS/GNSS navigation solution. These measurements helps in refining the value of aerodynamic coefficients, which take part of the estimated parameters. The accuracy of these additional observations are verified to be few *cm* in position, few *cm/s* in velocity and $\sim 0.05^\circ$ for roll and pitch and $0.1^\circ - 0.2^\circ$ for yaw [111]. These observations come from a virtual sensor called ‘pose’ sensor introduced in Sec. 5.3.1. Due to this method, a set of coefficients can be estimated in flight.

Flight with Special Sensors

For the purpose of the PP method, the best flight in terms of wind condition and sensor measurements quality and accuracy (GNSS, IMU, Barometer), data of which were collected on February 20th 2019 (*eBeeX_756*) with a special sensory payload that is presented in Sec. C.4. During the flight, control commands as well as sensors data were recorded to postprocess the measurements with the VDM-based navigation filter, which, apart the aerodynamic coefficients, also estimates the wind direction and intensity with respect to platform’s body frame. For this reason, a condition close to “zero wind” limits the impact of incorrect wind parameters estimation reverberated and compensated in the erroneous estimated aerodynamic coefficients. Such conditions were almost satisfied during the aforementioned flight and therefore this experiment was used to estimate the aerodynamic coefficients of the *eBeeX*

¹<http://tornado.redhammer.se/>

A.6 Aerodynamic Coefficient Refinement Thanks to 'pose' Sensor

platform. The whole flight is depicted in Fig. 6.11(a) showing the GNSS trajectory with carrier-phase differential treatment resulting in position and velocity precision at cm and cm/s level, respectively.

The trajectory resembles a standard photo mission flight plan with two perpendicular flight lines. Circular maneuvers in the middle of the figure were designed to excite the influence of aerodynamic coefficients to improve their estimation at the beginning of the trajectory.

The sets of aerodynamic coefficients obtained are given in Tab. C.5.

A.6.1 VDM-based Autonomous Positioning Performance

Estimating the aerodynamic coefficients and comparing their values does not reveal their intrinsic physical meaning due to their natural correlation. In order to test them at different portion of a trajectory (e.g., after $\sim 10, \sim 20$ and ~ 27 minutes) a situation with loss of GNSS data is simulated and the performance of VDM-navigation is evaluated during 2 minutes.

The results of VDM-based navigation under GNSS outage with both the set of initial coefficients and the two portions of the trajectory are depicted in Fig. A.10. The last 130 s of the trajectory is shown, 120 s of which with autonomous VDM-based navigation (i.e. without GNSS). The reference position is drawn as dashed blue line. The GNSS position and velocity are still available during the first 10 s and this period is represented with red crosses. To appreciate the benefit of VDM-based autonomous navigation over the currently implemented INS-based counterpart in eBeeX, the latter solution is depicted in purple in the left column of Fig.A.10. The maximum horizontal error of using INS-based navigation is ~ 290 m and ~ 190 m for GNSS outage at 10 min and 20 min, respectively.

The maximum error in horizontal position is less than 40 m after 2 min long absence of GNSS positioning, as shown in the first row of Fig. A.10. Tab. A.5 shows the percentage of the time when the autonomously derived position of the drone stays within ± 50 m and 150 m (1 and 3σ). Considering that 1σ is to the norm of empirical accuracy which corresponds to 68% and 40% probability in 1D and 2D, respectively, **the empirical testing meets the specifications of the first quantifiable deliverable^{II}**.

Method / Flight length	PP / 10min	CCE / 10min	PP / 20min	CCE / 20min
1σ (50m)	99.6%	98.7%	64.5%	71.0%
3σ (150m)	100%	100%	100%	100%

Table A.5: Percentage of time within 2 min long autonomous navigation during which the horizontal positioning error stays within the specs represented by 1σ and 3σ . This test is invoked after 10 min and 20 min of flight

^{II}CTI proposal document 25800.1 PFIW-IW

Appendix A. Additional Analysis and results

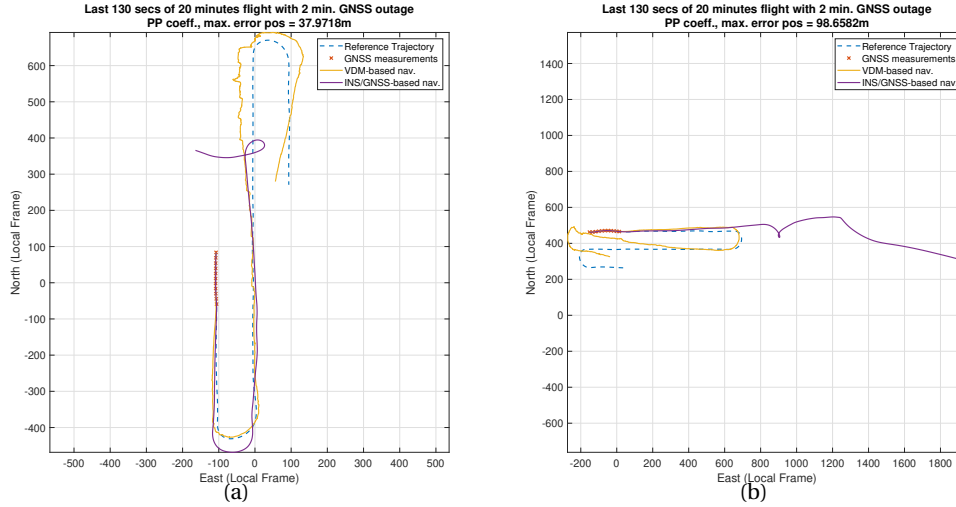


Figure A.10: Last 130 s of VDM-based navigation with 2 min of GNSS outage after 10 min (a) and 20 (b) min of flight with the aerodynamic coefficients estimated with the 'pose' method (Sec. 5.3.1)

A.7 Simulation VDM Parameters Correlations

For a particular trajectory, some VDM parameter pairs highly correlate but not necessarily for another trajectory with different dynamic, i.e., the pair $\bar{c} - C_{M_y\tilde{\omega}_y}$ that converges with a correlation higher than 90% for a ascending straight line and "infinity loop" with altitude changes trajectory but its correlation decreases for a more complex one. Or, the pair $C_{M_{y1}} -$

Level Flight	Asc. Straight Flight	'8 loops' with alt. changes	Combination
$C_{M_{y1}} - C_{M_y\alpha}$ 93.6%	$C_{M_{y1}} - C_{M_ye}$ 90.4%	$C_{M_ye} - C_{M_y\alpha}$ 90.2%	$C_{F_z\alpha} - C_{F_x\alpha^2}$ 90.9%
	$\bar{c} - C_{M_y\tilde{\omega}_y}$ 96.7%	$D - C_{F_{T1}}$ 90.6%	$C_{M_x\tilde{\omega}_x} - C_{M_x\tilde{\omega}_z}$ 90.9%
	$C_{M_{y1}} - C_{M_y\alpha}$ 97.8%	$C_{M_{y1}} - C_{M_y\alpha}$ 92.2%	$C_{M_z\delta_r} - C_{M_z\beta}$ 95.2%
		$C_{M_z\delta_r} - C_{M_z\beta}$ 95.4%	$S - C_{F_z\alpha}$ 97.1%
		$\bar{c} - C_{M_y\tilde{\omega}_y}$ 96.1%	$C_{M_{y1}} - C_{M_y\alpha}$ 97.5%
		$S - C_{F_z\alpha}$ 96.4%	$C_{M_ye} - C_{M_y\alpha}$ 97.5%
		$C_{M_ye} - C_{M_y\tilde{\omega}_y}$ 96.7%	$C_{M_{y1}} - C_{M_ye}$ 97.9%
			$C_{M_x\alpha} - C_{M_x\beta}$ 98.2%
			$D - C_{F_{T1}}$ 99.4%

Table A.6: Highly correlated VDM parameter pairs for particular trajectories

C_{M_ye} , which is well correlated for both the climbing line and the combination of trajectories, is not for the level and "infinity loop" ones. With these pairs, the highly correlated VDM parameters are good candidates to be re-unified within the Kalman filter into a common state. Such "parameter lumping" may prevent potential singularities and save processing power by reducing the states vector size and related matrices. Also, it may better to eliminate the geometrical parameters \bar{c} , S from unknowns and obtain their values from CAD models or other

measurements.

A.8 Experimental Validation of Wind Estimation

A comparison of the 2D horizontal wind estimate by the WMF method using an airspeed sensor and the measurements of the portable weather station using the test flight *STIM3* is given. The wind is assumed to be static throughout the flight. Fig. A.11 shows the effect of 2D static wind compensation on the measured-estimated airspeed comparison.

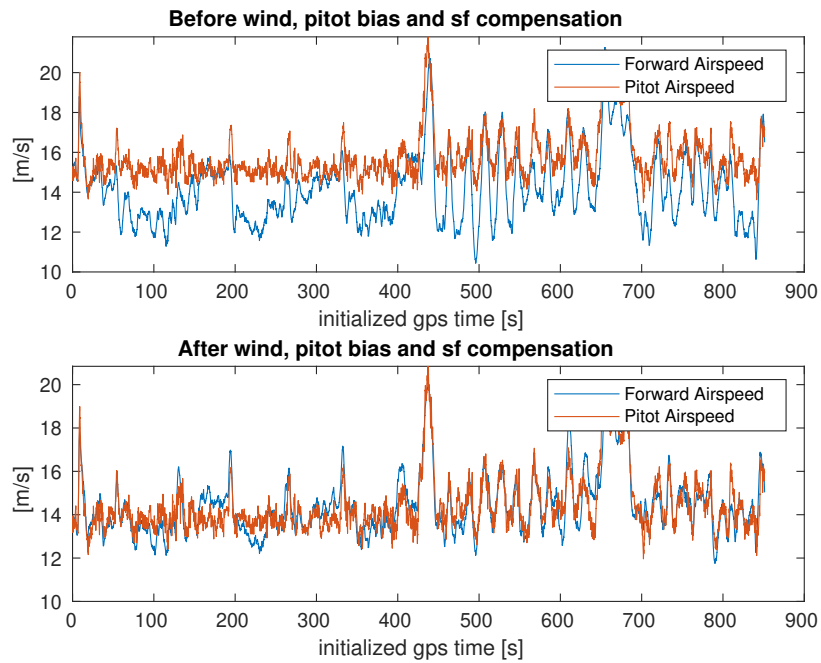


Figure A.11: Anemometer heading resolution

Portable Weather Station		Pitot Tube 2D wind estimation	
<i>magnitude</i> [m/s]	<i>direction</i> [deg]	<i>magnitude</i> [m/s]	<i>direction</i> [deg]
1.69	19	1.7	15

Table A.7: 2D static wind estimation comparison - mean values over flight duration

While the wind magnitude matches to a cm accuracy the wind direction results are slightly different, this is mainly due to the anemometer’s coarse heading resolution and the anemometer calibration. Indeed, the initial calibration of the anemometer might be subject to direction bias error according to the operator’s manual precision when initializing the North direction.

A.9 WMF TP2 and eBeeX Forces and Moment Residual with Flight STIM6

A.9.1 TP2 - Moment Residuals

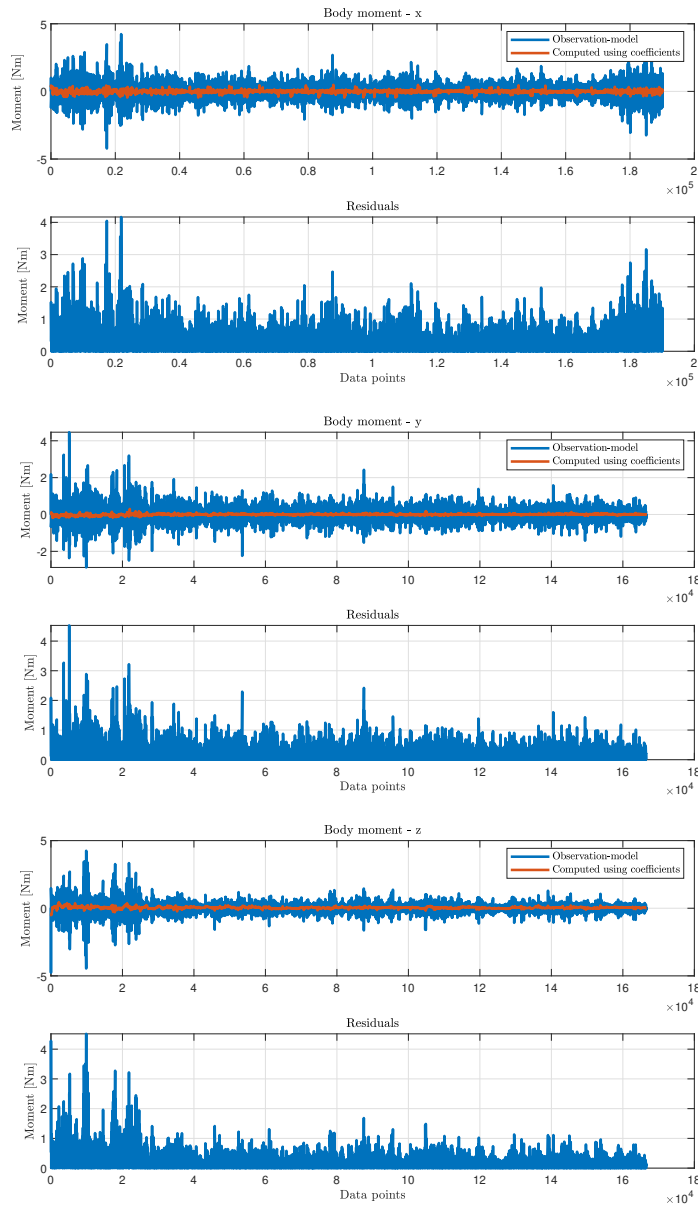


Figure A.12: Moment residual for each axis

A.9.2 TP2 - Force Residuals

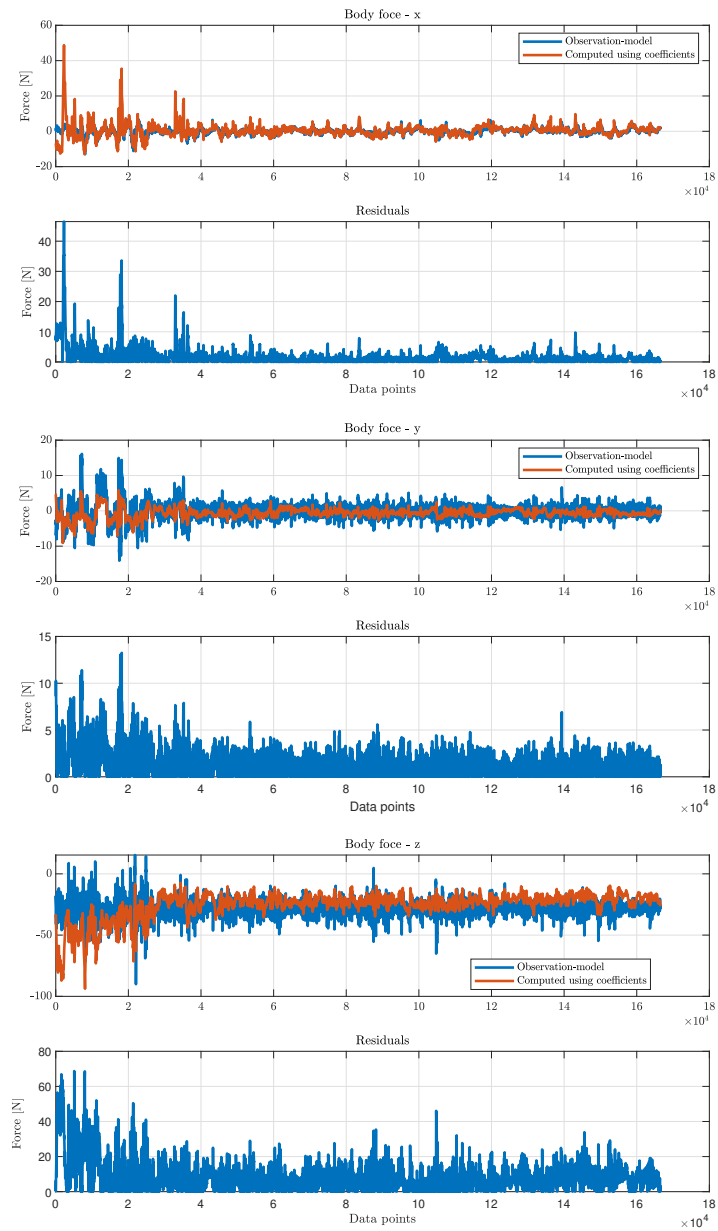


Figure A.13: Force residual for each axis

A.9.3 eBeeX - Moment Residuals

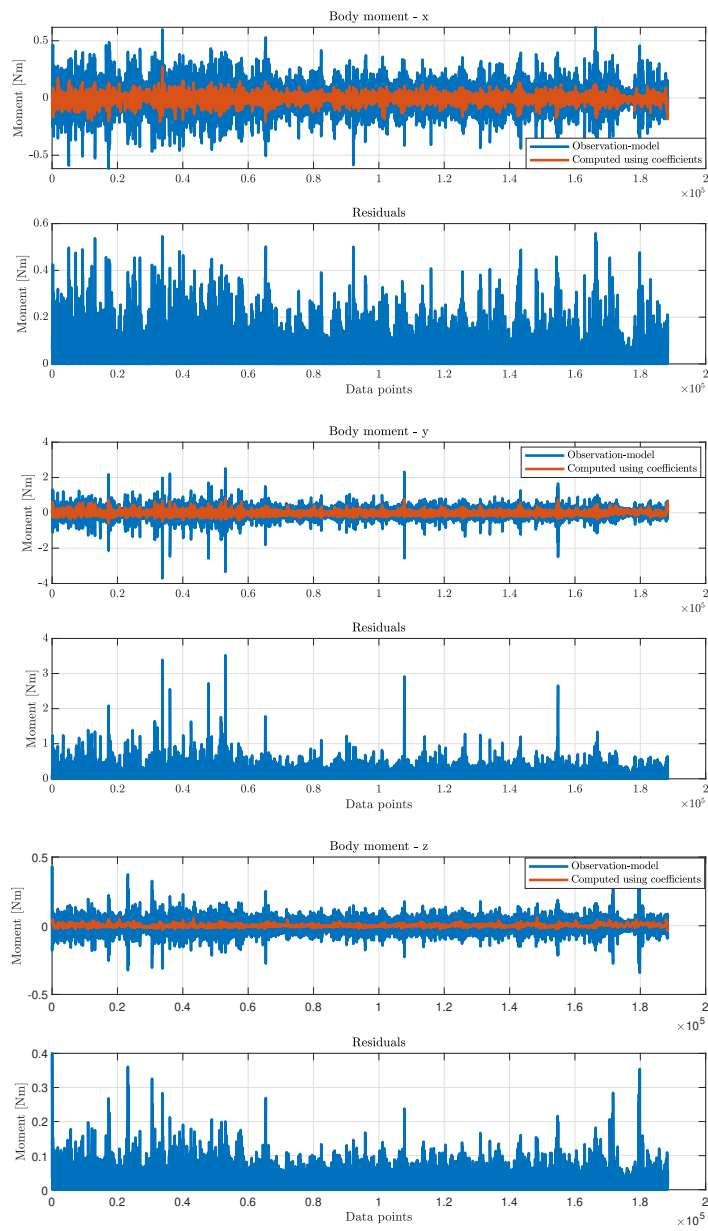


Figure A.14: Moment residual for each axis

A.9.4 eBeeX - Forces Residuals

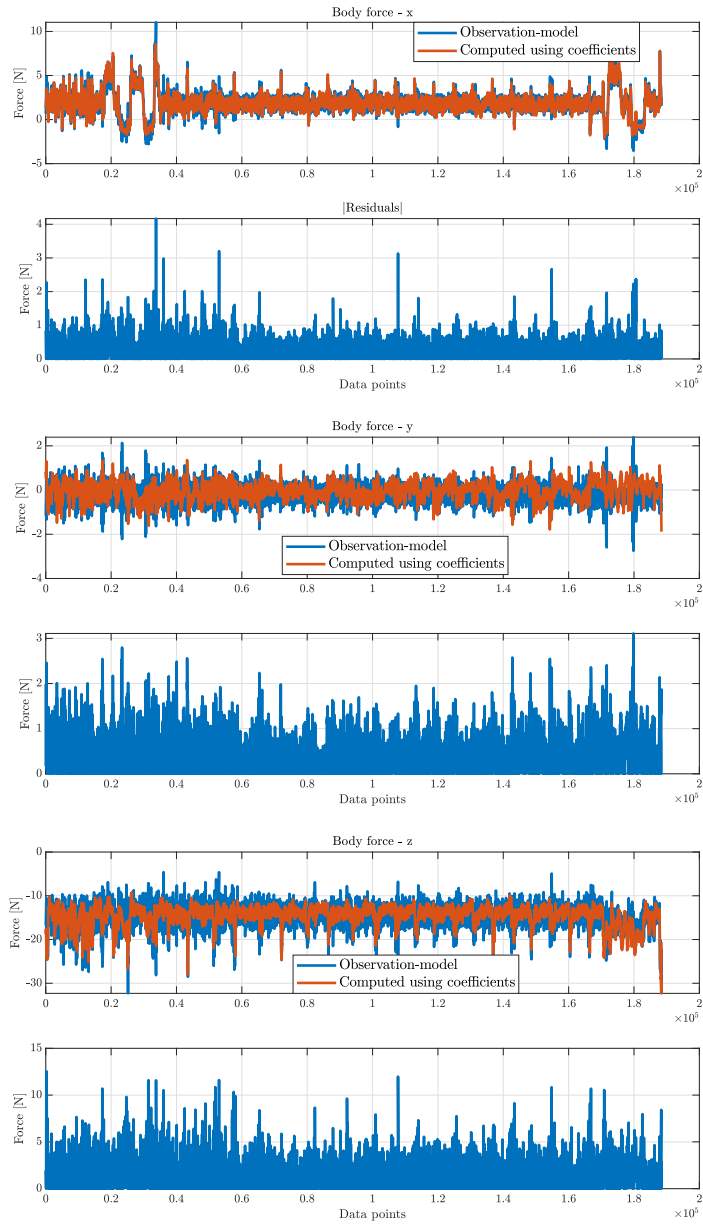


Figure A.15: Force residual for each axis

B Theoretical supplements

B.1 Static and Dynamic Pressure Conversion

B.1.1 From Airspeed to Ground Velocity

To obtain the true or ground speed of an aircraft based on the Pitot tube measurements, a transformation sequence may be required depending on the sensor type.

The airspeed without correction is called **Indicated Airspeed** (IAS) and is given from the dynamic pressure (Eq. 3.67) and the total air pressure measured with the Pitot tube (Eq. 3.80) as

$$IAS = \sqrt{\frac{2(p_{tot} - p)}{\rho_0}} \quad (B.1)$$

where ρ_0 is the air density at the International Standard Atmosphere (ISA) at sea level (ISO 2533:1975, 1.225 kg/m^3) [55].

The **Calibrated Airspeed** (CAS) is obtained by correcting the IAS for both the sensor errors and misalignment. While the first error depends on the quality of the sensor, the second is caused by the variation of the local wind velocity around the sensor and the probe not constantly pointing towards the forward body axis due to the aircraft's motion.

While the IAS and CAS are based on the Bernoulli's equation assuming the incompressibility of the air, the latter can be further corrected by considering the compressibility effects in the air and is called the **Equivalent Airspeed** (EAS). For small UAVs, this effect is negligible as it becomes significant for velocity $> 0.3 \text{ Mach}$ which is more than 350 km/h . The formulas can be found in [112].

At last, the **True Airspeed** (TAS) is obtained by scaling the EAS (or CAS in the case of small

Appendix B. Theoretical supplements

UAVs) with the local air density ρ

$$TAS = EAS \text{ or } CAS \times \sqrt{\frac{\rho_0}{\rho}} \quad (\text{B.2})$$

where ρ can be obtained as a function of the local weather conditions (air pressure, air temperature, dew point or relative humidity [112, 113]) which can be obtained from a local meteorological station, a radio broadcast METeorological Aerodrome Report (METAR) or a local weather station C.5.

The norm of the TAS $\|\mathbf{V}\|$ and the norm of the **Ground Speed** (GS) $\|\mathbf{v}\|$ are equivalent if the aircraft's surrounded wind is null and, therefore, $\mathbf{V} = \mathbf{v}$ from Eq. 3.45.

B.1.2 From Static Pressure to Altitude

The barometric altitude h is computed as

$$h = h_b + \frac{R \cdot T \cdot \ln\left(\frac{P_h}{P_0}\right)}{-g_0 \cdot M} \quad (\text{B.3})$$

where h_b is a bias depending on the weather conditions, $R = 8.31432 \left[\frac{\text{J}}{\text{mol} \cdot \text{K}} \right]$ is the universal gas constant, $g_0 = 9.80665 \left[\frac{\text{m}}{\text{s}^2} \right]$ the gravitational acceleration constant, $T[\text{K}]$ is the standard temperature at sea level, $P_h[\text{Pa}]$ is the measured static pressure with the sensor, $P_0 = 102325[\text{Pa}]$ the standard sea level pressure, and $M = 0.0289644[\text{kg/mol}]$ the molar mass of Earth's dry air. h_b can be determined using the altitude measured from a GNSS receiver as performed in Sec. D.3.3.

B.2 Extended Kalman Filter

B.2.1 From Continuous to Discrete KF - Matrix Derivation

- **from $\mathbf{F}(t)$ to Φ** : Consider the Taylor expansion of the system state $\mathbf{x}(t)$ at time t

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \dot{\mathbf{x}}(t - t_0) + \frac{1}{2!} \ddot{\mathbf{x}}(t - t_0)^2 + \frac{1}{3!} \dddot{\mathbf{x}}(t - t_0)^3 \dots \quad (\text{B.4})$$

From Eq. 2.15, $\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t)$ and $\ddot{\mathbf{x}}(t) = \mathbf{F}(t)\dot{\mathbf{x}}(t) = \mathbf{F}^2(t)\mathbf{x}(t)$. Therefore, the Taylor expansion can be written as

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \mathbf{F}(t - t_0)\mathbf{x}(t_0) + \frac{1}{2!} \mathbf{F}(t - t_0)^2 \mathbf{x}(t_0) + \frac{1}{3!} \mathbf{F}(t - t_0)^3 \mathbf{x}(t_0) \dots \quad (\text{B.5})$$

$$\mathbf{x}(t) = \left(1 + \mathbf{F}(t - t_0) + \frac{1}{2!} \mathbf{F}(t - t_0)^2 + \frac{1}{3!} \mathbf{F}(t - t_0)^3 + \dots \right) \mathbf{x}(t_0) \quad (\text{B.6})$$

$$\mathbf{x}(t) = e^{\mathbf{F}(t - t_0)} \mathbf{x}(t_0) \quad (\text{B.7})$$

therefore $\Phi = e^{\mathbf{F}(t-t_0)}$. For software implementation, the *exponential* function is expensive. Therefore, a first order approximation, $\Phi = I + \mathbf{F}(t - t_0)$ can be used. Other approximations can be found in Appendix. B.2.2

- **from $\mathbf{Q}(t)$ to \mathbf{Q}_k** : The derivation of covariance \mathbf{Q}_k is performed via further integration of Eq. 2.15 for the noise $\mathbf{w}(t)$ as following the discrete time definition of the standard model in Eq. 2.17. It leads that

$$\mathbf{x}_{k+1} = e^{\mathbf{F}(t_{k+1}-t_k)} \mathbf{x}_k + \int_{t_k}^{t_{k+1}} e^{\mathbf{F}(t_{k+1}-\tau)} \mathbf{G}(\tau) \mathbf{w}(\tau) d\tau \quad (\text{B.8})$$

$$= \Phi(t_{k+1}) \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \Phi(t_{k+1} - \tau) \mathbf{G}(\tau) \mathbf{w}(\tau) d\tau \quad (\text{B.9})$$

where by identification \mathbf{w}_k equals

$$\mathbf{G}_k \mathbf{w}_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) \mathbf{G}(\tau) \mathbf{w}(\tau) d\tau \quad (\text{B.10})$$

As for the continuous time system, Q_k is the covariance of the noise $\mathbf{G}_k \mathbf{w}_k$. By the definition of the covariance

$$Q_k = \text{cov}(\mathbf{G}_\tau \mathbf{w}_\tau, \mathbf{G}_s \mathbf{w}_s) \quad (\text{B.11})$$

$$= E[\mathbf{G}_\tau \mathbf{w}_\tau \mathbf{w}_s^T \mathbf{G}_s^T] + \underbrace{E[\mathbf{G}_\tau \mathbf{w}_\tau] E[\mathbf{G}_s \mathbf{w}_s]}_{0 \text{ because } E[w_k] = 0 \forall k \in \mathbb{R}} \quad (\text{B.12})$$

$$= E \left[\int_{t_k}^{t_{k+1}} \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) \mathbf{G}(\tau) \mathbf{w}(\tau) \mathbf{w}^T(s) \mathbf{G}^T(s) \Phi^T(t_{k+1}, s) d\tau ds \right] \quad (\text{B.13})$$

$$= \int_{t_k}^{t_{k+1}} \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) \mathbf{G}(\tau) E[\underbrace{\mathbf{w}(\tau) \mathbf{w}^T(s)}_{\mathbf{w}(k) \stackrel{\text{iid}}{\sim} N(0, \sigma^2) \forall k}] \mathbf{G}^T(s) \Phi^T(t_{k+1}, s) d\tau ds \quad (\text{B.14})$$

$$= \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) \mathbf{G}(\tau) \mathbf{Q}(\tau) \mathbf{G}^T(\tau) \Phi^T(t_{k+1}, \tau) d\tau \quad (\text{B.15})$$

When \mathbf{F} and \mathbf{Q} are both time invariant and the sampling period $\Delta t = t_{k+1} - t_k$ is small, the solution can be approximated to

$$\mathbf{Q}_k = \mathbf{G} \mathbf{W} \mathbf{G}^T \Delta t \quad (\text{B.16})$$

other approximation are given in Appendix. B.2.2

B.2.2 Transition Matrix and Covariance Noise Update Approximation

The covariance noise propagation $\mathbf{Q}_k = \underbrace{\mathbf{G} \mathbf{W} \mathbf{G}^T}_q \Delta t$ and the transition matrix $\Phi = I + \mathbf{F}(t - t_0)$ were approximated with the first order. Higher order approximation for Φ and \mathbf{Q}_k are given in Tab. B.1

Table B.1: Higher order approximation of Φ and \mathbf{Q}_k

Order	Φ
2	$I + \mathbf{F} * \Delta t + 0.5 * \mathbf{F} * \mathbf{F} * \Delta t * \Delta t$
3	$I + \mathbf{F} * \Delta t + 0.5 * \mathbf{F}^2 * \Delta t^2 + \mathbf{F}^3 * \Delta t^3 / 6$
4	$I + \mathbf{F} * \Delta t + 0.5 * \mathbf{F}^2 * \Delta t^2 + \mathbf{F}^3 * \Delta t^3 / 6 + \mathbf{F}^4 * \Delta t^4 / 24$
	\mathbf{Q}_k
2	$q * \Delta t + 0.5 * (\mathbf{F} * q + q * \mathbf{F}^T) * \Delta t * \Delta t$
3	$q * \Delta t + 0.5 * (\mathbf{F} * q + q * \mathbf{F}^T) * \Delta t^2 + (\mathbf{F} * q * \mathbf{F}^T + 0.5 * \mathbf{F}^2 * q + 0.5 * q * \mathbf{F}^{T2}) * \Delta t^3 / 3$
4	$q * \Delta t + 0.5 * (\mathbf{F} * q + q * \mathbf{F}^T) * \Delta t^2 + (\mathbf{F} * q * \mathbf{F}^T + 0.5 * \mathbf{F}^2 * q + 0.5 * q * \mathbf{F}^{T2}) * \Delta t^3 / 3 + (0.5 * \mathbf{F} * q * \mathbf{F}^{T2} + 0.5 * \mathbf{F}^2 * q * \mathbf{F}^T + 1/6 * \mathbf{F}^3 * q + 1/6 * q * \mathbf{F}^{T3}) * \Delta t^4 / 4$
“Van Loan”	$\begin{bmatrix} A = [-FG * W * G.' \\ 0 * FF.' * dt \end{bmatrix}$ $B = e^A, B12 = B(1 : n, n + 1 : 2 * n), B22 = B(n + 1 : 2 * n, n + 1 : 2 * n)$ $\Phi = B22^T, \mathbf{Q}_k = \Phi * B12$

B.3 Additional Material on Angles

B.3.1 ODE for Attitude in Euler

The attitude ODE in Euler angles θ_b^j is derived in a similar manner as that for quaternions. The rotation quantity to be applied to the attitude θ_b^l is the same as for the quaternion case: ω_{lb}^b . However, ω_{lb}^b can not be added directly to the current attitude θ_b^l to perform the rotation. The orientation vector of Euler angles do not exist in Euclidean space [114]. Let the attitude vector be

$$\theta = \begin{bmatrix} r \\ p \\ y \end{bmatrix} \quad (\text{B.17})$$

The Euler angle rates of change $\dot{\theta} \approx \omega_{lb}^b$ are equivalent to the cascaded rotation around each axis at initial orientation θ of the angular change following the rotation order $\mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3$. Using intermediate frames for the rotation around each axis, ω_{lb}^b is equivalent as

$$\omega_{lb}^b = \begin{bmatrix} \dot{r} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_1(r) \begin{bmatrix} 0 \\ \dot{p} \\ 0 \end{bmatrix} + \mathbf{R}_1(r) \mathbf{R}_2(p) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(p) \\ 0 & \cos(p) & \sin(r) \cos(p) \\ 0 & -\sin(r) & \cos(r) \cos(p) \end{bmatrix} \begin{bmatrix} \dot{r} \\ \dot{p} \\ \dot{y} \end{bmatrix} = (R_\theta)^{-1} \dot{\theta} \quad (\text{B.18})$$

Inverting the matrix $(R_\theta)^{-1}$ allow to rewrite

$$\dot{\theta} = R_\theta \omega_{lb}^b = \mathbf{R}_\theta \left(\omega_{ib}^b - \mathbf{R}_l^b \omega_{il}^l \right) = \mathbf{R}_\theta \left(\omega_{ib}^b - \mathbf{R}_l^b \left(\omega_{ie}^l + \omega_{el}^l \right) \right) \quad (\text{B.19})$$

which is the Euler attitude ODE. Note that Eq. B.19 shows that roll rate \dot{r} , pitch rate \dot{p} , and yaw rate \dot{y} are not equal to the angular rate measured by the gyroscope measurements ω_{ib}^b .

B.3.2 Euler Angle and Quaternion Conversion

The quaternion operations have multitude of advantages. However, they lack a direct representation for human to understand. Therefore, in many systems, the computations use quaternions but the results are presented using Euler angles which can be directly interpreted for orientation. The conversions between the two representations are given hereafter.

B.3.3 Euler Angle to Quaternion

As mentioned previously, a rotation around one axis in \mathbb{R}^3 equals half the rotation in \mathbb{H} . Therefore, a rotation vector $v = \phi \mathbf{u}$ be a rotation of ϕ around the axis u can be mapped to the exponential as in Eq. 3.23 to

$$\mathbf{q} = \text{Exp}(\phi \mathbf{u}) = e^{\phi \mathbf{u}/2} = \cos \frac{\phi}{2} + \mathbf{u} \sin \frac{\phi}{2} = \begin{bmatrix} \cos(\phi/2) \\ \mathbf{u} \sin(\phi/2) \end{bmatrix} \quad (\text{B.20})$$

and define the transformation rotation vector using Euler angle to quaternion. If a sequence of rotation is defined (Eq. 3.3), and the rotation vector (v) = $\phi \mathbf{u}$ is used three times for each axis in \mathbb{R}^3 and the Euler angles representation can be seen as three sequential rotations

$$\mathbf{q} = \begin{bmatrix} \cos(\alpha/2)\cos(\beta/2)\cos(\gamma/2) + \sin(\alpha/2)\sin(\beta/2)\sin(\gamma/2) \\ \sin(\alpha/2)\cos(\beta/2)\cos(\gamma/2) - \cos(\alpha/2)\sin(\beta/2)\sin(\gamma/2) \\ \cos(\alpha/2)\sin(\beta/2)\cos(\gamma/2) + \sin(\alpha/2)\cos(\beta/2)\sin(\gamma/2) \\ \cos(\alpha/2)\cos(\beta/2)\sin(\gamma/2) - \sin(\alpha/2)\sin(\beta/2)\cos(\gamma/2) \end{bmatrix} \quad (\text{B.21})$$

B.3.4 Quaternion to Euler angle

The conversion from quaternion to Euler angle is the inverse operation of Eq B.21 and is given here directly

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix} \quad (\text{B.22})$$

When implementing the \arctan operation in a system, one should be certain that it returns the result between $[-\pi; \pi]$.

C Hardware

C.1 Inter-Devices Communication

The necessary communication architecture is realized by several hardware and software components. The links between systems and devices are described below.

C.1.1 GNSS Receiver to Autopilot (GNSS to AP)

The GNSS receiver is connected to the PixHawk via a serial link (A on TP1, B on TP2) at 38400 bps over which it delivers number of messages^I at 5 Hz. The autopilot software running on the flight controller is extended (6.3.2) to decode the binary protocol used by the receiver (GREIS^{II}) and to make available the encoded information to the rest of the autopilot components on μ ORB topics (e.g., `gps_vehicle_position`). The GNSS receiver communicates its timescale to the AP board via PPS at 5 Hz that is received on the RC13 servo input. The reception of these pulses is monitored by modified AP software.

C.1.2 GNSS Receiver to Gecko4NAV / SentiBoard

The GNSS receiver is connected to the Gecko4Nav via a similar serial link than the autopilot (B on TP1, A on TP2) with the same baudrate and messages. These are forwarded to the embedded computer without any processing (Sec. C.1.3).

C.1.3 Gecko4NAV / SentiBoard to Embedded Computer (IMU to PC)

The Gecko4NAV board transmits its data (together with received GNSS data) via USB to the embedded Computer. The firmware of FPGA on Gecko board has been updated to publish data over USB every time it receives PPS from GPS. The GNSS receivers (Javad or Topcon) are

^I`em,/dev/ser/a,jps{/RT,/GT,/PG,/VG,/SI,/XA,/EE}:{0.2}`

^{II}https://javad.com/downloads/javadgnss/manuals/GREIS/GREIS_Reference_Guide.pdf

Appendix C. Hardware

configured to provide PPS at 5 Hz (to this board as well as to AP), thus, the data are available on the USB port 5 times a second.

Gecko4Nav board is interfaced with the ROS environment through the NavServer which broadcasts the IMU and GPS measurements within the ROS environment.

In the case of the "SODA-STIM" payload, the SentiBoard transmits the data to the embedded computer in real time through the USB port. The NavServer can access the data as they arrive, parse them, and format them in ROS messages to be published on the dedicated topics (IMU0, GPS0 and PITO). These are then used for the other ROS nodes as presented in Fig. 7.1.

C.1.4 Image Acquisition (AP to CAM to PC to GNSS)

This communication is related to image acquisition and is governed as a nested command that triggers the camera according to a flight plan and gets the image-GPS-time in a chain of events depicted in Fig. C.1. The AP generates the PWM and sends it to the computer via



Figure C.1: Image acquisition events

serial port. A system service called `soda.service` catches the pulse and generates the TRIG signal to the camera. The camera then opens the shutter to acquire the image and at the same time outputs a EV signal to the GNSS receiver that saves the time of image acquisition. The flight-plane is produced by the user according to desired criteria (e.g. zone extend, image overlap, etc.) at the GCS and uploaded to the autopilot. The images are stored on the camera SD card from which the images are collected later on manually. The dedicated system service manages the settings of the camera, monitors the trigger channel from the autopilot, and supervises the image acquisition.

C.2 Sensor

C.2.1 IMU Error Statistic

STIM318

ADIS-16475

Error Type	Notation	Value [Units]
Gyro bias	b_G	0.3 [deg/h]
Gyro white noise	$\sigma_{G_{WN}}^{PSD}$	0.15 [deg/ \sqrt{h}]
Acc. bias	b_A	3 [μg]
Acc. white noise	$\sigma_{A_{WN}}^{PSD}$	7.5 [mg/ \sqrt{Hz}]

Table C.1: STIM318 noise characteristics

Error Type	Notation	Value [Units]
Gyro bias	b_G	2 [deg/h]
Gyro white noise	$\sigma_{G_{WN}}^{PSD}$	2 [deg/ \sqrt{h}]
Acc. bias	b_A	3.6 [μg]
Acc. white noise	$\sigma_{A_{WN}}^{PSD}$?? [mg/ \sqrt{Hz}]

Table C.2: ADIS-16475 noise characteristics

NavCHIP

Error Type	Notation	Value [Units]
Gyro bias	b_G	720 [deg/h]
Gyro correlated noise	$\sigma_{G_{GM1}}^{PSD}$	0.0028 [deg/s/ \sqrt{Hz}]
	$1/\beta_G$	200 [s]
Gyro white noise	$\sigma_{G_{WN}}^{PSD}$	0.18 [deg/ \sqrt{h}]
Acc. bias	b_A	8 [mg]
Acc. correlated noise	$\sigma_{A_{GM1}}^{PSD}$	0.05 [mg]
	$1/\beta_A$	200 [s]
Acc. white noise	$\sigma_{A_{WN}}^{PSD}$	50 [mg/ \sqrt{Hz}]

Table C.3: NavChip noise characteristics

C.2.2 Stochastic Models for GNSS Noise

C.3 TP2

C.3.1 Geometric Parameters

Control Surfaces

The control surfaces of a standard fixed-wing aircraft are composed of four or more elements and are shown in Fig. C.2 for one platform used for this research (TP2). More details on the platform and its payload are given in Sec. 6.1 and Sec. 6.2, respectively. The right (green) and left (red) aileron deflection δ_a angles are with opposite sign to create a roll moment. The elevator δ_e create a pitching moment to provoke a *nose up* or *nose down* behavior. The rudder δ_r deflection will create a yaw moment to move the nose of the plane to the right or the left in

Table C.4: Typical uncertainty values for SPP and PPK GNSS solution

GNSS mode	Position error (m)		Velocity error (cm/s)	
	Horizontal	Vertical	Horizontal	Vertical
SPP	1	2	0.15	0.3
PPK	0.03	0.05	0.06	0.1

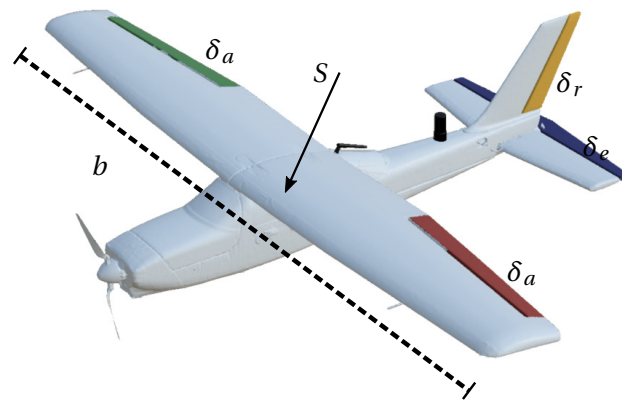


Figure C.2: Flight control surfaces, wing span b and surface S of a fixed-wing UAV: *TOPOPlane version 2*. The dense mesh of the platform was created with the help of the Laboratory of Intelligent System scanner system.

the horizontal plan. The three rotation angles roll, pitch and yaw representing the rotation between the local and the body frame and known as the attitude of the UAV were introduced in Sec. 3.3.4. Any control surface deflection will create a moment around several body axes in a complex manner.

Main Cord

The main cord \bar{c} is the imaginary line joining the leading edge and the trailing edge and is depicted in Fig. C.3. The angle between the projected wind and the cord is defined as the

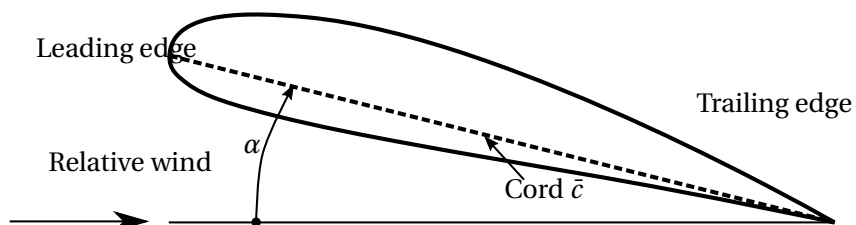


Figure C.3: The cord \bar{c} and the angle of attack α

angle of attack α .

Wing Span and Surface

The wing span b is the distance from one wingtip to the other one and the wing surface S is defined as the surface generating lift which is the main wing hosting the two ailerons as presented in Fig. C.2. The horizontal stabilizer positioned at the tail of the aircraft counters the moment due to the lift force being generally not applied exactly at the aerodynamic center of the UAV. A trim acting directly to the elevator deflection δ_e or to an additional control surface is used to reduce this moment in order to fly leveled when no input is given to the control surfaces. Note that a right horizontal trim can also be applied to the rudder to counter the yaw effect caused by the propeller rotation (torque, p-factor, gyroscopic precession, and spiraling slipstream [115]).

Propeller

The propeller diameter D measures the length of the propeller attached to the motor as presented in Fig. C.4. There is no mention of the angle of incidence, shape, and width of the

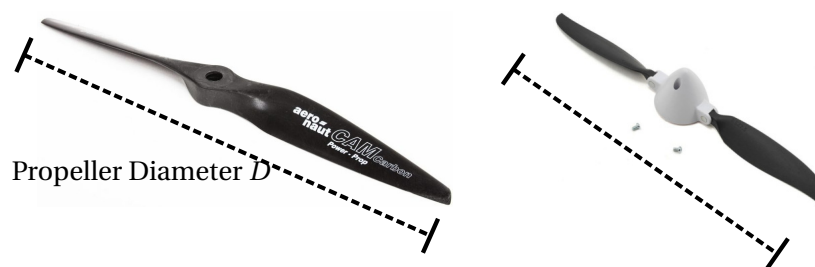


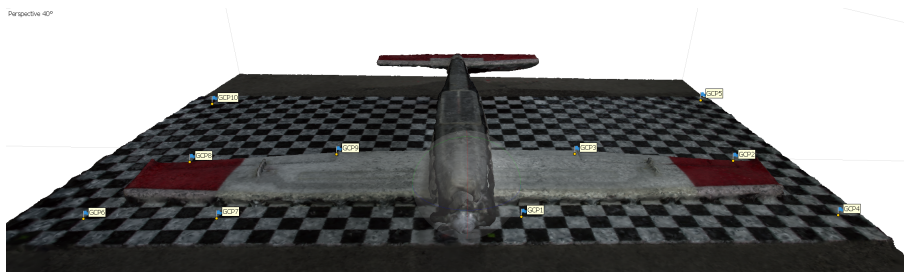
Figure C.4: Propeller diameter D for (left [116]) a full blade, and (right [117]) an half-folding propeller

blade in the VDM. The coefficient(s) C_T related to the thrust force as presented in Tab. 3.1 will absorb the non-modeled aspects of the blade. A more complex propeller model is proposed in [118] but is not used for this research.

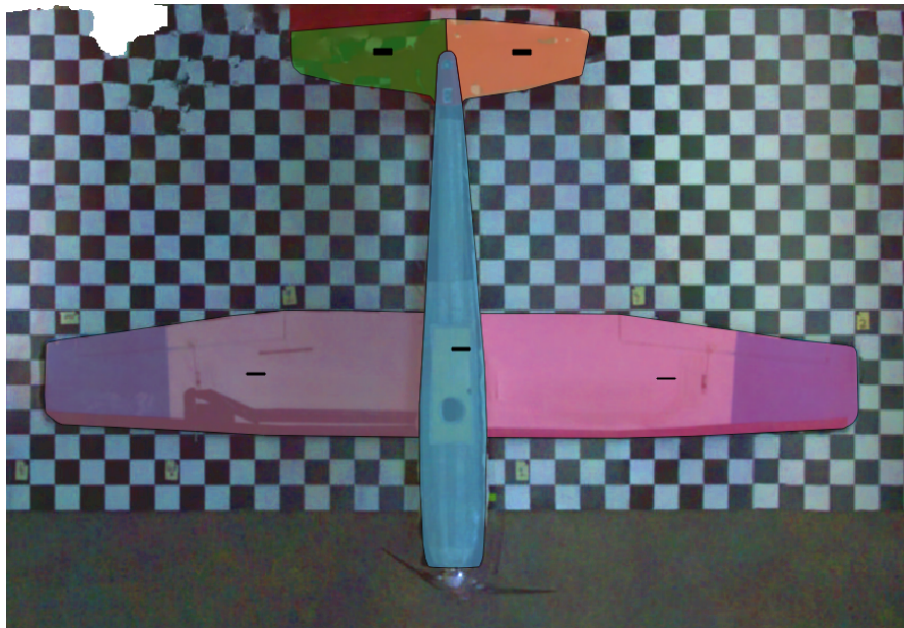
C.3.2 Wind Surface Determination

Among a few other coefficients, the wing area of a UAV is considered as a known constant when computing forces and moments in VDM-based navigation. So far within the project, the UAV wing surface was estimated at approximately. Recently, close-range imagery was collected and used to obtain a higher confidence estimation of the wing surface coefficient.

- Both UAVs were laid bottom-up on a surface of regular spacing (chessboard) where all crossings have known coordinates (Fig. C.5).
- Converging images of the surface and plane were taken at 1.5-2.0 m, with an equivalent



(a)



(b)

Figure C.5: A 3D point cloud reconstruction (a) and orthophoto with surface measurements (b) of the wing surface.

focal length of 50 mm. A 3D point-cloud of the imaged object was reconstructed using a professional photogrammetric software^{III} as shown in Fig. C.5 (a).

- The ortho-rectified nadir view was exported to another software to determine the surfaces of the different wing-areas, as depicted in Fig. C.5 (b).
- Sum of the surfaces provided the total wing surface with a confidence of a few cm^2

The precisely determined wing surfaces were corrected in the current VDM implementation. The two planes are based on the same design, and their respective surfaces are practically equivalent. Although the direct influence of more precise physical parameters, such as wing surface, on the VDM-navigation performance is complex to verify, such improvements will

^{III}Metashape from Agisoft <https://www.agisoft.com/>

enable a more realistic representation of the UAV to be used within the VDM framework and improve our ability to perform such verification.

C.4 eBeeX

C.4.1 "eBee-GECKO" Payload

The support was designed with vibration dampeners to isolate the payload from the engine resonance. The payload was connected to eBeeX via a USB cable through a custom connector powering the board and forwarding the receiver GNSS data along with a PPS that synchronizes the IMU clocks with the GPS time. These IMUs are *NavChip* from InterSense-Thales^{IV} (the same IMU as for the payload "SODA-GECKO" presented in Sec. 6.2). The sensors of each IMU were individually calibrated in the lab for improved performance.

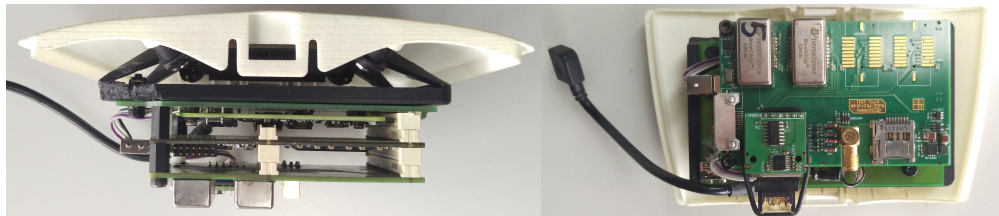


Figure C.6: 3D printed support placed in the eBeeX payload bay with a special board (Gecko4Nav) containing 2 MEMs-IMUs as seen from front (left) and bottom (right).

^{IV}<https://www.intersense.com/navchip>

C.4.2 eBeeX Aerodynamic Coefficients

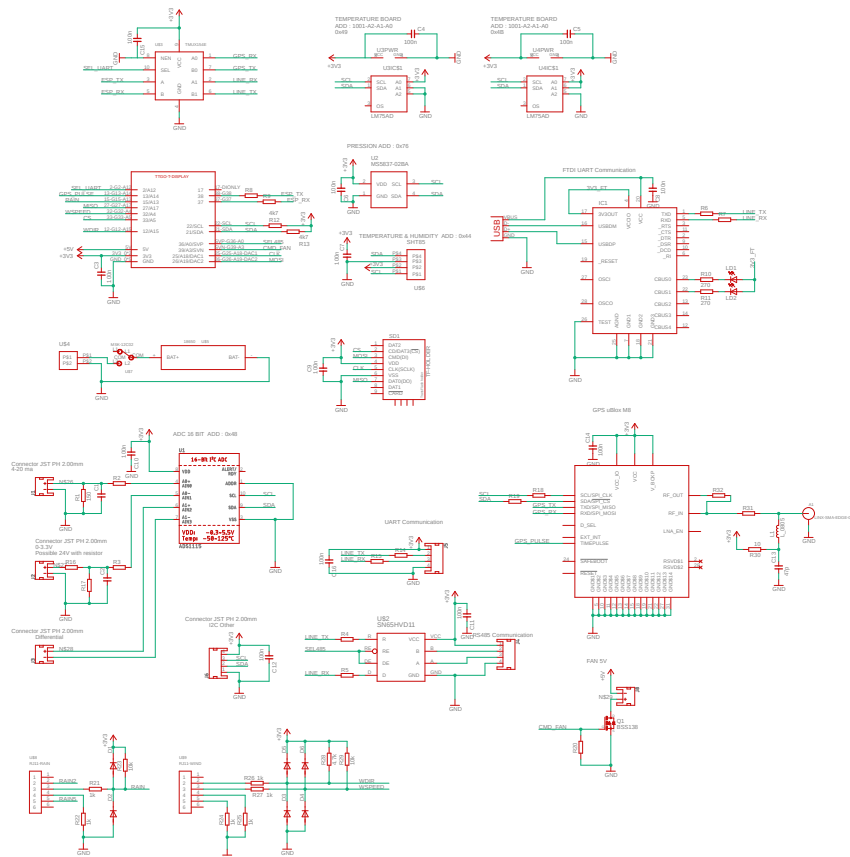
The significant difference between the two methods can be explained by (i) the WMF method estimate the moment and the force coefficients in two separate estimators whereas the 'pose' sensor method estimates the full state-space; (ii) the WMF method does not have any initial values whereas the second method is initialized with some coefficients obtained from wind tunnel and CFD, and others set to zero. In both cases, the model used is over-parametrized, and the analysis of individual coefficients is complex and was not performed during this thesis.

Force coefficients			Moment coefficients		
	WMF	'pose' sensor		WMF	'pose' sensor
C_{F_T0}	0.0056	0.0041	$C_{M_x\beta}$	-0.005	-0.0283
C_{F_T1}	-0.1361	-0.0664	$C_{M_x\beta^2}$	-0.018	-0.0372
C_{F_T2}	3.975	0.527	$C_{M_x\delta a}$	0.0123	0.00089
C_{F_x0}	-0.1479	0.0332	C_{M_xp}	0.0817	-0.0857
$C_{F_x\alpha}$	0.0043	0.351	C_{M_xq}	-0.1602	0.0305
$C_{F_x\alpha 2}$	-0.972	0.015	C_{M_xr}	0.0401	0.18
$C_{F_x\beta}$	-4.03E-06	-0.0441	C_{M_y0}	0.015	0.195
$C_{F_x\delta e}$	0.0152	0.000481	$C_{M_y\alpha}$	-0.249	-2.01
C_{F_xp}	-0.0834	0.0408	$C_{M_y\alpha 2}$	0.829	3.9
C_{F_xq}	-0.2977	0.195	$C_{M_y\delta eUp}$	0.0124	0.00036
C_{F_xr}	0.02715	-8.14e-05	$C_{M_y\delta eDo}$		0.00411
$C_{F_y\beta}$	0.4233	0.0598	C_{M_yp}	0.0524	-0.186
$C_{F_y\beta^2}$	-0.4033	-0.0007	C_{M_yq}	4.086	-0.368
$C_{F_y\delta a}$	-0.0400	0.00102	C_{M_yr}	-0.246	0.712
C_{F_yp}	-0.0172	0.043	$C_{M_z\beta}$	0.005	0.0102
C_{F_yq}	-0.8098	-0.0241	$C_{M_z\beta^2}$	0.0137	0.0347
C_{F_yr}	-0.0062	0.00261	$C_{M_z\delta a}$	-0.001	0.000158
C_{F_z0}	-0.2939	0.165	C_{M_zp}	-0.004	-0.0235
$C_{F_z\alpha}$	1.0974	0.906	C_{M_zq}	0.0269	-0.00924
$C_{F_z\alpha 2}$	-3.666	-0.0633	C_{M_zr}	0.004	-0.0323
$C_{F_z\beta}$	-0.068	-0.263			
$C_{F_z\delta e}$	0.9168	-0.0212			
C_{F_zp}	-0.026	0.621			
C_{F_zq}	3.815	2.73			
C_{F_zr}	-0.543	-0.199			

Table C.5: Coefficients obtained with the WMF and 'pose' sensor method

C.5 Weather Station

C.5.1 Weather Station Schematic

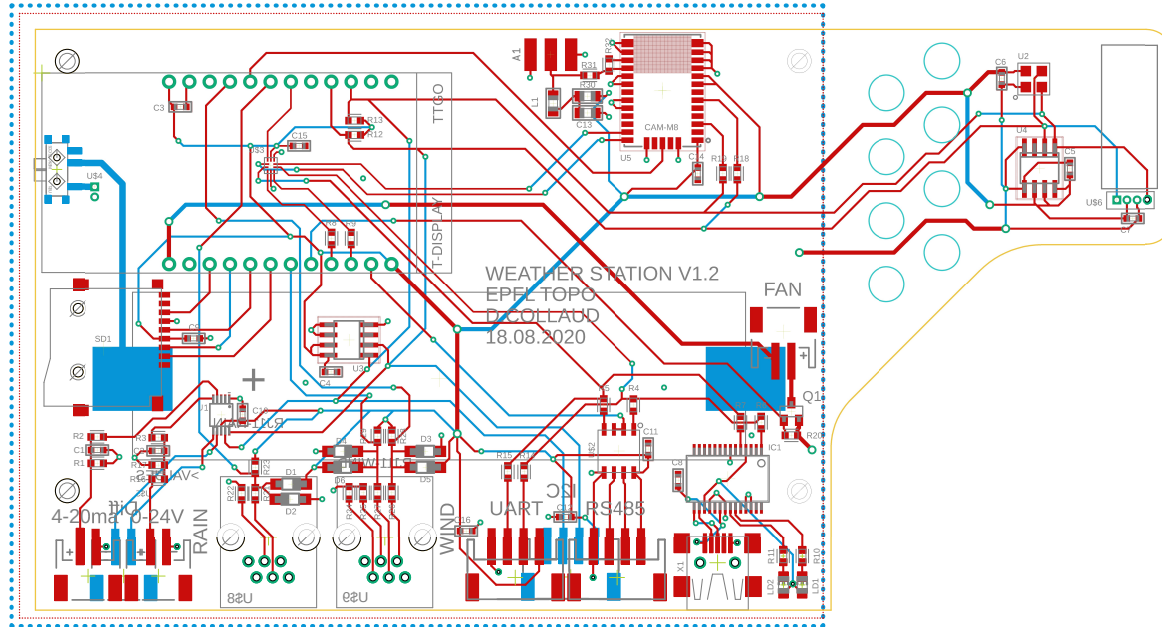


A été changer sur les boards V1
 A3 -> CMD_FAN -> 12
 12 -> WDIR -> A3
 38 -> ESP_TX -> 15
 15 -> RAIN -> 38

38 is read only
 12 doit être à la masse au démarrage sinon ça part en 1.8v
 (Certain ne peuvent pas être utilisé en analogique 32)

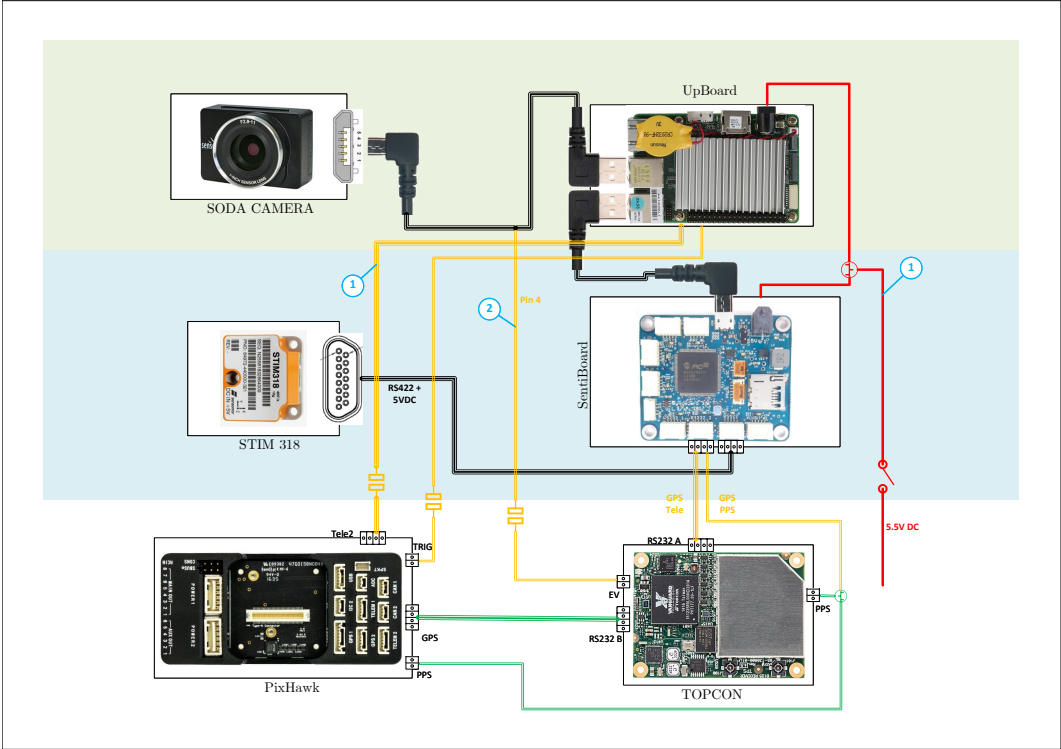
A changer pour version 2
 - Tourner les connecteur RJ11
 - Corriger le footprint du capteur pression
 - GPIO12 doit être à la masse au démarrage
 - Changer la résistance wind direction à 10k

C.5.2 Weather Station Layout



04.10.2020 15:02 f=1.30 F:\Google Drive\Eagle\WeatherStation\ESP32\main.brd

C.6 SODA-STIM Payload Schematic



C.6 SODA-STIM Payload Schematic

D Software

D.1 Software Communication

An overview of the different software and their links are presented hereafter and is depicted in Fig. D.1

QGroundControl

A GUI display, as seen in Fig. D.2(a), eases the operator to send commands to the UAV and monitor in real-time the vehicle position, and flight track with defined waypoints as well as the UAV instruments and subsystems.

The GCS communicates with the UAV via long-range radio signals in the 433 MHz band for telemetry (downlink) and telecommand (uplink) with a set of transceiver from SiK Telemetry Radio (named previously 3DR radio) [119]. The module attached to the UAV is depicted in Fig. D.2(b) on the left side and the module attached to the ground station is on the right side. For maintaining a reliable bi-directional communication, the baud rate between plane and ground radios on this link is chosen to be limited to 38400 bps. The protocol used to embed these messages is MAVLink, an opened and widely used protocol in the small-unmanned-vehicle community. Its major aim is to put packet messages into a data structure of 17 bytes for communicating from the GCS to the UAV and the inter-communication of UAV subsystems. Each message contains a cyclic redundancy check (CRC) to ensure message integrity. Whenever the operator sends commands to the autopilot via the GCS, the messages are carried out using this protocol.

PX4

The autopilot runs several parallel tasks/threads, so called internal applications. The inter-thread/inter-process communication is achieved via an asynchronous publish / subscribe messaging scheme called μ ORB [120]. Communication channels are called topics. Each

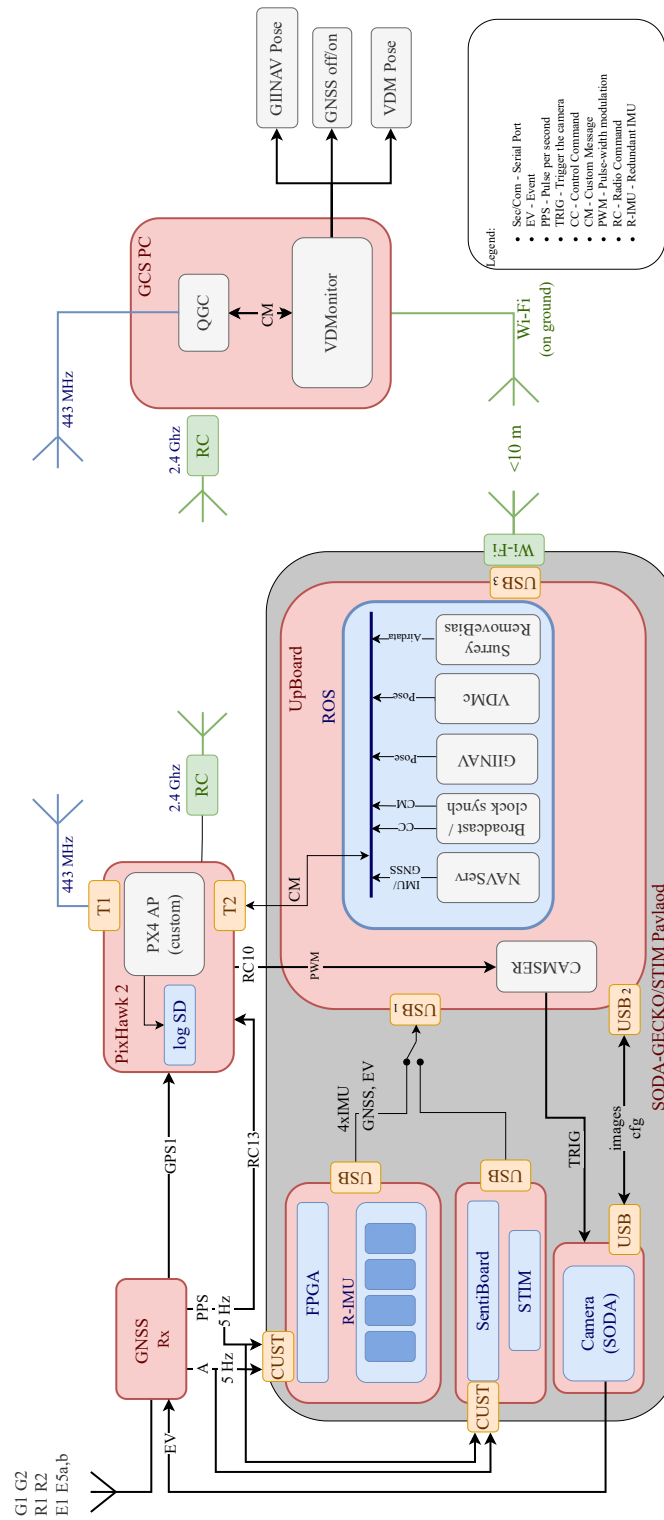


Figure D.1: Information flow between SW/HW components

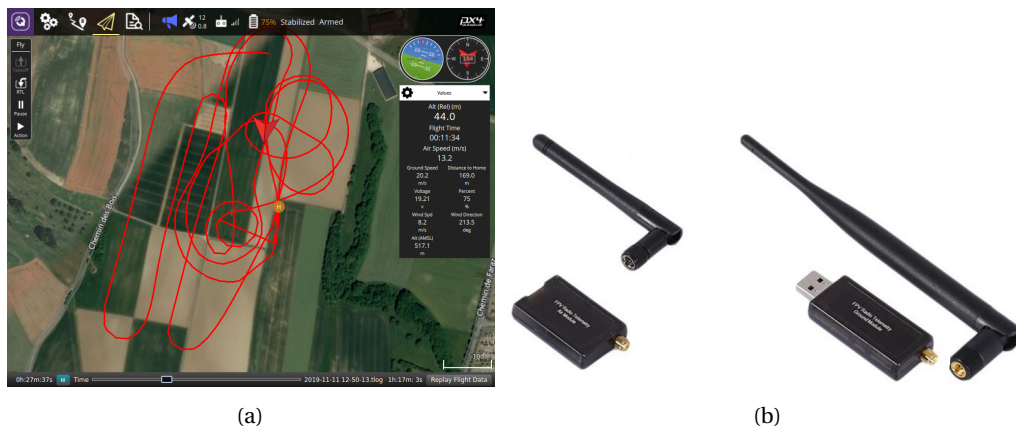


Figure D.2: (a) QGround Control software with a flight trajectory in manual mode and (b) Pair of transceiver module for the UAV (left), and Ground station

topic is associated to a specific *theme*, e.g., the vehicle position, the status of the actuators, etc., and it defines the type of message that can transit on it. Each application willing to provide information on a topic does it by, first, *advertising* the topic and then *publishing* messages on it every time is appropriate according to the application semantics. Applications can also *subscribe* to a topic. In this case, each time a new message is published on the topic, they are asynchronously notified, and the message is delivered to them via a *callback function*. All standard topics in the PX4 firmware are listed in the overview of μ ORB communication network¹. The AP software modification related to sensors and topics is described in Sec. 6.3.2.

One special application exists in PX4 which translates MAVLink messages received at any of the (serial) telemetry port to equivalent μ ORB ones, and vice versa. This allows messages coming to the GCS (such as commands) to be delivered to the appropriate applications in the PX4 firmware. A bridging mechanism also allows GCS MAVLink messages, received via the radio link, to be first converted to μ ORB ones, then converted back to MAVLink, and finally delivered to the computer via the serial link on telemetry port 2. The downlink works in a similar manner.

The onboard connection between the autopilot and the embedded computer is obtained via a high-speed serial link (112.5 kB/s) connecting the serial port Telemetry 2 on the autopilot on the COM 2 port of the embedded computer, allowing MAVLink communication between the devices. The autopilot sends the flight control commands to the UpBoard via the μ ORB-MAVLink mechanism, the transformation between the autopilot system time and the GPS time (Sec. 7.1), and the `debug_vector` ROS topics that are used to transmit messages between the GCS and the UpBoard computer with the autopilot as the bridge. More details about these messages are presented in Sec. 7.2.

The PixHawk2 autopilot board receives data from the GNSS receiver through the serial port

¹https://dev.px4.io/v1.9.0/en/middleware/uorb_graph.html

Appendix D. Software

(details in Sec. C.1.1), logs them together with other internal sensors on the SD card and communicates with the GCS as well as the embedded computer via the telemetry ports 1 and 2, respectively. The messaging protocol on these ports is MAVLink. The same protocol is used to communicate with the GCS (4.6 kB/s) to the GCS via radio link on.

The general protocol architecture is summarized in Fig. D.3

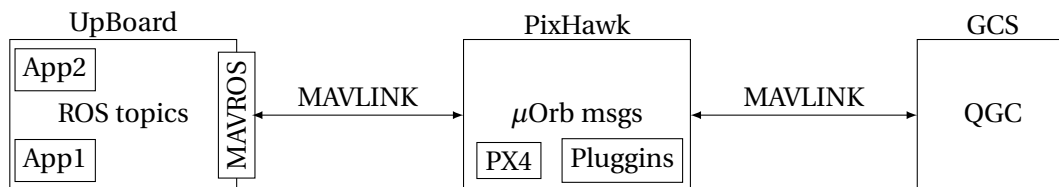


Figure D.3: Messaging protocols between the embedded computer, autopilot and the GCS

D.2 ROS Nodes

D.2.1 Start all Nodes

To launch all the nodes, the ROS package `vdm_field` has to be used. The package possesses the configuration and launch scripts for the different nodes. The directory tree of the package is presented hereafter and the explanations are given in-text using the comment character `#`.

```
.
|-- CMakeLists.txt           # Node compilation cmake
|-- README.md
|-- bags                     # rosbag output saved for post processing and debug
|-- config                  # configuration for giinav, and IMU: stim or NavChip
|   |-- giinav
|   |   |-- STIM318_flight
|   |   |   |-- EKF.ini
|   |   |   |-- GPS.ini
|   |   |   |-- Global.ini
|   |   |   |-- IMU.ini
|   |   --- gecko
|   |       |-- EKF.ini
|   |       |-- GPS.ini
|   |       |-- Global.ini
|   |       --- IMU.ini
|   |-- mavros_config.yaml  # mavros configuration to enable the debug_vector messages
|   --- mavros_pluginlists.yaml # enable plugins for obtaining some PX4 messages
|-- giinav_output          # giinav navigation solution output
|-- launch                  # configuration script to launch the different nodes
|   |-- all.launch
|   |-- ap_clk_synch.launch
|   |-- broadcastnode.launch
|   |-- giinav.launch
|   |-- imu_ds_ma.launch
|   |-- mavros.launch
|   |-- nav_server_ros.launch
|   --- nav_server_ros_static.launch
```

```

| |-- no_nav.launch
| |-- rosbag.launch
| |-- surrey_calibrate.launch
| |-- surrey_sensor.launch
| --- vdm_c.launch
|-- log_raw                # Raw IMU, GNSS and pitot output from NavServer
|-- package.xml            # package ROS configuration
|-- scripts                # scripts to launch/stop ROS nodes and gather data
| |-- configure_screen.sh
| |-- download_data.sh     # download the flight data
| |-- node_data.txt
| |-- start_nodes.sh       # >> START HERE << script used to start all nodes
| --- stop_nodes.sh        # script used to stop all nodes
--- vdmc_output            # Saved VDMc navigation output and config. per run
    |-- VDMc_20220825_082725 # STIM13 flight
        |-- NAV.dat         # VDMc: navigation solution
        |-- config
        | | |-- TP2         # TP2 init files
        | | | |-- initCovariance.ini
        | | | |-- initStates.ini
        | | | |-- models.ini
        | | | |-- outputs.ini
        | | | --- processNoise.ini
        | | |-- filter.ini
        | | --- vdm_c.ini
        --- logs.txt       # VDMc output logs

```

D.2.2 Node Subscribers and Publishers

ap_clk_synch

Node [/ap_clk_synch]

Publications:

- * /airpressure_tagged [sensor_msgs/FluidPressure]
- * /airspeed_tagged [mavros_msgs/VFR_HUD]
- * /cc_tagged [mavros_msgs/RCOut]
- * /rosout [rosgraph_msgs/Log]

Subscriptions:

- * /mavros/debug_value/debug_vector [mavros_msgs/DebugValue]
- * /mavros/imu/static_pressure [sensor_msgs/FluidPressure]
- * /mavros/rc/out [mavros_msgs/RCOut]
- * /mavros/vfr_hud [mavros_msgs/VFR_HUD]

broadcastnode

Node [/broadcastnode]

Publications:

- * /mavros/debug_value/send [mavros_msgs/DebugValue]
- * /rosout [rosgraph_msgs/Log]
- * /timeSync [std_msgs/Float32MultiArray]
- * /toggleOnOffGnss [std_msgs/Bool]

Subscriptions:

- * /GIINAV_POSE [nav_msgs/Odometry]
- * /giinav_status [unknown type]

Appendix D. Software

```
* /mavros/debug_value/debug_vector [mavros_msgs/DebugValue]
* /mavros/debug_value/named_value_float [mavros_msgs/DebugValue]
* /vdmc_pose [nav_msgs/Odometry]
* /vdmc_status [unknown type]
```

GiiNav

Node [/giinav]

Publications:

```
* /GIINAV_POSE [nav_msgs/Odometry]
* /GIINAV_STATUS [std_msgs/String]
* /rosout [roscpp_msgs/Log]
```

Subscriptions:

```
* /GPS0 [std_msgs/UInt8MultiArray]
* /IMU0 [std_msgs/UInt8MultiArray]
```

mavros

Node [/mavros]

Publications:

```
* /mavros/debug_value/debug [mavros_msgs/DebugValue]
* /mavros/debug_value/debug_vector [mavros_msgs/DebugValue]
* /mavros/debug_value/named_value_float [mavros_msgs/DebugValue]
* /mavros/debug_value/named_value_int [mavros_msgs/DebugValue]
* /mavros/rc/out [mavros_msgs/RCOut]
[...]
```

Subscriptions:

```
* /mavlink/to [unknown type]
* /mavros/debug_value/send [mavros_msgs/DebugValue]
[...]
```

nav_server_ros

Node [/nav_server_ros]

Publications:

```
* /GPS0 [std_msgs/UInt8MultiArray]
* /IMU0 [std_msgs/UInt8MultiArray]
* /IMU1 [std_msgs/UInt8MultiArray]
* /PITO [std_msgs/UInt8MultiArray]
* /rosout [roscpp_msgs/Log]
```

Subscriptions:

```
* /toggleOnOffGnss [std_msgs/Bool]
```

Services:

```
* /nav_server_ros/get_loggers
* /nav_server_ros/set_logger_level
```

surrey

Node [/surrey]

Publications:

```

* /airData [surrey_sensor/AirData]
* /filt_airSpeed [std_msgs/Float64]
* /pitotBiasServer/cancel [actionlib_msgs/GoalID]
* /pitotBiasServer/goal [surrey_sensor/MeanActionGoal]
* /rosout [rosgraph_msgs/Log]
* /surrey [surrey_sensor/Surrey]

```

Subscriptions:

```

* /PITO [std_msgs/UInt8MultiArray]
* /pitotBiasServer/feedback [unknown type]
* /pitotBiasServer/result [unknown type]
* /pitotBiasServer/status [unknown type]

```

vdm_c

Node [/vdm_c]

Publications:

```

* /giinavInitialPose_topic [nav_msgs/Odometry]
* /rosout [rosgraph_msgs/Log]
* /vdmc_pose [nav_msgs/Odometry]

```

Subscriptions:

```

* /GIINAV_POSE [nav_msgs/Odometry]
* /GPSO [std_msgs/UInt8MultiArray]
* /cc_tagged [mavros_msgs/RCOut]
* /imu_formatted [sensor_msgs/Imu]
* /removeCoeffs_topic [unknown type]

```

removeSensorBiasServer

Node [/removeSensorBiasServer]

Publications:

```

* /baroBiasServer/feedback [surrey_sensor/MeanActionFeedback]
* /baroBiasServer/result [surrey_sensor/MeanActionResult]
* /baroBiasServer/status [actionlib_msgs/GoalStatusArray]
* /gpsBiasServer/feedback [surrey_sensor/MeanActionFeedback]
* /gpsBiasServer/result [surrey_sensor/MeanActionResult]
* /gpsBiasServer/status [actionlib_msgs/GoalStatusArray]
* /pitotBiasServer/feedback [surrey_sensor/MeanActionFeedback]
* /pitotBiasServer/result [surrey_sensor/MeanActionResult]
* /pitotBiasServer/status [actionlib_msgs/GoalStatusArray]
* /rosout [rosgraph_msgs/Log]
* /tempMeanServer/feedback [surrey_sensor/MeanActionFeedback]
* /tempMeanServer/result [surrey_sensor/MeanActionResult]
* /tempMeanServer/status [actionlib_msgs/GoalStatusArray]

```

Subscriptions:

```

* /GPSO [std_msgs/UInt8MultiArray]
* /baroBiasServer/cancel [actionlib_msgs/GoalID]
* /baroBiasServer/goal [surrey_sensor/MeanActionGoal]
* /gpsBiasServer/cancel [actionlib_msgs/GoalID]
* /gpsBiasServer/goal [surrey_sensor/MeanActionGoal]
* /pitotBiasServer/cancel [actionlib_msgs/GoalID]
* /pitotBiasServer/goal [surrey_sensor/MeanActionGoal]
* /surrey [surrey_sensor/Surrey]
* /tempMeanServer/cancel [actionlib_msgs/GoalID]
* /tempMeanServer/goal [surrey_sensor/MeanActionGoal]

```

D.3 VDMc

D.3.1 VDM C++ class architecture

Version 1

Fig. D.4 presents class architecture of VDMc software version 1. Two main parts can be identified, which are (i) the data handling (upper part) and (ii) the estimation (lower part). The software is still evolving, and some classes might be added, but the main architecture will remain the same.

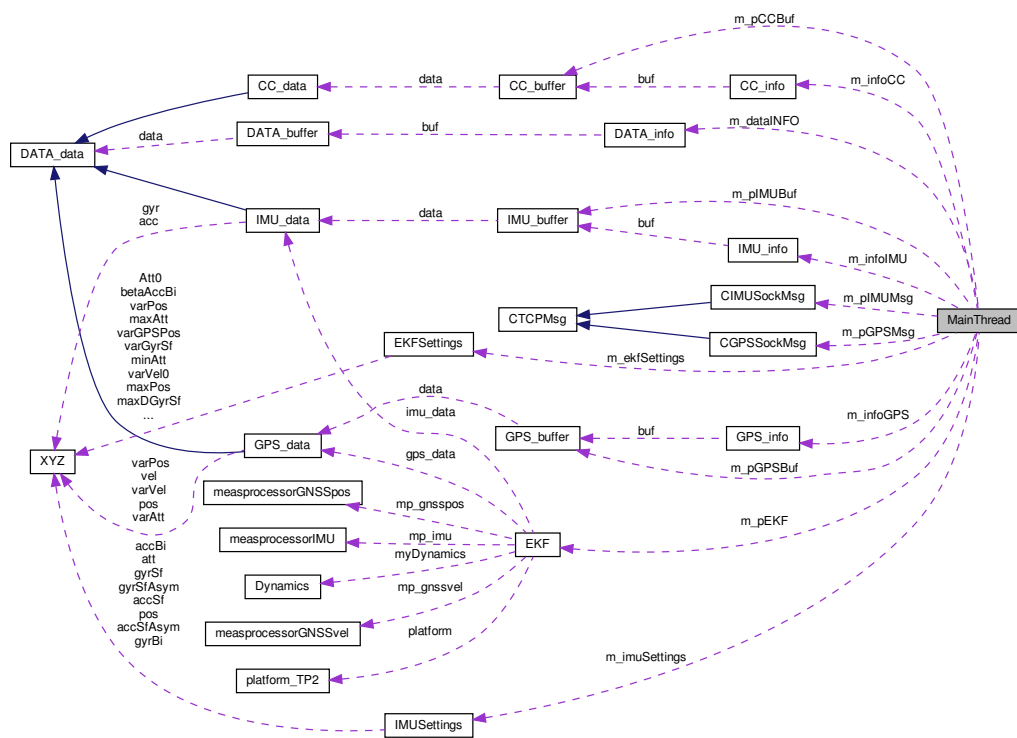


Figure D.4: VDMc C++ version 1 (2020) class architecture

Version 2

The addition of timers, the backpropagation mechanism, and the configuration files, among others, have evolved into a new class architecture presented in Fig. D.5 The code currently

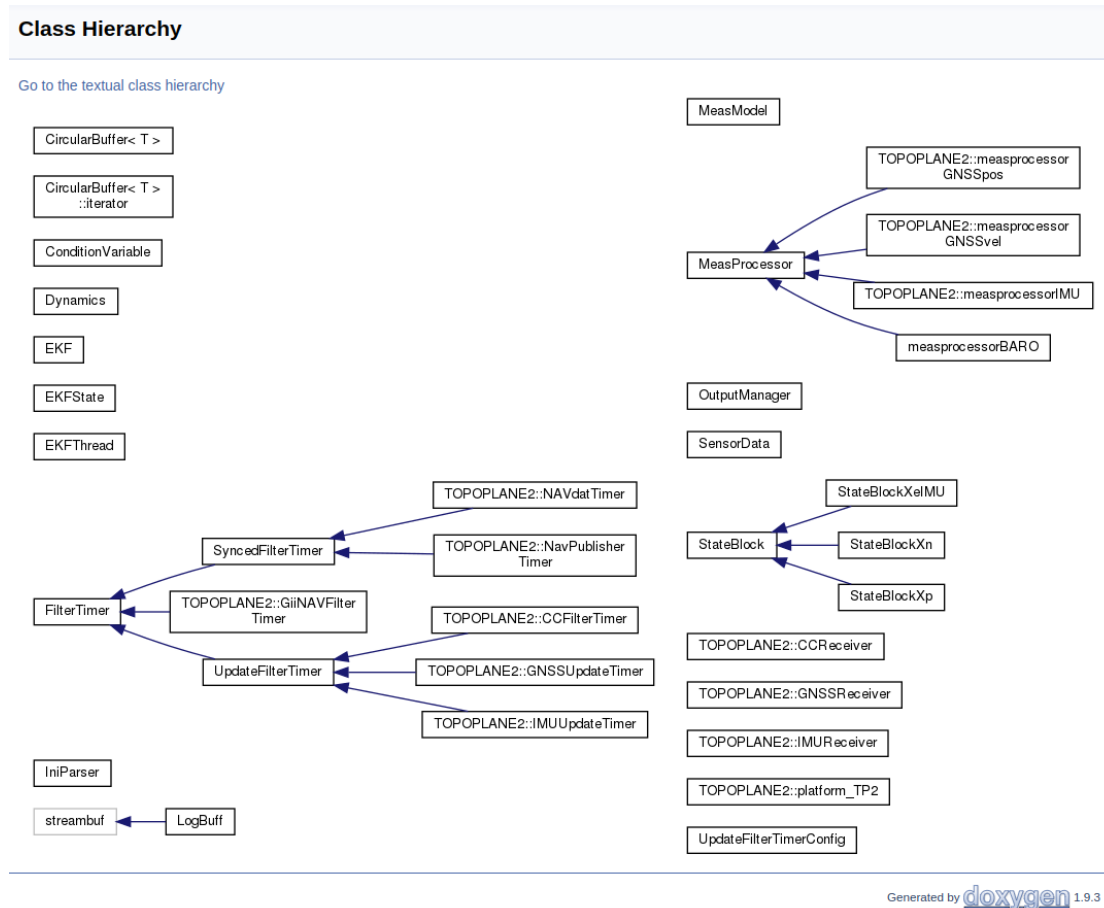


Figure D.5: VDMc version 2 (2022) C++ class architecture

runs independent threads, which greatly simplify the complexity of the classes' dependencies.

D.3.2 VDMc Main Time Loop

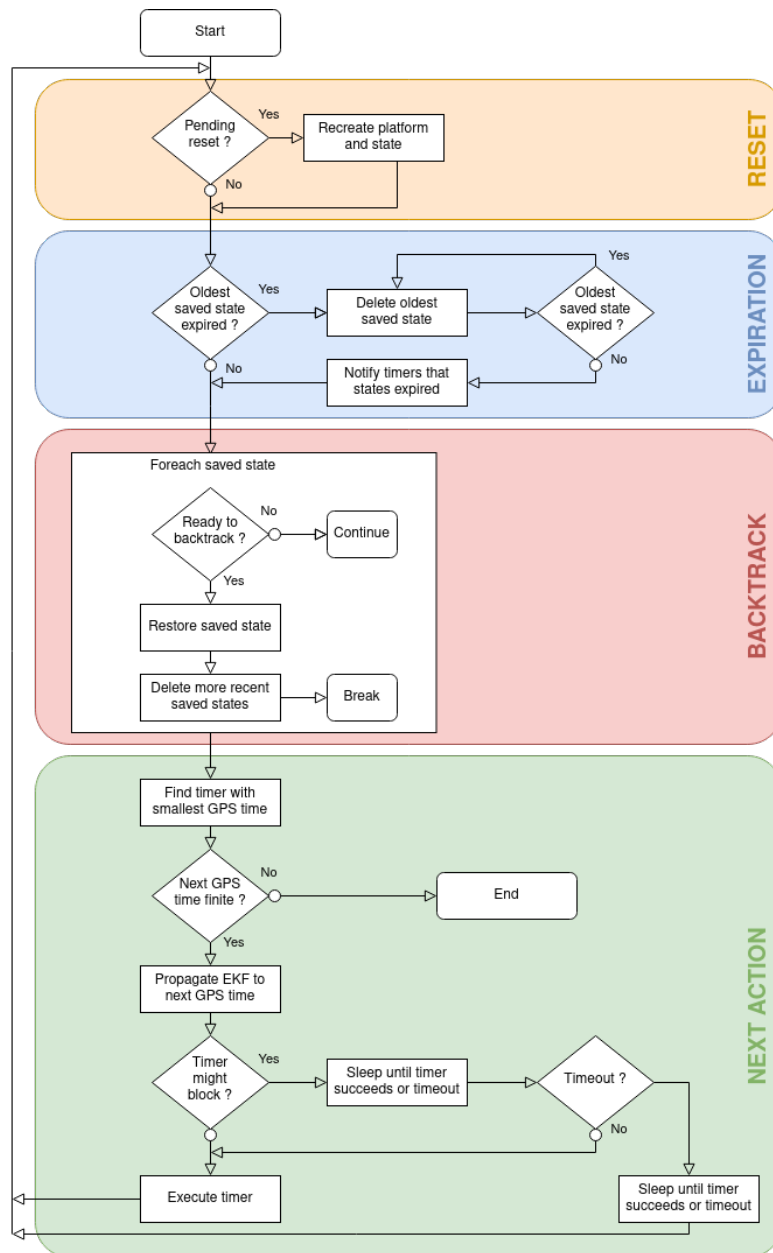


Figure D.6: Main EKF Loop of VDMc

D.3.3 Flight Phases

Pre-flight and Initialization

All nodes presented in Sec. 7.2 are started by script. A dedicated ROS package named `vdm_field` handles the correct launch of the nodes (in separate “screens”) with their cor-

responding configuration. Details of that node are given in Sec. D.2.1.

Static and Dynamic Pressure Calibration

The air data (from the *Surrey* Pitot tube) are calibrated while the platform is static on the ground and the Pitot tube opening is covered to limit the input of dynamic pressure. The *Surrey* responsible for parsing the data sent by the NavServer via the topic PITO, waits for a potential calibration performed with a Server-Client ROS mechanism. The `removeSensorBiasServer` node is called and removes the dynamic pressure bias by using the mean values over 10 seconds. A similar bias removal is applied for barometric altitude with the averaged GPS attitude. The following two listings present the modified values of the *surrey* (Lst. D.1) and *airData* (Lst. D.2) topics before and after calibration.

Listing D.1: *surrey* topic

```

1  time: 376748.242124
2  P0: 96269.03125
3  P1: -4.5485534668
4  P_atm: 96306.53125
5  T_ext: 30.7266731262
6  T_int: 45.0999984741
7  Humidity: 22.1220703125
8  ---
9  time: 376748.252124
10 P0: 96265.7734375
11 P1: 0.0187552012503
12 P_atm: 96306.53125
13 T_ext: 30.7266731262
14 T_int: 45.0999984741
15 Humidity: 22.1220703125

```

Listing D.2: *airData* topic

```

1  time: 376748.242124
2  airSpeed: 2.87045826344
3  baroAltitude: 428.428502203
4  density: 1.10408072118
5  ---
6  time: 376748.252124
7  airSpeed: 0.184321267785
8  baroAltitude: 520.353207142
9  density: 1.10408072118

```

The determination of the barometric altitude is given in Appendix B.1.2.

It should be noted that while airspeed and barometric sensor models are implemented in the real-time applications VDMc, the *Surrey* air data are currently used only to update the air density ρ parameters in the online demonstrator.

Navigation States \mathbf{x}_n

\mathbf{x}_n are provided by the GiiNav application as presented in Sec. 7.4. After receiving the first pose, VDMc waits two minutes before its own alignment to obtain a reliable initialization of attitude. The GiiNav attitude uncertainty is transmitted, but is not currently used. A decision based on uncertainty with a threshold can be a further implementation. The decision to not implement this automatic detection at this stage of development is made based on the idea of redesigning an independent alignment procedure, free of GiiNav software. This would make the VDMc software self-sufficient. The alignment logs are given in Lst. D.3.

Listing D.3: Navigation states initialization after 2 minutes

```

28 Now, wait: 120.0000000000 seconds for alignment.

```

Appendix D. Software

```
29 INFO: Initializing the navigation at time 378228.200000
30 Initializing navigation states:
31     0.8127811067
32     0.1142002117
33 630.1126324753
34 15.3994625111
35 -0.6883438040
36 1.4721671699
37 0.0140216270
38 -0.0820468683
39 -0.0099143834
40 0.0015296677
41 0.0000000000
42 0.0000000000
43 0.0000000000
```

It can be seen from Lst. D.3 that the initial angular velocities are not initialized, as *GiiNav* does not provide these values. In the current implementation, the drone operator should ensure that the drone is leveled at initialization. An addition to the software would be to detect a “stable” attitude, in combination with the uncertainty threshold mentioned above on the attitude to start the initialization. These values are generally close to zero and are rapidly updated with the IMU updates. **Aerodynamic coefficients \mathbf{x}_p**

The initial aerodynamic coefficients come from the WMF methodology (Sec. 5.2) used on the flight *CF_STIM6* with values summarized in Tab. 8.9. They are estimated for 10 minutes after the alignment given by *GiiNav*

IMU bias \mathbf{x}_e

As previously mentioned, the IMU bias and deterministic errors can be removed using the methodology proposed by [103]. However, this methodology is not applied for the payload "STIM-SODA" as the IMU *STIM-318* applies temperature calibrated control of sensor biases. The IMU bias states \mathbf{x}_e are initialized as zero.

Wind \mathbf{x}_w

For this particular flight (*STIM_12*), the wind was low and is initialized to zero. But it is always a good idea to check the weather station to adjust the wind as suggested in Sec. 9.4.1.

ρ

The air density ρ is updated as long as the *surrey_node* is working. As ρ is used as a variable and not as a state, there is no process model for air density in the current implementation. This would be an interesting addition to the software in future versions. If there is no sensor with the capacity to approximate the air density, the local air density can be determined based on observations of portable the weather stations such as the one presented in Sec. 6.3.4 using the formula given in Eq. 6.1.

Actuators \mathbf{x}_a

As soon as the VDMc is aligned, the control command input from the autopilot (and correctly time-tagged thanks to the node `ap_clock_synch`) are used to update the state \mathbf{x}_a following the model given in Sec. 3.5.2.

Listing D.4: Actuator initialization

```
44 INFO: Initializing the command control at time 378228.489828
```

Dynamic State Reduction

10 minutes after alignment and continuous GNSS observations, the state reduction is applied. The recorded logs give:

Listing D.5: Dynamic state reduction log

```
45 Changed state dimension to: 26 at: 378828.2744043290 and it took 7692 us.
```

Note that after state reduction, only 26 remain: the state s added as scaled (Sec. 8.2) called with an abuse of language ρ , is directly updated with the node `Surrey`, and therefore not added in the state space. A comparison of the CPU load of the embedded computer before (47-states) and after (26 states) state space reduction of the two more consuming VDMc threads drops from 16.4% and 11.3% to 6.9% and 3.1%, respectively. A visual comparison of the load is given in Fig. D.7. The results of autonomous navigation for the online demonstrator are given in Sec. 10.3.1.

D.3.4 Configuration

The VDMc application needs a set of configuration parameters, for example, the frequency at which the filter should run, how long the filter should stay in Warm-Up phase after initialization (Sec.4.2.3) or where to write the navigation solution. The KF needs its own set of parameters, as well as the initial values of auxiliary states and the corresponding covariances $(\mathbf{x}(0), \mathbf{P}(0))$, the uncertainty of the model process \mathbf{Q} and the sensor covariance \mathbf{R} . Each platform (*TP1*, *TP2*, *eBeeX* or other) will have its model with its set of sensors and properties. This information is stored in the configuration files, so it can be modified without recompiling the whole program. Configuration files use a syntax similar to the traditional `.ini` format and support textual key-value pairs in different sections. An automatic parser will read the `.ini` files to initiate the filter with the correct settings. If some values are missing from the configuration files, the default values have been hardcoded in the platform-dependent source files. VDMc tells the operator if the configuration files are loaded correctly (Lst. D.6)

Appendix D. Software

```

up@up: ~/ros_ws 105x31
 1 [|||||] 100.0%] Tasks: 53, 55 thr; 3 running
 2 [|||||] 26.8%] Load average: 2.47 2.15 1.39
 3 [|||||] 33.1%] Uptime: 00:35:58
 4 [|||||] 34.8%]
Mem[|||||] 691M/3.27G]
Swp[|||||] 0K/3.27G]

  PID USER   PRI NI  VIRT  RES  SHR S  CPU% MEM%   TIME+  Command
 3194 up    20  0  409M 20336 10728 S  16.4 0.6  0:20.71 /home/up/ros_ws/devel/lib/vdm_c/vdm_c
 3200 up    20  0  409M 20336 10728 S  11.3 0.6  0:15.05 /home/up/ros_ws/devel/lib/vdm_c/vdm_c
 3196 up    20  0  409M 20336 10728 S  1.9 0.6  0:03.63 /home/up/ros_ws/devel/lib/vdm_c/vdm_c
 3179 up    20  0  360M 62588 10852 S  0.6 1.8  0:01.92 /usr/bin/python /opt/ros/melodic/bin/r
 3186 up    20  0  360M 62588 10852 S  0.6 1.8  0:00.29 /usr/bin/python /opt/ros/melodic/bin/r
 3197 up    20  0  409M 20336 10728 S  0.0 0.6  0:00.07 /home/up/ros_ws/devel/lib/vdm_c/vdm_c
 3199 up    20  0  409M 20336 10728 S  0.0 0.6  0:00.05 /home/up/ros_ws/devel/lib/vdm_c/vdm_c
 3187 up    20  0  360M 62588 10852 S  0.0 1.8  0:00.03 /usr/bin/python /opt/ros/melodic/bin/r
 3198 up    20  0  409M 20336 10728 S  0.0 0.6  0:00.00 /home/up/ros_ws/devel/lib/vdm_c/vdm_c

```

(a)

```

up@up: ~/
 1 [|||||] 34.9%] T
 2 [|||||] 36.5%] L
 3 [|||||] 22.9%] U
 4 [|||||] 100.0%]
Mem[|||||] 690M/3.27G]
Swp[|||||] 0K/3.27G]

  PID USER   PRI NI  VIRT  RES  SHR S  CPU% MEM%
 3194 up    20  0  409M 20336 10728 S  6.9 0.6
 3200 up    20  0  409M 20336 10728 S  3.1 0.6
 3196 up    20  0  409M 20336 10728 S  3.1 0.6
 3179 up    20  0  360M 62588 10852 S  0.6 1.8
 3186 up    20  0  360M 62588 10852 S  0.0 1.8
 3199 up    20  0  409M 20336 10728 S  0.0 0.6
 3197 up    20  0  409M 20336 10728 S  0.0 0.6
 3187 up    20  0  360M 62588 10852 S  0.0 1.8
 3198 up    20  0  409M 20336 10728 S  0.0 0.6

```

(b)

Figure D.7: CPU load before (47 states) and after (26 states) the dynamic state reduction

Listing D.6: VDMc initialization with loading of the configuration files

```

1  *=====
2  | VDMNav - connects to ROS topic, then Kalman Filter |
3  | (c) Gabriel Laupre, Simon Gilgien, EPFL, 2020 |
4  | Inspired from ORIGINAL Giinav-Linux-Version |
5  | Version compiled on Jul 26 2022, 13:14:29 |
6  *=====
7
8  INFO (EKF): read and parse the configuration files
9  INFO (EKF): init the filter
10 INFO (TP2): Platform created
11 INFO (EKFThread): Waiting for Giinav initial alignment...

```

or the default values have been loaded (Lst. D.7).

Listing D.7: VDMc loading default initial values (covariance matrix)

```

9  INFO (EKF): read and Parse the configuration files
10 Parse error at line 0 in file config/filter.ini

```

```

11 Missing key 'EKF/dt_min' in file config/filter.ini
12 Using default value "0.001000" instead.
13 Missing key 'EKF/backpropEnabled' in file config/filter.ini
14 Using default value "1" instead.
15 Missing key 'EKF/saveStateDelay' in file config/filter.ini
16 Using default value "0.001000" instead.
17 Missing key 'EKF/filterMethod' in file config/filter.ini
18 Using default value "1" instead.
19 Missing key 'Backpropagation/max_delay' in file config/filter.ini
20 Using default value "0.200000" instead.
21 Missing key 'WarmUp/WU_time' in file config/filter.ini
22 Using default value "50.000000" instead.
23 Missing key 'WarmUp/WU_gamma' in file config/filter.ini
24 Using default value "0.100000" instead.
25 INFO (EKF): init the filter
26 INFO (TP2): Platform created
27 Parse error at line 0 in file config/TP2/models.ini
28 Missing key 'Models/estimate.Xn' in file config/TP2/models.ini
29 [...]

```

The .ini files are written again for bookkeeping and error tracking at the location specified in the vdm_.ini configuration file, along with the logs and the navigation output.

D.3.5 Sensors Classes

A sensor can be defined following a set of rules. The MeasProcessor class is an abstract class representing a sensor configuration for a particular platform type. There are currently several subclasses: IMU, GNSS position, GNSS velocity, barometric, and airspeed measurements for TP2. To implement a new sensor configuration, a new subclass should be created with the definition of the sensor model $h()$ with the method `h_func`, and the linearized sensor model $H()$ with the method `linH_func` (Fig. 7.7). Then the class needs to be linked to the type of platform. In the current implementation, it is the class dedicated to *TP2* as presented in Fig 7.4. Then, a timer dedicated to the sensor has to be created. The timer will handle the update synchronization and, in particular, when it is expected to be performed. This mechanism allows saving the states in case of missing or delayed sensor observations to perform a late update and propagate the updated states. The general methodology of the state backpropagation is explained below.

D.3.6 The FilterTimer Class

Any event that modifies the filter at a certain time must be implemented as a `FilterTimer`. The class is mostly designed for events that repeatedly occur at a fixed frequency, but it is also possible to use it for irregular events or events that occur only once. Examples of regular events include sensor measurement updates and outputs at a fixed frequency. Single-shot events are typically initialization tasks.

The most important parts of the `FilterTimer` class are the `m_nextGPSTime` and `m_nextSystemTime`

fields. The semantic is that an event should occur when the filter time is at `m_nextGPSTime` and that the necessary data is expected to be available at the system time `m_nextSystemTime`. The main loop processes the timers by increasing next GPS time, and for each of them, it propagates the EKF to the next GPS time, wait until the current system time is greater or equal to the next system time of the timer, and call the user-defined `action()` method of the timer.

The `action()` pure virtual method should perform the intended action and optionally update the next GPS and system times. By default, the timer increments the times by the nominal period of the timer, but this can be changed. The return value of the timer indicates whether the state should be saved for later backpropagation. If the timer cannot perform the intended action because it misses some data, but this data might be available later, the timer should indicate that the state should be saved.

The actual backpropagation is triggered by the user-defined `canBacktrack()` method. When a state is saved, this function is called regularly on the saved copy of the timer, and it should check whether the missing data is now available. If this function indicates that it can backtrack, the saved state is restored so that the event can be processed.

D.3.7 The Main Loop

The main loop is implemented (Fig. D.6) in the `EKF::process()` method. It is composed of the following steps:

1. Delete all saved states that are older than the maximum backtracking delay. If one or more states are deleted, the timers are notified via the `expirationHook()` callback, so they can release any shared resources they were holding for the event of a backpropagation.
2. For every saved state, call the `canBacktrack()` method on the timer that requested it to be saved. If that call returns `true`, replace the current state with the saved state and delete all more recent saved states.
3. Find the timer with the smallest next GPS time.
4. Propagate the filter to that GPS time. Propagation intervals under a certain threshold will not be performed, so the GPS time of the filter might not exactly match the time requested by the timer.
5. Call the user-defined `action()` method of the timer.
6. If the timer requests to save the state, the state of the filter, including the timers, is copied and the state is added to the list of saved states along with the copy of the timer that requested the state to be saved.

If backpropagation is not desirable, (e.g. for post-processing recorded data), it can be disabled in the `filter.ini` configuration file. In this case, when a timer requests the state to be saved, the last step is not performed, and it will result in the `action()` method of the filter to be called repeatedly until it increases its `m_nextGPSTime`.

The EKF state is a structure composed of the following fields:

- `X`: The state vector of the filter.
- `X_tot`: The full state vector of the filter, including states that are not evaluated.
- `P`: The covariance matrix (or the U member if the Thornton-Bierman factorization is used)
- `t`: The GPS time of the filter
- `UD_D`: The D diagonal matrix of the Thornton-Bierman factorization of the covariance matrix
- `U`: A vector of matrices storing platform-dependent additional state (e.g. control commands).
- `ekf`: A pointer to the EKF object the state belongs to
- `timers`: A vector of `FilterTimers`

When in the saving state, the timers are copied, since they maintain internal state about the timing. Implementations of timers are expected to keep timers lightweight and use shared memory for most data, such as buffers.

The saved states are stored in a doubly linked list in the EKF itself. The saved objects are of type `std::pair<FilterTimer *, EKFSState *>`, and composed of the saved state and the saved copy of the timer that requested the state to be saved, and therefore is responsible of determining if the state can be restored.

D.3.8 Implementation of Timers

When implementing a timer, the `FilterTimer` class should be subclassed and the following methods should be implemented. Alternatively, one of the specialized classes may be used (see below).

- **Constructor:** The `FilterTimer` constructor needs a nominal frequency and a name. If the timer is for an event occurring only once, such as an initialization, it is possible to pass `+0` as the frequency. This will result in the period being a positive infinity.

- Copy constructor: This should be private, and only used in the `copy()` method. It is used to make copies of the constructor on state save. This should be implemented even if the timer never requires to save states, as it will need to be saved when other timers require to save state. To minimize overhead, it is recommended to use shared memory across copies for most of data.
- `copy()`: this method should allocate and return a copy of the timer. This is a workaround for a virtual copy constructor. Implementation should be something similar to

```
1     return new FooTimer(*this);
```

- `action()`: This is the actual action of the filter. The function receives the current EKF state, and can modify it. The time of the EKF state is guaranteed to be close to the next GPS time of the timer, except if the timer is not yet fully initialized (its next GPS time is still the initial value of -1).

After this function returns, the next GPS and system times are incremented by the nominal period of the filter, except if the function sets the `m_disableTimeUpdate` flag. This applies regardless of whether the times are modified by the `action()` function.

This function returns `true` if the current state should be saved. The state is saved before the time is automatically incremented. To be compatible with the post-processing mode, this function should allow to be called repeatedly when it returns `true`.

Additionally, the following virtual functions might be overwritten:

- Destructor: If you need to release resources. Remember that the overhead should be minimal for best backpropagation performances, since many copies might be deleted at once.
- `initialized()`: This functions returns whether the timer is fully initialized. The default implementation just checks whether the next GPS time is -1 . This function is currently not used on every timer.
- `canBacktrack()`: This function should be implemented if the timer might request to save states. It is passed the saved state, and should returns whether the saved state can be backtracked. This method is called on the saved copy, not on the currently running timer.
- `savedStateHook()`: This function is called when the timer requested to save state, after the timer and state has been copied. The state after the copy is passed as an argument. Both the timer and the state are the original copies that will continue to run. If you override this method, you should call the base class implementation.
- `expirationHook()`: This function is called when one or more saved states have been deleted after their expiration delay, regardless of which timer requested to save those states. It should be used to release any shared resources that were kept in case of a

backpropagation to those states. The parameters are the current state of the EKF and the oldest copy of this timer that is still saved.

If the filter has no more actions to perform, the next GPS time should be set to positive infinity. This will ensure that it will no longer be called. Removing the timer from the `state.timers` vector is not supported.

D.3.9 Specialized Timers

Two subclasses of `FilterTimer` have been written for common tasks. They can be used as base class for actual timer implementations.

UpdateFilterTimerConfig

The `UpdateFilterTimerConfig` is designed to perform actions on external data using a circular buffer for storage. Although the main purpose of this timer are the sensor updates, it is also possible to use it for other events requiring external data (c.f. the `TOPOplane 2 CC` timer for an example).

The type of used data is the `SensorData` structure. It is composed of a GPS time at which the data should apply, a system time of reception of the data used for real-time synchronization, and a vector of data.

The `action()` method is defined to keep the timer synchronized with the external data source, and support backtracking. The algorithm is

1. If the filter is not initialized, initialize the next times with the times of the first data if it is available and return.
2. If no new data is available, request to save the state and return.
3. While new data is available and in the past, discard it. If the data is probably the one for the current time, pass it to the user-defined `updateOldData()` function, synchronize the timer on this data and return.
4. If no more data is available, request the state to be saved and return.
5. If a data is available for the current time, pass it to the user-defined `update()`, synchronize the timer on it and discard it.
6. If the next data is available, synchronize the timer on this data.

Note that if the next data is available, but the data for the current time is not, the state is not saved. This is because data is supposed to arrive in order.

Discarded data is only removed from the buffer if there are no saved states. Otherwise discarded data is kept in the buffer to be reused in case of backpropagation. If saved states expire, the `expirationHook()` method removes data that is not required by any copy of the filter anymore from the buffer.

The `canBacktrack()` function is implemented by searching the buffer for the data at the time at which the state was saved.

To implement a filter using this class, the constructors and `copy()` function must be implemented in the same way as when using the `FilterTimer` directly. Additionally, the method `update` must be implemented. The arguments are the EKF state and the `SensorData` for the current time. The `updateOldData()` method can optionally be overridden to treat data when the filter is not synchronized with the data. The default implementation does nothing. If you override the `expirationHook()` method, you should call `UpdaterFilterTimer`'s implementation in order to free the buffer.

SyncedFilterTimer

The `SyncedFilterTimer` class specializes the `FilterTimer` to perform an action synchronized with another `FilterTimer`, even when it drifts. By performing multiple actions at the same time, less propagation steps are required. It can also be interesting to output the state estimation just after processing a measurement when the uncertainty is minimal. However, if the source timer is not regular, the `SyncedFilterTimer` also won't be regular. So the use of this timer is a trade-off between computational cost and regularity.

The `SyncedFilterTimer` implements the `action()` method to find the source timer in the `state->timers` vector, synchronize the next GPS and system time with it, and call the implementation-defined `syncedAction()` method. It is important that the `SyncedFilterTimer` appears in the `state->timers` vector before its source timer. This ensures that the `SyncedFilterTimer` executes after the source and does not cause a deadlock.

To implement a timer using this class, the `SyncedFilterTimer` class must be sub-classed and the following methods must be defined:

- **Constructor:** The `SyncedFilterTimer` constructor takes a pointer to the source timer and the name of the filter. The frequency is automatically defined to be the same that the frequency of the source.
- `syncedAction()`: This method must be implemented to perform the actual action. It takes a pointer to the current EKF state as a parameter.
- **Copy constructor and `copy()`** must be defined similarly as when sub-classing the `FilterTimer` class directly.

Other optional functions from the `FilterTimer` class can be defined. If the `savedStateHook()` method is overridden, the `SancedFilterTimer` implementation must be called instead of the `FilterTimer` one.

D.3.10 Thorton Bierman Implementation for UDU Factorization

UDU Decomposition

Algorithm 1: UDU Decomposition

```

Data:  $P$ 
Result:  $U, D$ 
 $n \leftarrow$  number of states;
 $U \leftarrow [0]_{n \times n}$ ;
 $D \leftarrow [0]_{1 \times n}$ ;                                     /* Diagonal matrix */
 $ud_{tol} \leftarrow 1 \times 10^{-18}$ ;                       /* Tolerance for positive definiteness */
for  $j \leftarrow n$  to 1 do
  for  $i \leftarrow j$  to 1 do                               // Triangle matrix
     $tmp \leftarrow P(i, j)$ ;
    for  $k \leftarrow (j + 1)$  to  $n$  do
       $tmp \leftarrow tmp - U(i, k) * D(k) * U(j, k)$ 
    end
    if  $i = j$  then
      if  $tmp \leq ud_{tol}$  then                               // Force values to ensure positiveness
         $D(j) \leftarrow 1$ ;
         $U(j, j) \leftarrow 0$ ;
      else
         $D(j) \leftarrow tmp$ ;
         $U(j, j) \leftarrow 1$ ;
      end
    else
       $U(i, j) \leftarrow tmp / D(j)$ 
    end
  end
end
end

```

Thornton Propagation

Algorithm 2: Thornton propagation

Data: $\mathbf{U}_{k-1}, \mathbf{D}_{k-1}, \Phi_k, \delta t, \mathbf{w}_k, \mathbf{G}_k$

Result: $\mathbf{U}_k, \mathbf{D}_k$

$[n, r] \leftarrow$ dimension of \mathbf{G}_k ;

$\Phi_{\mathbf{U}_{k-1}} \leftarrow \Phi_k * \mathbf{U}_{k-1}$;

for $i \leftarrow n$ **to** 1 **do**

$\sigma = 0$;

for $j \leftarrow 1$ **to** n **do**

$\sigma \leftarrow \sigma + \Phi_{\mathbf{U}_{k-1}}(i, j) * \mathbf{D}_{k-1}(j, j)$; /* Recover equivalent noise from $\hat{\mathbf{P}}_{k-1}$ */

if $j < r$ **then**

$\sigma \leftarrow \sigma + \mathbf{G}_k(i, j)^2 * \mathbf{w}_k(j, j) * \delta t$; /* Update new process noise \mathbf{Q}_k */

end

end

$\mathbf{D}_k(i, i) \leftarrow \sigma$;

for $j \leftarrow 1$ **to** $i - 1$ **do**

$\sigma \leftarrow 0$;

for $k \leftarrow 1$ **to** n **do**

$\sigma \leftarrow \sigma + \Phi_{\mathbf{U}_{k-1}}(i, k) * \mathbf{D}_{k-1}(k, k) * \Phi_{\mathbf{U}_{k-1}}(j, k)$;

end

for $k \leftarrow 1$ **to** r **do**

$\sigma \leftarrow \sigma + \mathbf{G}_k(i, k) * \mathbf{w}_k(j, j) * \mathbf{G}_k(j, k) * \delta t$

end

$\mathbf{U}_k(j, i) \leftarrow \frac{\sigma}{\mathbf{D}_k(i, i)}$;

for $k \leftarrow 1$ **to** n **do**

$\Phi_{\mathbf{U}_{k-1}}(j, k) \leftarrow \Phi_{\mathbf{U}_{k-1}}(j, k) - \mathbf{U}_k(j, i) * \Phi_{\mathbf{U}_{k-1}}(i, k)$;

end

for $k \leftarrow 1$ **to** r **do**

$\mathbf{G}_k(j, k) = \mathbf{G}_k(j, k) - \mathbf{U}_k(j, i) * \mathbf{G}_k(i, k)$;

end

end

end

Bierman Update

Algorithm 3: Bierman observation update

Data: $\tilde{\mathbf{U}}_k, \tilde{\mathbf{D}}_k, \mathbf{R}_k, \tilde{\mathbf{x}}_k$
Result: $\hat{\mathbf{U}}_k, \hat{\mathbf{D}}_k, \hat{\mathbf{x}}$
 $n \leftarrow \#$ of states;
 $n^* \leftarrow \#$ of states to be updated;
 $\mathbf{H}_k \leftarrow$ first evaluation $\mathbf{H}_k(\tilde{\mathbf{x}}_k)$; /* Eq. 2.31 */
 $\delta \mathbf{z}_k \leftarrow \mathbf{z}_k - \mathbf{h}(\tilde{\mathbf{x}}_k)$; /* first innovation */
for $i \leftarrow 1$ **to** n^* **do**
 $\mathbf{a} \leftarrow \tilde{\mathbf{U}}_k^T * (\mathbf{H}_k(i, :))^T$; /* i^{th} row of \mathbf{H}_k^T */
 $\mathbf{b} \leftarrow \tilde{\mathbf{D}}_k * \mathbf{a}$; /* $|\mathbf{b}| = n$ */
 $\alpha \leftarrow \mathbf{R}(i, i) \gamma \leftarrow \frac{1}{\alpha}$;
 for $j \leftarrow 1$ **to** n **do**
 $\beta \leftarrow \alpha, \alpha \leftarrow \alpha + \mathbf{a}(j) * \mathbf{b}(j)$;
 $\lambda \leftarrow -\mathbf{a}(j) * \gamma, \gamma \leftarrow \frac{1}{\alpha}$; /* intermediate $P(i, j)$ */
 $\hat{\mathbf{D}}_k(j, j) \leftarrow \beta * \gamma * \tilde{\mathbf{D}}_k(j, j)$;
 for $k \leftarrow 1$ **to** $j - 1$ **do**
 $\beta \leftarrow \hat{\mathbf{U}}_k(k, j)$;
 $\hat{\mathbf{U}}_k(k, j) \leftarrow \beta + \mathbf{b}(k) * \lambda$;
 $\mathbf{b}(k) \leftarrow \mathbf{b}(k) + \mathbf{b}(j) * \beta$;
 end
 end
 $\delta \hat{\mathbf{x}}_k \leftarrow \gamma * \delta z_k(i) * \mathbf{b}$;
 $\hat{\mathbf{x}}_k \leftarrow \tilde{\mathbf{x}}_k + \delta \hat{\mathbf{x}}_k$;
 $\delta z_k \leftarrow \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_k)$;
 $\mathbf{H}_k \leftarrow$ new evaluation of $\mathbf{H}_k(\hat{\mathbf{x}}_k)$;
end

Bibliography

- [1] Pamela Cohn, Alastair Green, Meredith Langstaff, and Melanie Roller. *Commercial drones are here: The future of unmanned aerial systems*. <https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems>. Accessed: 2022-02-22.
- [2] Masiri Kaamin, Siti Nooraini Mohd Razali, Nor Farah Atiqah Ahmad, Saifullizan Mohd Bukari, Norhayati Ngadiman, Aslila Abd Kadir, and Nor Baizura Hamid. "The application of micro UAV in construction project". In: *AIP Conference Proceedings*. Vol. 1891. 1. AIP Publishing LLC. 2017, p. 020070.
- [3] Tarek Rakha and Alice Gorodetsky. "Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones". In: *Automation in Construction* 93 (2018), pp. 252–264.
- [4] Vikram Puri, Anand Nayyar, and Linesh Raja. "Agriculture drones: A modern breakthrough in precision agriculture". In: *Journal of Statistics and Management Systems* 20.4 (2017), pp. 507–518.
- [5] Emmanuel CleDAT, Laurent Valentin Jospin, Davide Antonio Cucci, and Jan Skaloud. "Mapping quality prediction for RTK/PPK-equipped micro-drones operating in complex natural environment". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (2020), pp. 24–38.
- [6] Parvathi Sanjana and M Prathilothamai. "Drone design for first aid kit delivery in emergency situation". In: *2020 6th international conference on advanced computing and communication systems (ICACCS)*. IEEE. 2020, pp. 215–220.
- [7] Nikolaos Tsiamis, Loukia Efthymiou, and Konstantinos P Tsagarakis. "A comparative analysis of the legislation evolution for drone use in OECD countries". In: *Drones* 3.4 (2019), p. 75.
- [8] Mehran Khaghani and Jan Skaloud. "Autonomous Navigation of Small UAVs based on Vehicle Dynamic Model". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XL-3/W4. Lausanne, Switzerland, 2016, pp. 117–122. DOI: 10.5194/isprs-archives-XL-3-W4-117-2016. URL: <http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-3-W4/117/2016/> (visited on 05/17/2016).

Bibliography

- [9] Mehran Khaghani and Jan Skaloud. "Application Of Vehicle Dynamic Modeling In Uavs For Precise Determination Of Exterior Orientation". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 41 (2016), p. 827.
- [10] Mehran Khaghani and Jan Skaloud. *Autonomous and Non-Autonomous Dynamic Model Based Navigation System for Unmanned Vehicles*. US Patent 15176283. 2016. URL: <https://patents.google.com/patent/US20160364990>.
- [11] Mehran Khaghani. "Vehicle Dynamic Model Based Navigation for Small UAVs". English. PhD thesis. Lausanne, Switzerland: EPFL, 2018.
- [12] Robert Fitzgerald. "Divergence of the Kalman filter". In: *IEEE Transactions on Automatic Control* 16.6 (1971), pp. 736–747.
- [13] Hery A Mwenegoha, Terry Moore, James Pinchin, and Mark Jabbal. "A model-based tightly coupled architecture for low-cost unmanned aerial vehicles for real-time applications". In: *IEEE Access* (2020).
- [14] Gabriel Laupré and Jan Skaloud. "On the self-calibration of aerodynamic coefficients in vehicle dynamic model-based navigation". In: *Drones, MDPI* 4.3 (2020), p. 32.
- [15] The autoware Foundation. *Robot Operating System*. <https://www.ros.org/>. Accessed: 2022-07-16. 2022.
- [16] Yannick Stebler. "Modeling and Processing Approaches for Integrated Inertial Navigation". English. PhD thesis. Lausanne, Switzerland: EPFL, 2013.
- [17] Skaloud Jan. *Sensor Orientation*. Laboratoire Topométrie: École Polytechnique F;d;rale Lausanne, 2013.
- [18] Jay A. Farrell. *Aided Navigation: GPS with High Rate Sensors*. New York, New York, USA: McGraw-Hill, 2008.
- [19] Alexander C Aitken. "IV.—On least squares and linear combination of observations". In: *Proceedings of the Royal Society of Edinburgh* 55 (1936), pp. 42–48.
- [20] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*. Vol. 4. Prentice hall Upper Saddle River, NJ, 2002.
- [21] Roland Longchamp. *Commande numérique de systèmes dynamiques: cours d'automatique*. Vol. 1. PPUR Presses polytechniques, 2010.
- [22] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. "Unsupervised scale-consistent depth and ego-motion learning from monocular video". In: *Advances in neural information processing systems* 32 (2019).
- [23] Venkatesh Madyastha, Vishal Ravindra, Srinath Mallikarjunan, and Anup Goyal. "Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation". In: *AIAA Guidance, Navigation, and Control Conference*. 2011, p. 6615.
- [24] Eun-Hwan Shin. "Estimation techniques for low-cost inertial navigation". In: *UCGE report* 20219 (2005).

-
- [25] James S Meditch. *Stochastic optimal linear estimation and control*. McGraw-Hill, 1969.
- [26] Gerald J Bierman. “A new computationally efficient fixed-interval, discrete-time smoother”. In: *Automatica* 19.5 (1983), pp. 503–511.
- [27] Herbert E Rauch, F Tung, and Charlotte T Striebel. “Maximum likelihood estimates of linear dynamic systems”. In: *ALAA journal* 3.8 (1965), pp. 1445–1450.
- [28] Gabriel Laupré and Jan Skaloud. “Calibration of Fixed-Wing UAV Aerodynamic Coefficients with Photogrammetry for VDM-based Navigation”. In: *Proceedings of the 2021 International Technical Meeting of The Institute of Navigation, ION*. 2021, pp. 775–786.
- [29] Jan Skaloud and Klaus Legat. “Theory and reality of direct georeferencing in national coordinates”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.2 (2008), pp. 272–282.
- [30] Richard Pio. “Euler angle transformations”. In: *IEEE Transactions on automatic control* 11.4 (1966), pp. 707–715.
- [31] Howard Curtis. *Orbital mechanics for engineering students*. Butterworth-Heinemann, 2013.
- [32] Bernhard Hofmann-Wellenhof, Klaus Legat, and Manfred Wieser. *Navigation*. Springer Science & Business Media, 2003.
- [33] David Hoag. “Apollo guidance and navigation: Considerations of apollo imu gimbal lock”. In: *Canbridge: MIT Instrumentation Laboratory* (1963), pp. 1–64.
- [34] Bartel Leednert Van Der Waerden. “Hamilton’s discovery of quaternions”. In: *Mathematics Magazine* 49.5 (1976), pp. 227–234.
- [35] William Rowan Hamilton. *Elements of quaternions*. London: Longmans, Green, & Company, 1866.
- [36] John Baez. “The octonions”. In: *Bulletin of the american mathematical society* 39.2 (2002), pp. 145–205.
- [37] Joan Sola. “Quaternion kinematics for the error-state Kalman filter”. In: *arXiv preprint arXiv:1711.02508* (2017).
- [38] Ljudmila Meister. “Quaternions and their application in photogrammetry and navigation”. In: *Habilitation der TU Freiberg* (1998).
- [39] William Rowan Hamilton. “Xi. on quaternions; or on a new system of imaginaries in algebra”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 33.219 (1848), pp. 58–60.
- [40] David M Henderson. *Euler angles, quaternions, and transformation matrices for space shuttle analysis*. Tech. rep. 1977.
- [41] NK Ibragimov. *Elementary Lie group analysis and ordinary differential equations*. Vol. 197. Wiley New York, 1999.
- [42] P Vaniček and E Krakiwsky. “Geodesy: the Concepts North Holland”. In: *Amsterdam* (1986).

Bibliography

- [43] John B Schleppe. *Development of a real-time attitude system using a quaternion parameterization and non-dedicated GPS receivers*. University of Calgary Alberta, 1996.
- [44] Mohinder Grewal and Angus Andrews. *Kalman filtering: theory and practice. Incl. 1 disk*. Prentice Hall, Jan. 1, 1993. 381 pp. ISBN: 0-13-211335-X.
- [45] Albert Einstein. “The general theory of relativity”. In: *The Meaning of Relativity*. Springer, 1922, pp. 54–75.
- [46] James R Clynych. “Radius of the Earth-Radii used in geodesy”. In: *Naval Postgraduate School* (2002).
- [47] Mehran Khaghani and Jan Skaloud. “Assessment of VDM-based Autonomous Navigation of a UAV under Operational Conditions”. In: *Robotics and Autonomous Systems* 106 (2018), pp. 152–164.
- [48] Yannick Stebler. “Modeling and processing approaches for integrated inertial navigation”. In: (2013).
- [49] NIMA WGS84 Update Committee. Department of Defense. *World Geodetic System 1984, Its Definition and Relationships with Local Geodetic Systems*. Technical report. Springfield, Virginia, USA, 3rd edition: National Imagery and Mapping Agency, 2000.
- [50] P Kenneth Seidelmann. *Explanatory supplement to the astronomical almanac*. University Science Books, 2006.
- [51] Guillaume JJ Ducard. *Fault-tolerant flight control and guidance systems: Practical methods for small unmanned aerial vehicles*. Springer Science & Business Media, 2009.
- [52] Tor A Johansen, Andrea Cristofaro, Kim Sørensen, Jakob M Hansen, and Thor I Fossen. “On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors”. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2015, pp. 510–519.
- [53] Philipp Clausen. *Calibration Aspects of INS Navigation*. Tech. rep. EPFL, 2019.
- [54] M. Rehak and J. Skaloud. “Applicability Of New Approaches Of Sensor Orientation To Micro Aerial Vehicles”. In: vol. 3. *International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences* 3. Gottingen: Copernicus Gesellschaft Mbh, 2016, pp. 7. 441–447. DOI: 10.5194/isprsannals-III-3-441-2016. URL: <http://infoscience.epfl.ch/record/225789>.
- [55] Standard Atmosphere. “ISO 2533: 1975”. In: *International Organization for Standardization* (1975), pp. 11–12.
- [56] MathWorks. *Symbolic Math Toolbox- Perform symbolic math computations*. <https://www.mathworks.com/products/symbolic/>. Accessed: 2022-08-16. 2022.
- [57] Mohinder S Grewal and Angus P Andrews. “Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]”. In: *IEEE Control Systems Magazine* 30.3 (2010), pp. 69–78.

- [58] Arthur Gelb, ed. *Applied optimal estimation*. Cambridge, Massachusetts, London: M.I.T. Press, 1974. ISBN: 0-262-20027-9.
- [59] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. 2nd. Hoboken, New Jersey, USA: John Wiley & Sons, 2007.
- [60] Mohinder S Grewal and Angus P Andrews. “Kalman Filtering Theory and Practise”. In: *Printice Hall, Englewood Cliffs, New Jersey* (1993).
- [61] MS Grewal and AP Andrews. “Application of Variants of the Kalman Filter to Various Programs”. In: *Proceedings of the 58th Annual Meeting of The Institute of Navigation and CIGTF 21st Guidance Test Symposium (2002)*. 2002, pp. 336–339.
- [62] Michel Verhaegen and Paul Van Dooren. “Numerical aspects of different Kalman filter implementations”. In: *IEEE Transactions on Automatic Control* 31.10 (1986), pp. 907–917.
- [63] M. Khaghani and J. Skaloud. “Autonomous vehicle dynamic model based navigation for small UAVs”. In: *NAVIGATION, Journal of The Institute of Navigation* 63.3 (Sept. 2016), pp. 345–358.
- [64] J Russell Carpenter and Christopher N D’Souza. “Navigation filter best practices”. In: (2018), p. 149.
- [65] C. Thornton and G. Bierman. “Filtering and error analysis via the UDU^T covariance factorization”. In: *IEEE Transactions on Automatic Control* 23.5 (1978), pp. 901–907. DOI: 10.1109/TAC.1978.1101863.
- [66] Stanley F Schmidt. “Application of state-space methods to navigation problems”. In: *Advances in control systems*. Vol. 3. Elsevier, 1966, pp. 293–340.
- [67] Gerald J Bierman and Catherine L Thornton. “Numerical comparison of Kalman filter algorithms: Orbit determination case study”. In: *Automatica* 13.1 (1977), pp. 23–35.
- [68] Kevin M Brink. “Partial-update schmidt–kalman filter”. In: *Journal of Guidance, Control, and Dynamics* 40.9 (2017), pp. 2214–2228.
- [69] Michael Green and John B Moore. “Persistence of excitation in linear systems”. In: *Systems & control letters* 7.5 (1986), pp. 351–360.
- [70] Hery Mwenegoha, Terry Moore, James Pinchin, and Mark Jabbal. “Model-based autonomous navigation with moment of inertia estimation for unmanned aerial vehicles”. In: *Sensors* 19.11 (2019), p. 2467.
- [71] Mehran Khaghani and Jan Skaloud. “Assessment of VDM-based autonomous navigation of a UAV under operational conditions”. In: *Robotics and Autonomous Systems* 106 (2018), pp. 152–164.
- [72] Rudolph E Kalman and Richard S Bucy. “New results in linear filtering and prediction theory”. In: *Journal of Basic Engineering* (1961).

Bibliography

- [73] Mehran Khaghani and Jan Skaloud. "VDM-based UAV Attitude Determination in Absence of IMU Data". In: *European Navigation Conference (ENC 2018)*. IEEE, Aug. 2018, pp. 84–90.
- [74] Martin Rehak and Jan Skaloud. "Fixed-wing Micro Aerial Vehicle for Accurate Corridor Mapping". English. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. II-1/W1. Toronto, Canada, 2015, pp. 23–31. DOI: 10.5194/isprsannals-II-1-W1-23-2015. URL: <http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-1-W1/23/2015/> (visited on 12/12/2016).
- [75] D. Cucci and Skaloud J. "Joint adjustment of raw inertial data and image observations: methods and benefits". In: *Photogrammetric Week*. 2019, p. 8.
- [76] Gabriel François Laupré, Pasquale Longobardi, Jan Skaloud, and Jean-Christophe Charlaix. "Model Based Navigation of Delta-Wing UAV-In-Flight Calibration and Autonomous Performance". In: *European Journal of Navigation* 21.ARTICLE (2021), pp. 22–30.
- [77] Multiplex Modellsport GmbH & Co.KG. *Multiplex Elapor*. <https://www.multiplex-rc.de/>. Accessed: 2022-08-31. 2022.
- [78] Analog Device. *Precision, Miniature MEMs IMU (2000dps, 8g)*. <https://www.analog.com/en/products/adis16475.html>. Accessed: 2022-08-15. 2022.
- [79] Olivier Martin, Christophe Meynard, Marc Pierrot Deseilligny, Jean-Philippe Souchon, and Christian Thom. "Réalisation d'une caméra photogrammétrique ultralégère et de haute résolution". In: *Proceedings of the colloque drones et moyens légers aéroportés d'observation, Montpellier, France*. 2014, pp. 24–26.
- [80] T. Kluter. *GECKO4NAV Technical Reference Manual*. 1.0. HuCE-microLab, Bern University of Applied Sciences. 2013.
- [81] Stéphane Guerrier, Roberto Molinari, and Jan Skaloud. "Automatic identification and calibration of stochastic parameters in inertial sensors". In: *NAVIGATION, Journal of the Institute of Navigation* 62.4 (2015), pp. 265–272.
- [82] AAeon Europe B.V. *UP Board Series*. <https://up-board.org/up/specifications/>. Accessed: 2022-07-15. 2022.
- [83] Didier Negretto. "R-IMU System Development". In: (2021). URL: <http://infoscience.epfl.ch/record/296057>.
- [84] S. M. Albrektsen and T. A. Johansen. "User-configurable timing and navigation for UAVs". In: *Sensors* 18.8 (2018), p. 2468. DOI: <https://doi.org/10.3390/s18082468>.
- [85] Lorenz Meier. *Pixhawk Autopilot*. 2016. URL: <https://pixhawk.org/modules/pixhawk>.
- [86] Computer Vision and Geometry Lab of ETH Zurich. *PX4 autopilot*. 2009. URL: (<http://http://px4.io/>).
- [87] Jonas Vautherin and Lorenz Meier. *Micro Air Vehicle Communication Protocol*. <https://mavlink.io/en/>. Accessed: 2022-07-15. 2022.

- [88] Philipp Clausen, Jan Skaloud, Pierre-Yves Gilliéron, Bertrand Merminod, Harris Perakis, Vassilis Gikas, and Ioanna Spyropoulou. “Position accuracy with redundant MEMS IMU for road applications”. In: *European Journal of Navigation* 13.2 (2015), pp. 4–12.
- [89] P. Clausen, J. Skaloud, S. Orso, and S. Guerrier. “Construction of dynamically-dependent stochastic error models”. In: *IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE ION PLANS, 2018. DOI: <https://doi.org/10.1109/PLANS.2018.8373524>.
- [90] M. Daakir, M. Pierrot-Deseilligny, P. Bosser, F. Pichard, C. Thom, Y. Rabot, and O. Martin. “Lightweight UAV with on-board photogrammetry and single-frequency GPS positioning for metrology applications”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 127 (2017), pp. 115–126.
- [91] Inc. Dronecode Project. *QGround Control*. <http://qgroundcontrol.com/>. Accessed: 2022-07-12. 2019.
- [92] Richard Shelquist. “An introduction to air density and density altitude calculations”. In: *Internet Survey, Visited on 25th of March* (2012).
- [93] Dylan Collaud and Gabriel Laupré. *TOPO Weather station*. <https://gitlab.epfl.ch/topo/weatherstation/>. Accessed: 2022-07-15. 2022.
- [94] Jan Skaloud, Davide Antonio Cucci, and Kenneth Joseph Paul. “Fixed-wing micro UAV open data with digicam and raw INS/GNSS”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences V-1* (2021), pp. 105–111. DOI: 10.5194/isprs-annals-V-1-2021-105-2021.
- [95] Gabriel Laupré, Mehran Khaghani, and Jan Skaloud. “Sensitivity to Time Delays in VDM-Based Navigation”. In: *Drones, MDPI* 3.1 (2019), p. 11.
- [96] ArduPilot Dev Team. *ArduPilot Autopilot*. 2016. URL: (<http://ardupilot.org/>).
- [97] Maxtena. *Rugged L1/L2 GPS GLONASS Active Antenna*. <http://www.maxtena.com/products/helicore/m1227hct-a2-sma/?v=1ee0bf89c5d1>. Accessed 2022-08-21. (Visited on 10/31/2016).
- [98] Vladimir Ermakov. *MAVLink extendable communication node for ROS with proxy for Ground Control Station*. <http://wiki.ros.org/mavros>. Accessed: 2022-11-09. 2022.
- [99] Phillip Tomé. “Integration of Inertial and Satellite Navigation Systems for Aircraft Attitude Determination”. PhD thesis. Oporto, Portugal: University of Oporto, 2002.
- [100] Gabriel Laupré. *VDMc*. https://gitlab.epfl.ch/laupre/vdm_c. Accessed: 2022-07-16. 2020.
- [101] WOLFRAM. *MATHEMATICA - The world’s definitive system for modern technical computing*. <https://www.wolfram.com/mathematica/>. Accessed: 2022-07-16. 2022.
- [102] Tux Family. *Eigen*. <https://eigen.tuxfamily.org/>. Accessed: 2022-07-19. 2020.
- [103] Philipp Clausen. “Calibration Aspects of INS Navigation”. PhD thesis. Lausanne: IIE, 2019, p. 237. DOI: 10.5075/epfl-thesis-9357. URL: <http://infoscience.epfl.ch/record/264793>.

Bibliography

- [104] Stéphane Guerrier, Jan Skaloud, Yannick Stebler, and Maria-Pia Victoria-Feser. “Wavelet-Variance-based Estimation for Composite Stochastic Processes”. In: *Journal of the American Statistical Association* 108.503 (2013), pp. 1021–1030.
- [105] Emmanuel Cledat, Davide A. Cucci, and Jan Skaloud. “Camera calibration models and methods in corridor mapping with UAVs”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences V-1-2020* (2020), pp. 231–238. DOI: <https://doi.org/10.5194/isprs-annals-V-1-2020-231-2020>.
- [106] FAA DoT. *Operation and Certification of Small Unmanned Aircraft Systems*. Tech. rep. FAA-2015-0150: Notice, 2015.
- [107] Lucas Pirlet. “Kalman Filter Adaptation for Stable Autonomous Drone Navigation”. In: (2021). URL: <http://infoscience.epfl.ch/record/295193>.
- [108] Leonhard Euler. *Institutiones calculi integralis*. Vol. 1. impensis Academiae imperialis scientiarum, 1792.
- [109] Carl Runge. “Über die numerische Auflösung von Differentialgleichungen”. In: *Mathematische Annalen* 46.2 (1895), pp. 167–178.
- [110] Michael S Andrie and John L Crassidis. “Geometric integration of quaternions”. In: *Journal of Guidance, Control, and Dynamics* 36.6 (2013), pp. 1762–1767.
- [111] M. Rehak and J. Skaloud. “Fixed-wing micro aerial vehicle for accurate corridor mapping”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences II-1/W4* (2015), pp. 23–31. DOI: 10.5194/isprsannals-II-1-W1-23-2015. URL: <http://dx.doi.org/10.5194/isprsannals-II-1-W1-23-2015>.
- [112] US Standard Atmosphere. *US standard atmosphere*. National Oceanic and Atmospheric Administration, 1976.
- [113] Wilfried Brutsaert. *Evaporation into the atmosphere: theory, history and applications*. Vol. 1. Springer Science & Business Media, 2013.
- [114] Hashim A Hashim. “Special orthogonal group SO (3), euler angles, angle-axis, rodriguez vector and unit-quaternion: Overview, mapping and challenges”. In: *arXiv preprint arXiv:1909.06669* (2019).
- [115] Federal Aviation Administration. *Airplane Flying Handbook (FAA-H-8083-3A)*. Skyhorse Publishing Inc., 2011.
- [116] <https://aero-naut.de/>. *MODEL CONSTRUCTION MADE IN GERMANY*. Accessed: 2022-07-05. URL: <https://aero-naut.de/produkt/power-prop-lufts-7x-7/>.
- [117] <https://www.albatrosmodelbouw.be/fr/>. Accessed: 2022-07-05. URL: <https://www.albatrosmodelbouw.be/fr/folding-prop/6759-hobbyzone-folding-prop-and-spinner-conscendo-s.html>.
- [118] WB Garner. “Model airplane propellers”. In: *Air-Propeller research document, wbgarner08@verizon.net* (2009).

- [119] Ardupilot Project. *SiK Telemetry Radio*. <https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html>. Accessed: 2022-07-15. 2022.
- [120] PX4 autopilot. *uorb*. <https://dev.px4.io/v1.9.0/en/middleware/uorb.html>. Accessed: 2022-07-15. 2022.

Acronyms

SO(3) Special Orthogonal Group. 18, 23, 176

AGL Attitude Above Ground Level. 85

AP Autopilot. 81, 93, 94, 135, 159, 199, 200

AR Auto-Regressive. 41

BA Bundle Adjustment. 67–69

BLUE Best Linear Unbiased Estimator. 10

CAS Calibrated Airspeed. 193

CC Control Commands. 39, 82, 94, 119

CFD Computational Fluid Dynamic. 2, 3, 55, 57, 59, 206

CPU Central Processing Unit. 144, 145, 223

DCM Direct Cosine Matrix. 18–20

ECEF Earth Center Earth Fixed. 27

ECI Earth Center Inertial. 27

EKF Extended Kalman Filter. 3, 9, 13, 14, 36, 38, 47, 58, 93, 96, 99, 100, 144, 154

ENU East-North-Up. 28, 69, 113

EO External Orientation. 67

ESEKF Error State Extended Kalman Filter. 13, 14, 25, 43, 50, 175

FFT Fast Fourier Transform. 125

GCP Ground Control Point. 68, 130

GCS Ground Control Station. xix, 73, 78, 79, 84, 93, 96, 108, 143, 151, 200, 211, 213, 214

GLONASS GLObalnaïa Navigatsionnaïa Spoutnikovaïa Sistéma. 76

GM Gauss Markov. 41

GMWM Generalized Method of Wavelet Moments. 114

GNSS Global Navigation Satellite System. xvi–xviii, xxi–xxiii, 1, 2, 4, 5, 17, 25, 38, 41, 42, 47, 49, 52–54, 58, 59, 61, 62, 66–69, 73, 74, 76, 78, 80–82, 86, 87, 90, 94, 96–104, 106–108, 112, 114, 116, 120, 121, 130–132, 134, 137, 138, 141, 143, 145, 146, 153–165, 169–172, 174, 177–186, 194, 199, 200, 202, 205, 213, 223, 225

GPS Global Positioning System. 76, 80, 81, 94, 96, 135, 170

Acronyms

- GSD** Ground Sampling Distance. 130
- GUI** Graphical User Interface. 84, 96, 143, 211
- IGN** Institut National de l'Information Géographique et Forestière. 83
- iid** independently and identically distributed. 12
- IIR** Infinite Impulse Response. 125
- IMU** Inertial Measurement Unit. xv, xvii, xxiii, 2, 4, 17, 29, 32, 34, 36–38, 40, 41, 47, 53, 57, 59, 66–70, 73, 75–78, 81, 82, 87, 89, 93, 94, 96, 98, 100, 101, 103, 104, 106, 111, 114, 116, 120, 125–127, 134, 137, 144, 146, 154–156, 158, 159, 161, 165, 169–171, 174, 180, 181, 183, 184, 200, 205, 222
- INS** Inertial Navigation System. xvii, xviii, 1, 34, 38, 58, 59, 61, 62, 66–69, 75, 94, 96, 97, 106, 120, 121, 130, 138, 143, 146, 153, 154, 156, 158, 160–162, 164, 177–181, 184
- ISA** International Standard Atmosphere. 193
- KF** Kalman Filter. 13, 14, 24, 48, 50, 51, 61, 62, 144, 145, 174, 175, 223
- LS** Least Squares. 9, 11, 58, 59
- MAVLink** Micro Air Vehicle communication protocol LINK. 81, 84, 95, 96, 143, 211, 213, 214
- MAVROS** MAVLink extendable communication node for ROS. 81, 95, 96
- METAR** Meteorological Aerodrome Report. 40
- NED** North-East-Down. 28–30, 69
- ODE** Ordinary Differential Equation. 23, 34, 175, 197
- PE** Persistence of Excitement. 54, 64
- PP** Post-Processing. 3, 184
- PPK** Post-Processed Kinematic. xxii, 2, 41, 42, 49, 68, 132, 137, 158, 159, 181, 202
- PPP** Precise Point Positioning. 41
- PPS** Pulse Per Second. 76, 80–82, 94, 135, 199, 205
- PSKF** Partial-update Schmidt Kalman Filter. 60, 62, 63, 145
- PWM** Pulse-Width Modulation. 39, 134, 171, 200
- QGC** QGroundControl. xvi, xvii, 84, 96, 98, 142, 143, 161, 214
- RC** Radio Control. xv, 84
- RK4** Runga Kutta 4. 175
- RLS** Recursive Least Squares. 10–13, 62, 63, 112, 128
- ROS** Robot Operating System. 4, 81, 90, 93–99, 135, 142, 171, 200, 213, 214, 220, 221
- RPM** Rotation Per Minute. 138, 171
- RTK** Real time Kinematic. 2, 41, 49, 68, 87
- RTL** Return to Land. 107
- SPP** Single Point Positioning. xxii, 41, 42, 158, 202

TOV Time Of Validity. 82

UAV Unmanned Aerial Vehicle. xvii, 1–3, 5, 17, 34–36, 38–40, 42, 47, 54, 58, 60, 63, 68, 69, 73, 75, 77, 78, 80, 83–85, 88, 93, 97, 107, 108, 111, 124, 135, 136, 138–140, 142, 143, 151, 153, 154, 161, 163, 182, 203, 211

UTC Coordinated Universal Time. 94

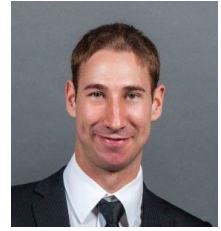
VDM Vehicle Dynamic Model. xvi–xviii, xxi, xxii, xxviii, 2–6, 16, 17, 29, 34, 36–40, 42, 43, 47–51, 53–55, 57–59, 61, 66, 70, 75–78, 82, 83, 85–87, 89, 90, 92–94, 96, 97, 99, 100, 103, 108, 111, 114–121, 125, 129–131, 134–143, 145, 147, 148, 150, 151, 153–156, 158, 160–165, 169–172, 177–182, 186, 203

WGS84 World Geodetic System 84. 28, 33, 41, 49

WMF Wind Moments Forces. 66

Gabriel François LAUPRÉ

Rue Neuve 6C, 1020 Renens, CH | gabriel@laupre.org | linkedin.com/in/glaupre | +41 78 858 0446



TECHNICAL SKILLS

Telecommunication: Digital and Analog Signal Processing, Signal Tracking & Acquisition
Navigation: Estimation (Kalman Filter), Sensor Integration, Global Navigation Satellite System
UAV / Robotic: Robotic Operating System, PX4 autopilot, QGroundControl
Virtualization: OpenStack - Neutron, VxLAN, NVGRE, open vSwitch, SR-IOV
Programming / Package: C, C++, Java, Matlab, Simulink, GNUradio, GMWM

EXPERIENCES

Specialist Officer, Swiss Armed Forces, Bern, Switzerland, *Joint Operations Command*

Since June 2019

Militia Specialist Officer at the Space Cell.

Satellite Communication Engineer, Astrocast SA, Lausanne, Switzerland, *Development Team*

June 2016 – Mars 2017

Worked on the future design of a nanosatellites network providing machine-to-machine services to global businesses.

- Designed a baseband software with Gnuradio for “X-Term MdP” on the Field programmable RF chip LMS6002
- Worked on “Astrocast”, part of the European Space Agency Advanced Research in Telecom. Systems (ARTES)

Firmware Consultant, Chelsio Communications, Sunnyvale CA, USA, *Software/Firmware Department*

August 2015 – July 2016

Coded tools on the embedded multiport switch of Chelsio’s Network Interface Cards for the TCP offloading engine.

- Deployed OpenStack using different tenant isolation methods (VxLAN, NVGRE, VLAN, flat network) to identify throughput, CPU, and memory usage bottlenecks that could be improved with Chelsio’s hardware

Research Assistant, Swiss Federal Institute of Technology (EPFL), Lausanne, *Signal Processing Laboratory 2*

January 2014 - April 2014

European Union Funded SceneNet Project - 3D object modelization based on images.

- Use epipolar algorithms from pictures taken with multi-view stereo cameras for 3D reconstruction using OpenCV
- Develop a C++ library to calculate descriptors and epipolar lines given a set of images of the same object

University Trainee, Universidad del Valle, Cali, Colombia, *Departamento de Quimica*

July - September 2013

Designed and assembled a PCB to control a bioreactor using an Arduino.

- Program the Arduino to log data from the sensors and control the automatic bioreactor’s functions
- Configure the Ethernet module to provide remote access to the bioreactor logs and controls

Intern, Televic Conference, Izegem, Belgium, *R&D Department*

July - September 2012

Created a RF scan module for the “Confidia” wireless conference system.

- Provide a module to scan and have a real-time sense of the 2.4 and 5Ghz channels’ occupancy
- Design a prototype that securely changes channels as soon as heavy wireless traffic is detected

EXTRACURRICULAR ACTIVITIES

Firefighter	2016 - 2020	Volunteering in local and University fire department
Private Pilot (N° 59006)	2017	Private Pilot License started while in the USA, 2015
Head of IEEE SB EPFL	2017 - 2021	Organizing technical conferences and workshops
Radio Amateur (N° 501512)	2019	Call sign HB9HCN, keeps me in the satellite communication world
Project Management	2021	Certificate from PRINCE2 Foundation

EDUCATION AND ACADEMIC PROJECTS

Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

Ph.D. Candidate, Teaching Assistant, Swiss Federal Institute of Technology (EPFL), Lausanne, Geodetic Engineering Lab.
March 2017 – Expected graduation on November 2022

Pursuing research in autonomous aerial navigation in the Robotics, Control, and Intelligent Systems doctoral school.

- Investigating different methods for aerodynamic coefficients self-calibration via state-space augmentation
- Designing the first real-time prototype autonomous VDM-based navigation
- Principal teaching assistant for the course: *Sensor Orientation and Advanced Satellite Positioning*
- Teaching Assistant Award, School of Computer and Communication Sciences, 2017

Bachelor and Master in Communication Systems with Minor in Space Technologies, EPFL

September 2009 – September 2015

Master Thesis, Chelsio Communications, Sunnyvale, CA, Software/Firmware Department

Define the scope of limitations of Chelsio's NIC virtualization support.

- Modify the OpenStack code and identify hardware vendor-specific requirements to be compatible with Neutron
- Give improvement directions for hardware/firmware for future chips and point out driver features that need to be implemented to be compatible with the OpenStack project and, more generally, for virtualization (SR-IOV)

Minor (2014), MS Semester (2013) and BS Project (2012), EPFL, Swiss Space Center

Communication Software Design for the satellite CubETH.

- Program the microcontroller to configure a high-performance narrowband ISM transceiver
- Design and implement the state machine of the communication board for Tx and Rx

Benchmark error-correcting methods for an S-band transmitter.

- Create signals with error-correcting mechanisms (Conv., RS, Interleaving). Send it over a wireless channel, corrupt the data, demodulate the signals using an EXA Signal Analyzer and analyze the error correcting capabilities

Simulation of a demodulator (PM/BPSK) for space application on Simulink and testing the robustness of the design.

- Implement two Costas loops and test the robustness of typical Doppler effects and carrier tracking dynamics

Centre Professionnel du Littoral Neuchâtelois (CPLN), Neuchâtel, Switzerland

Federal Certificate of Competence (CFC) in Computer Science, with honors

August 2005 – July 2008

LANGUAGES

French	●●●●●●	Mother tongue
English	●●●●○	Fluent, daily used at work, working two years for a US company
German	●●●○○	level B2 with Language Certification of Goethe Institute, five months in Germany, 2007
Spanish	●○○○○	Basic knowledge

MAIN PUBLICATIONS

- *Sensitivity to Time Delays in VDM-Based Navigation*, G Laupré, M Khaghani, J Skaloud, *Drones*, 2019, 3(1), 11
- *On the Self-Calibration of Aerodynamic Coefficients in Vehicle Dynamic Model-Based Navigation*, G Laupré, J Skaloud, *Drones*, 2020, 4(3), 32
- *Calibration of Fixed-Wing UAV Aerodynamic Coefficients with Photogrammetry for VDM-based Navigation*, G Laupré, J Skaloud, Proceedings of the 2021 International Technical Meeting of the ION. 2021. p. 775-786
- *Model-Based Navigation of Delta-Wing UAV-In-Flight Calibration and Autonomous Performance*. G Laupré, at al., *European Journal of Navigation*, 2021, vol. 21, p. 22-30.
- *Numerical Strategies for Model-Based Navigation with Fixed-Wing Drone*, G Laupré, at al., submitted for publication, 2022
- *Identifying Aerodynamics of Small Fixed-Wing Drones using Inertial Measurements for Model-Based Navigation*, A Sharma at al., submitted for publication, 2022