

Handling GNSS-Outliers in Multisensor State Estimation

Behandlung von GNSS-Messausreißern in der
Multisensor-Zustandsschätzung

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen
Technischen Hochschule Aachen zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Haoming Zhang

Berichter/in:

Univ.-Prof. Dr.-Ing. Heike Vallery
Prof. Timothy Barfoot, PhD
Assoc. Prof. Dr. Manon Kok

Tag der mündlichen Prüfung:

27.10.2025

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

To my wife, Shihan.

Acknowledgements

My research career in Aachen, Germany has been a long yet colorful journey, filled with many memorable stories that include not only success and gratitude, but also frustration and unexpected surprises. As this journey now approaches its end, I find myself more content and at ease. I no longer regard the doctorate as something “marvelous” or “to be proud of” in itself. This does not mean that it is not valuable, nor that I did not enjoy it or find opportunities to reflect on my self-worth along the way. Rather, I am satisfied and grateful to have discovered that I truly fit into a research career, and that this journey has allowed me to grow both professionally and personally.

During this journey, I would like to sincerely thank my supervisor, *Prof. Dr.-Ing. Heike Vallery*, for her excellent professional guidance and continuous support. I also thank the former chair, *Prof. Dr.-Ing. Dirk Abel*, for his support during the early stage of my doctoral studies. I also cannot express my gratitude enough to *Prof. Timothy D. Barfoot, PhD*, who has always been a leading figure and source of inspiration in my research. His generous mentorship, encouraging words, and our many conversations while visiting research venues have become some of the most unforgettable memories of my life. Finally, it was a great honor to have *Prof. Dr. Manon Kok* as my co-supervisor for this doctoral thesis, from whom I received numerous constructive comments and valuable feedback during the final stage of this journey.

I must also express my sincere thanks to *Dr.-Ing. René Zweigel*, who welcomed me into his research group. His remarkable leadership and trust gave me extensive freedom to grow and develop as a researcher. I also owe countless thanks to my former colleagues, *Dr.-Ing. Jan Gehrt*, *Dr.-Ing. Martin Kosch*, *Dr.-Ing. Shuchen Liu*, *Marvin Huang*, *Rinat Prezdneyakov*, and *Chih-Chun Chen*, for their friendship, support, and many inspiring discussions throughout this journey. I was very fortunate to have had the opportunity to collaborate with an excellent group of students, *Xiao Xia*, *Felix Widmayer*, *Lars Lünemann*, *Zhanxin Wang*, *Xu Chen*, *Shiping Ma*, and *Jiandong Chen*, whose enthusiasm and dedication made our collaborations especially rewarding.

My research journey has rewarded me with lifelong friendships, in particular with my close friends *Dr.rer.nat. Sebastian Schönitz* and *Dr.-Ing. Philipp Ennen*, with whom I shared both close collaboration and many unforgettable personal moments. This reward has further crossed continents and reached Hong Kong, where some of my most cherished friendships, with *Prof. Dr. Li-Ta Hsu*, *Prof. Dr. Wensong Wen*, *Dr. Xiwei Bai*, and *Dr. Feng Huang*, were formed.

Finally, my deepest gratitude goes to my wife, *Shihan*, whose unwavering support made my research career possible.

Abstract

Vehicle localization using classic state estimation approaches generally conducts *least-square* techniques to infer unknown state variables by fusing nonlinear system dynamics and sensor models. In this setting, model uncertainties are typically assumed to follow a *zero-mean Gaussian* distribution. An optimal state estimate can be achieved by minimizing covariances using redundant state representations (aka models), provided the predefined noise parameters match the actual uncertainties.

However, the assumption of Gaussian noise has been shown to be overly conservative and unrealistic in real-world environments, where sensor failures and signal interference frequently result in measurement outliers. In this case, measurement outliers typically exhibit time-varying noise dynamics associated with heavy-tailed, skewed, and multimodal distributions that cannot be adequately modeled in advance. Consequently, a state estimator employing the least-square technique may degrade or even completely diverge once an inconsistent measurement model with overly confident hand-crafted noise parameters is used. Therefore, developing a vehicle localization approach that achieves robust long-term operation in diverse environments remains a challenging problem.

This dissertation examines methods for handling outliers in a graph-optimization-based state estimator, specifically designed for vehicle localization using Global Navigation Satellite Systems (GNSS). In addition to developing novel state estimation techniques, this thesis places noticeable emphasis on two key areas: (a) a theoretical analysis of the state-of-the-art robust outlier rejection approaches, and (b) learning-based methods, supported by extensive ablation studies.

In this regard, I explore two methods for handling GNSS outliers. The first considers the classic M-estimation techniques, which are commonly used to reduce the impact of outliers in state estimation and have proven effective in practice. However, their application to ranging sensors, particularly in terms of robustness and efficiency across different loss kernels, has received little attention. The second method leverages learning-based techniques to predict outlier measurements. Beyond traditional machine learning models, such as Support Vector Machines, I introduce a novel deep network architecture that performs spatiotemporal modeling, specifically considering the characteristics of GNSS noise propagation.

In summary, this dissertation advances the understanding of outlier handling using M-estimation and learning-based methods to enhance the robustness of vehicle localization for long-term operation in challenging environments. By emphasizing these approaches, the thesis aims to provide critical insights and inspire future advancements in robust state estimation for vehicle localization. Additionally, I prioritize reproducibility and comparability by open-sourcing the code for all methods discussed.

Kurzfassung

Fahrzeuglokalisierung unter Verwendung klassischer Ansätze nutzt in der Regel *least-square* Techniken, um unbekannte Zustandsvariablen durch die Fusion von nichtlinearen Systemdynamiken und Sensormodellen abzuleiten. In diesem Rahmen wird angenommen, dass Modellunsicherheiten einer *Gaußschen* Verteilung folgen. Ein optimaler Zustandsschätzwert lässt sich erreichen, vorausgesetzt, dass die vordefinierten Rauschparameter den tatsächlichen Unsicherheiten entsprechen.

Die Annahme einer Gaußschen Rauschverteilung hat sich jedoch in realen Umgebungen unrealistisch erwiesen, wo Sensorausfälle und Signalstörungen häufig zu Messausreißern führen. Die Messausreißer weisen zeitlich variierende Rauschdynamiken auf, die mit “long-tail”, asymmetrischen und multimodalen Verteilungen verbunden sind. Folglich kann ein Zustandsschätzer degradieren oder sogar vollständig divergieren, wenn ein inkonsistentes Messmodell mit übermäßig selbstsicheren, handgefertigten Rauschparametern verwendet wird. Daher bleibt es eine herausfordernde Aufgabe, einen Fahrzeuglokalisierungsansatz zu entwickeln, der einen robusten Langzeitbetrieb in verschiedenen Umgebungen gewährleistet.

Diese Dissertation untersucht Methoden zum Umgang mit Ausreißern in einem graphenoptimierungsbasierten Zustandsschätzer, der speziell für die Fahrzeuglokalisierung mit Globalen Satellitennavigationssystemen (GNSS) entwickelt wurde. Neben der Entwicklung neuartiger Zustandsabschätzungsmethoden liegt diese Arbeit besonderen Wert auf zwei zentrale Bereiche: (a) eine theoretische Analyse moderner Ansätze zur Ausreißererkennung und (b) lernbasierte Methoden, unterstützt durch umfangreiche Ablationsstudien.

Insbesondere werden zwei Methoden zur Behandlung von Ausreißern in unterschiedlichen Anwendungen der Fahrzeuglokalisierung untersucht. Die erste Methode umfasst klassische M-estimation, die häufig verwendet wird, um den Einfluss von Ausreißern in der Zustandsschätzung zu reduzieren und sich in der Praxis als effektiv erwiesen hat. Die zweite Methode setzt auf lernbasierte Techniken zur Prognose von Ausreißermessungen. Neben klassischen maschinellen Lernmodellen stelle ich eine neuartige Deep-Learning-Netzwerkarchitektur vor, die die raum-zeitliche Modellierung unter Berücksichtigung der GNSS-Rauschfortpflanzungseigenschaften durchführt.

Zusammenfassend trägt diese Dissertation zum Verständnis des Umgangs mit Ausreißern bei und verbessert so die robuste Fahrzeuglokalisierung für den Langzeitbetrieb in herausfordernden Umgebungen. Durch die Betonung dieser Ansätze ziele ich darauf ab, wichtige Einblicke zu geben und zukünftige Fortschritte in der robusten Zustandsschätzung für die Fahrzeuglokalisierung voranzutreiben.

Contents

Acknowledgements	iii
Abstract	v
Kurzfassung	vii
Author's Publications	xvi
1 Introduction	1
1.1 Motivation	1
1.2 A Glance through Robust State Estimation	6
1.2.1 Outlier Exclusion	6
1.2.2 Robust Error Modeling	7
1.2.3 Multi-Hypothesis Methods	9
1.2.4 Solver Certification	11
1.2.5 Learning-based Methods	11
1.3 Delimitation	12
1.4 Research Objectives	13
1.4.1 Problem Statement	14
1.4.2 Research Questions and Hypotheses	15
1.5 Contributions	16
1.6 Thesis Outline	16
2 Test Vehicles, Applications, and Datasets	19
2.1 Motivation	19
2.2 Test Vehicles	20
2.3 Applications	22
2.3.1 Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments	22
2.3.2 Application II: GNSS/Multisensor Fusion in Urbanized Areas	23
2.4 Datasets	25
2.4.1 Dataset I: Inland Waterways	25
2.4.2 Dataset II: Urban Areas	25
2.4.3 Dataset III: Machine Learning	26
3 Continuous-Time Factor Graph Optimization	31
3.1 Motivation and Objective	32

3.2	Related Work	33
3.2.1	Graph Optimization for GNSS-based Vehicle Localization	33
3.2.2	Graph Optimization for Multisensor Fusion	34
3.2.3	Continuous-Time Trajectory Representation	35
3.2.4	Modular Estimation Framework	36
3.3	GP-based Continuous-Time Trajectory Representation	36
3.3.1	The Gaussian Process Model	37
3.3.2	The Gaussian Process Motion Prior in $SE(3)$	38
3.3.3	GP-WNOA Motion Prior	39
3.3.4	GP-WNOJ Motion Prior	41
3.4	Time-Centric Factor Graph Optimization	43
3.5	Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments	47
3.5.1	UKF for Tightly Coupled GNSS/INS Fusion	47
3.5.2	FGO for Tightly Coupled GNSS/INS Fusion	49
3.5.3	System Overview	49
3.5.4	Essential Experimental Results	50
3.5.5	Discussion on Application I	56
3.6	Application II: GNSS/Multisensor Fusion for Robust Vehicle Localization .	57
3.6.1	Loosely Coupled Factor Graph Optimization	57
3.6.2	Tightly Coupled Factor Graph Optimization	58
3.6.3	System Overview	59
3.6.4	Essential Experimental Results	60
3.6.5	Discussion on Application II	68
4	Handling Outliers using Classic M-Estimators	73
4.1	Motivation and Objective	73
4.2	Related Work	74
4.2.1	M-Estimation for GNSS-based Localization	74
4.2.2	Self-Tuning M-Estimators	75
4.3	M-Estimation	76
4.3.1	Robustness and Relative Efficiency	77
4.3.2	Kernel Functions	78
4.3.3	Influence Analysis	79
4.3.4	Breakdown Point	82
4.4	Kernel Evaluation	82
4.4.1	Experiment Setup	82
4.4.2	Essential Experimental Results	83
5	Handling Outliers using Learning-based Methods	93
5.1	Motivation and Objective	93
5.2	Related Work	94

5.3	Dataset Generation and Feature Selection	95
5.3.1	Dataset Generation	95
5.3.2	Feature Selection	97
5.4	Problem Formulation	101
5.5	Classic Machine Learning Models	101
5.6	Deep Learning Models	102
5.6.1	Reference Model	102
5.6.2	Proposed Model	103
5.6.3	Loss Function of the Proposed Model	105
5.7	Experiment Design and Results	106
5.7.1	Performance Metrics of NLOS Classification	106
5.7.2	Performance of Pseudorange Error Prediction	110
5.7.3	Vehicle Localization with GNSS Fault Exclusion	114
6	Conclusion	119
6.1	Thesis Summary	119
6.2	Contribution and Results	120
6.2.1	C1: Generalized Factor Graph Optimization	120
6.2.2	C2: Handling Outliers using M-Estimators	121
6.2.3	C3: Handling Outliers using Learning-based Methods	123
6.3	Limitations	124
6.4	Open Questions and Future Work	125
A	Theoretical Foundation	129
A.1	Motivation	129
A.2	Primer on State Estimation	130
A.2.1	State Representation and Observation	131
A.2.2	Probabilistic Theory and Inference	132
A.2.3	Least-Square Estimation	140
A.2.4	Kalman Filter Family	147
A.2.5	Filtering versus Smoothing	156
A.2.6	Inference using Graphical Models	158
A.2.7	Factor Graph Optimization	161
A.2.8	Incremental Smoothing and Mapping	165
A.2.9	Kalman Filter versus Factor Graph Optimization	168
A.3	Primer on Vehicle Localization	171
A.3.1	Frames and Frame Transformation	171
A.3.2	State Space for Vehicle Localization	173
A.3.3	Lie Group and Lie Algebra	174
A.3.4	Related Sensors and Observations	177
A.3.5	Sensor Fusion	185
A.3.6	Simultaneous Localization and Mapping	186

A.3.7	Performance Metrics of Vehicle Localization	188
A.4	Primer on Machine Learning	190
A.4.1	Learning from Data	190
A.4.2	Classic Machine Learning	191
A.4.3	Deep Learning	194
A.4.4	Performance Metrics of Learning-based Methods	200
B	Useful Mathematical Expressions	203
B.1	Practical Transformations	203
B.1.1	Transformation from <i>ECEF</i> to <i>WGS84</i> in Closed Form	203
B.1.2	Transformation from <i>ECEF</i> to <i>WGS84</i> Iteratively	204
B.2	Supplementary Calculations for the GP Motion Priors	207
B.2.1	Derivative of the GP-WNOA Motion Prior	207
B.2.2	Derivative of the GP-WNOA Interpolation	208
B.2.3	Derivative of the GP-WNOJ Motion Prior	210
B.2.4	Derivative of the GP-WNOJ Interpolation	211
B.3	Factor Formulation	215
B.3.1	Pre-Integrated IMU Factor	215
B.3.2	IMU-Bias Factor	215
B.3.3	Between-Pose Factor	216
B.3.4	Velocity Factor	216
B.3.5	GNSS-PVT Factor	216
B.3.6	Pseudorange and Doppler-shift (PrDo) Factor	217
B.3.7	GNSS Receiver Clock Error Factor	217
B.4	Noise Models	217
C	Supplementary Materials for Continuous-Time Factor Graph Optimization	219
C.1	Implementation Details	219
C.1.1	Application I: GNSS/INS Fusion	219
C.1.2	Application II: GNSS-Multisensor Fusion	220
C.2	Extensive Experimental Analysis of GNSS/INS Fusion	221
C.2.1	General Solver Configurations	221
C.2.2	Computation Time	223
C.2.3	Size of Fixed Lag	223
C.3	Extensive Analysis of GNSS/Multisensor Fusion	226
C.3.1	Analysis of Challenging Scenarios	226
C.3.2	Smoother Type and Computation Time	229
C.3.3	GP-WNOA/WNOJ Motion Model	231
D	Supplementary Materials for M-Estimation	233
D.1	Details of M-Estimators	233
D.1.1	Supplementary Plots of Loss Functions	233

D.1.2	Fair Estimator	235
D.1.3	Cauchy Estimator	237
D.1.4	Geman-McClure Estimator	237
D.1.5	Welsch Estimator	237
D.1.6	Huber Estimator	239
D.1.7	Tukey Estimator	239
D.2	Extensive Experimental Results	241
E	Supplementary Materials for Learning-based Methods	249
E.1	Implementation and Training Protocols	249
E.1.1	Data Preparation	249
E.1.2	Training Protocols	253
E.2	Extensive Experimental Analysis	254
E.2.1	NLOS Classification	255
E.2.2	Pseudorange Error Prediction	255
E.2.3	Ablation Study of Proposed Deep-Learning Model	258
E.3	Supplementary Data Specifications	259
	List of Figures	268
	List of Tables	269
	List of Abbreviations	272
	List of symbol	277
	Bibliography	302

Author's Publications

The research presented in this dissertation is partially based on the following publications by the author:

- [M1] ZHANG, H. ; XIA, X. ; NITSCH, M. ; ABEL, D.: Continuous-Time Factor Graph Optimization for Trajectory Smoothness of GNSS/INS Navigation in Temporarily GNSS-Denied Environments. In: *IEEE Robot. Autom. Lett.* 7. no. 4 (2022), pp. 9115–9122. DOI: 10.1109/LRA.2022.3189824.
- [M2] ZHANG, H. ; CHEN, C.-C. ; VALLERY, H. ; BARFOOT, T. D.: GNSS/Multisensor Fusion Using Continuous-Time Factor Graph Optimization for Robust Localization. In: *IEEE Transactions on Robotics* 40 (2024), pp. 4003–4023. DOI: 10.1109/TRO.2024.3443699.
- [M3] ZHANG, H. ; WANG, Z. ; VALLERY, H.: Learning-based NLOS Detection and Uncertainty Prediction of GNSS Observations with Transformer-Enhanced LSTM Network. In: *IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 910–917. DOI: 10.1109/ITSC57777.2023.10422672.
- [M4] ZHANG, H.: Learning-based GNSS Uncertainty Quantification using Continuous-Time Factor Graph Optimization. In: *2025 1st German Robotics Conference*. 2025. DOI: 10.48550/arXiv.2503.04933.
- [M5] ZHANG, H.: Robust Statistics vs. Machine Learning vs. Bayesian Inference: Insights into Handling Faulty GNSS Measurements for Robust Vehicle Localization. In: *2nd Workshop on Safety of Intelligent and Autonomous Vehicles: Formal Methods vs. Machine Learning approaches for reliable navigation, IEEE IROS2025*. 2025. DOI: 10.48550/arXiv.2504.06015.

The following publications of the author are not related to this dissertation:

- [N1] NAUCK, C. ; LINDNER, M. ; SCHÜRHOFT, K. ; ZHANG, H. ; SCHULTZ, P. ; KURTHS, J. ; ISENHARDT, I. ; HELLMANN, F.: Predicting basin stability of power grids using graph neural networks. In: *New Journal of Physics* 24. no. 4 (Apr. 2022), p. 043041. DOI: 10.1088/1367-2630/ac54c9.
- [N2] KOTHARI, R. ; KARIMINEZHAD, A. ; MAYR, C. ; ZHANG, H.: Object Detection and Heading Estimation from Radar Raw data. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–7. DOI: 10.1109/IV55152.2023.10186591.
- [N3] MA, S. ; ZHANG, H. ; TOUSSAINT, M.: SVN-ICP: Uncertainty Estimation of ICP-Based LiDAR Odometry Using Stein Variational Newton. In: *IEEE Robotics and Automation Letters* 10. no. 11 (2025), pp. 12181–12188. DOI: 10.1109/LRA.2025.3619836.

CHAPTER 1

Introduction

“The difficulty of state estimation is often underestimated for real-world problems, and thus it is important to put it on an equal footing with control.”

– Timothy D. Barfoot [1, p.1]

1.1	Motivation	1
1.2	A Glance through Robust State Estimation	6
1.2.1	Outlier Exclusion	6
1.2.2	Robust Error Modeling	7
1.2.3	Multi-Hypothesis Methods	9
1.2.4	Solver Certification	11
1.2.5	Learning-based Methods	11
1.3	Delimitation	12
1.4	Research Objectives	13
1.4.1	Problem Statement	14
1.4.2	Research Questions and Hypotheses	15
1.5	Contributions	16
1.6	Thesis Outline	16

1.1 Motivation

The rapid growth of autonomous vehicles¹ is revolutionizing multiple facets of daily life, ranging from mobility, logistics to healthcare and emergency response. Among these, accurately determining a system’s state—framed as the *state estimation* problem—is crucial for downstream tasks such as decision-making and motion control [1]. The state of autonomous systems represents the key system variables of interest as they interact with their environments. However, accurately modeling or directly observing these states is often

¹In this thesis, the term “autonomous vehicle” encompasses all types of autonomous systems that move and interact within their environments, including street vehicles, mobile robots, surface vessels, and drones.

challenging due to unpredictable system behaviors and uncertainties caused by environmental influences or sensor failures.

The most well-established approach to address this issue involves utilizing redundant *state representations*, such as *dynamic models* and heterogeneous *sensing modalities*, into a fusion pipeline using a probabilistic state estimator [2, Chapter 25]. As the state representations are associated with a predefined noise distribution that defines a weighting factor, the state estimator can leverage *Bayesian inference* (for a mean) to determine an optimal state.

This fundamental principle has been implemented in many state-of-the-art state estimation methods that employ *least-square-like techniques*, where the noise models of all state representations are assumed to follow the *zero-mean Gaussian distribution* [3]. This assumption generally reflects the fact that a system's unimodal state can often be inferred from sensor data, where uncertainties are quantified based on the sensor's characteristics or signal quality.

However, assuming Gaussian noise often proves to be too conservative and unrealistic in real-world scenarios, where sensor malfunctions and signal interference frequently result in measurement outliers. For example, optical sensors like cameras and light detection and ranging (lidar) can misidentify landmarks in dynamic or deteriorating conditions, while ranging sensors such as global navigation satellite system (Global Navigation Satellite System (GNSS)) and ultra-wideband (UWB) suffer from corrupted measurements due to reflection errors or atmospheric interference. In such scenarios, measurement outliers often exhibit time-varying noise dynamics characterized by heavy-tailed, skewed, and multimodal distributions, which are challenging to model in advance. As a result, a state estimator based on the least-squares method may degrade or even fail entirely when faced with an inconsistent measurement model that relies on overly confident, manually tuned noise parameters.

As shown in Figure 1.1, this challenge arises consistently across various applications: a warehouse robot performing a visual simultaneous localization and mapping (SLAM) approach is frozen due to fatal loop closure registrations during mapping (Figure 1.1a); a mobile robot operating lidar SLAM suffers from incorrect feature associations, resulting in fuzzy lidar maps (Figure 1.1b); a surface vehicle suffers from positioning failures while crossing bridges (Figure 1.1c); and an autonomous vehicle's pose drifts in urban areas (Figure 1.1d).

Compared to a household robot (e.g., a robotic vacuum cleaner) that operates in a confined space, an autonomous vehicle in large-scale environments must handle complex scenarios containing diverse sources of sensor measurement degradation. For instance, a street vehicle equipped with GNSS may travel through large cities, where signal propagation consistently follows a Gaussian noise pattern, parameterized by the signal-to-noise ratio, similar to that observed in rural areas. However, when the vehicle enters urban canyons or tunnels, GNSS observations become degraded due to multipath effects and non-line-of-sight receptions, causing the noise distribution to shift to a multimodal, long-tailed form.

Over the decades, numerous methods have been explored to achieve accurate and robust vehicle localization, especially for long-term operations in large-scale environments. One of

the most straightforward strategies involves fusing the state observations from multiple heterogeneous sensors that can mitigate the impairment when a minority of sensors is degraded. This concept has been explored in various applications. Since traditional state estimators continue to assume sensor models follow a Gaussian noise distribution, which underestimates the complexity and magnitude of uncertainty, they struggle to efficiently downweight or exclude faulty sensor observations. Therefore, the performance of multisensor fusion for vehicle localization is compromised without advanced engineering and precise parameter tuning, limiting its scalability for long-term operations in large-scale environments.

In summary, when a traditional state estimator continues to assume sensor models follow a Gaussian noise distribution, it oftentimes **underestimates the complexity and magnitude of uncertainty**, leading to sub-optimal solutions or even failures. Consequently, designing a vehicle localization method that ensures robust, long-term operation across diverse environments remains a significant challenge.

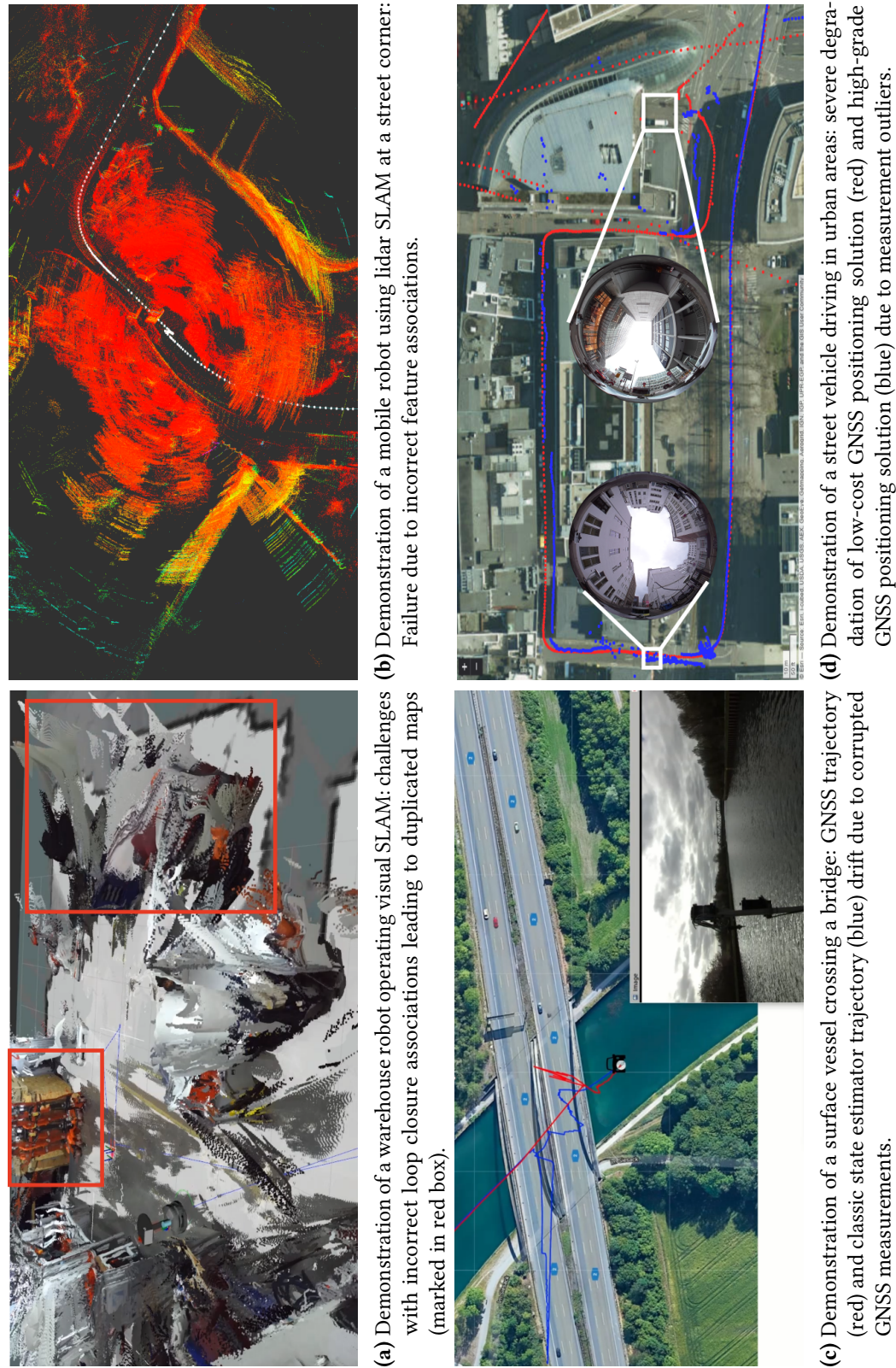


Figure 1.1: Demonstrations of failures in vehicle localization approaches due to measurement outliers.

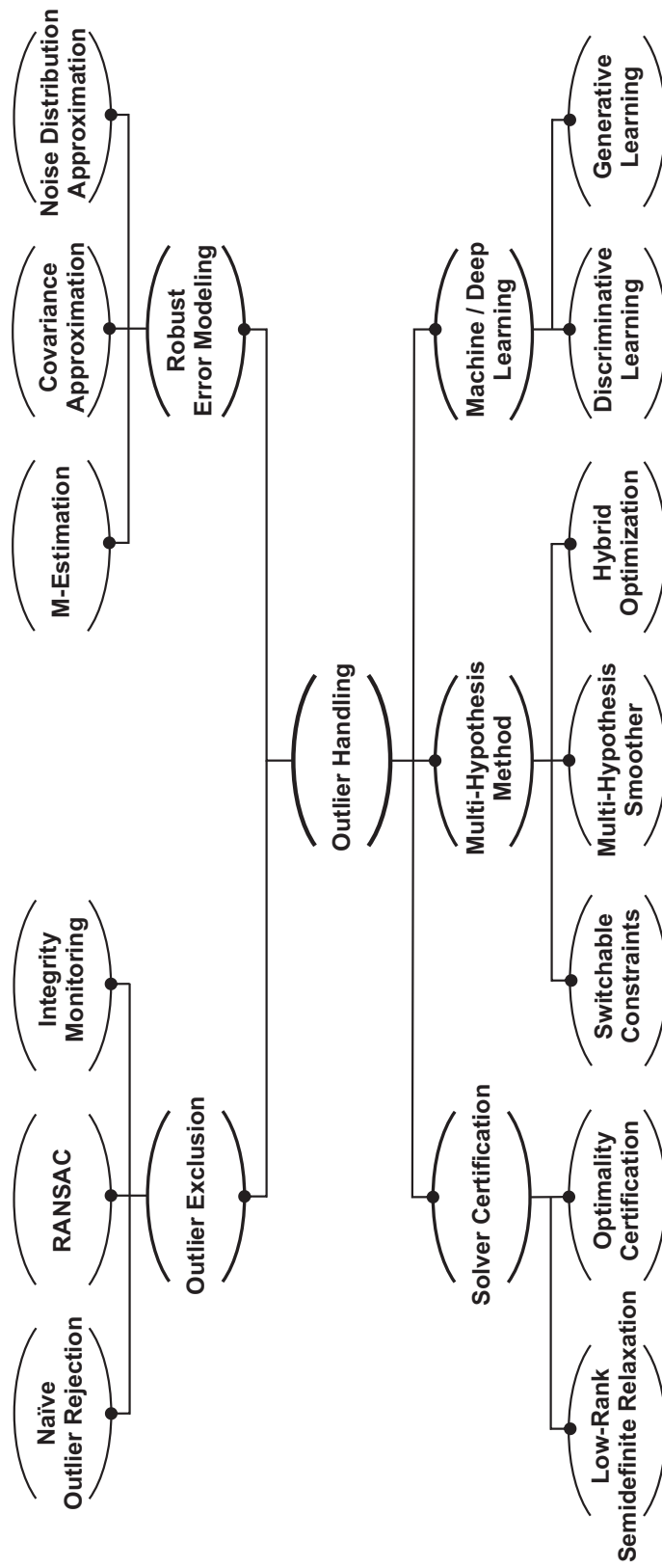


Figure 1.2: Scheme of common approaches to handle measurement outliers toward robust vehicle localization.

1.2 A Glance through Robust State Estimation

Given that faulty sensor observations are a critical factor in this problem, previous research has concentrated on addressing measurement outliers, which can be broadly categorized into four main approaches: *robust error modeling*, *outlier exclusion*, *model switching*, and *solver certification*. Figure 1.2 illustrates the hierarchy of related works on outlier handling within state estimation frameworks.

This section provides a brief overview of the most established approaches, grouped into five categories, to address measurement outliers and enhance robustness in state estimation applications, **laying a grounded foundation and a clear delimitation for the research objectives of this thesis.**

1.2.1 Outlier Exclusion

Among other approaches, *outlier exclusion* serves as a discriminative strategy for handling measurement outliers in state estimation problems. Rather than relying on statistical modeling of the outlier distribution, this approach uses heuristic methods to detect and exclude faulty measurements, ensuring they are not incorporated into the state estimator.

1.2.1.1 RANSAC

One of the most popular methods falling into this category is the *Random Sample Consensus* (RANSAC), initially introduced by Fischler and Bolles [4]. This algorithm iteratively fits the model that best aligns with the majority of inlier measurements. Measurements that deviate from the model are classified as outliers and are excluded from further processing [5].

RANSAC-like algorithms have demonstrated significant effectiveness across various applications, especially in scenarios with a high inlier-to-outlier ratio. These include point cloud registration [6, 7], vision-based localization [8–13], and GNSS outlier detection [14–16].

However, RANSAC-like algorithms are known to become inefficient and unstable when dealing with high outlier rates ($> 50\%$) [17], and they struggle to scale effectively in high-dimensional problems [18].

1.2.1.2 Naïve Outlier Rejection

Besides RANSAC, numerous application-specific approaches have been developed to handle *outlier rejection* in state estimation problems, such as rejecting faulty measurements in

SLAM [18–20] and robust solver design [21–23]. These approaches demonstrate that sophisticated algorithm designs can effectively prevent performance degradation by enhancing robustness.

1.2.1.3 Integrity Monitoring

Additionally, *integrity monitoring* offers a promising method for outlier rejection, extensively explored in GNSS-based applications [24] through statistical testing [25, 26] and interval analysis [27–29]. This concept has also been extended to assess the integrity of other sensor observations [30–32] and to develop integrity-augmented state estimators [33–35].

However, similar to RANSAC, integrity monitoring algorithms designed for sensor measurements typically assume the availability of sufficient healthy measurements to maintain consistency. Consequently, the calculated protection level loses accuracy when outliers dominate the measurement epoch or when there are too few measurements available. Another downside of integrity-based methods is their high computational cost, particularly in high-dimensional measurement spaces, which makes them difficult to generalize in large-scale applications.

1.2.2 Robust Error Modeling

Robust error modeling presents another effective approach to outlier rejection, achieved by either shaping the cost function or analyzing the outlier distribution to enhance resilience against outliers. Since least-square estimators formulate a quadratic optimization problem that minimizes the *sum-of-squared-error cost function*, they become highly sensitive to outliers that lead to unstable gradients during optimization epochs.

1.2.2.1 M-Estimation

In this scope, *M-estimation* (maximum likelihood type estimates) has been widely used thanks to its flexibility and generalizability, provided a proper scaling parameter in various applications [36, Chapter 3]. The original quadratic cost function of the state estimator is embedded within a bounded, nonlinear function (also known as a kernel function) that retains the same characteristics, such as the unique zero and monotonic increase [1, 37].

Various kernel functions can be applied to specific applications [38, 39]. These functions modify the covariance matrix to strongly penalize cost weighting as quadratic error increases, which is often a result of measurement outliers. Numerous state estimation applications utilize the M-estimate to enhance the robustness, including SLAM front-end [40–42] and back-end [43, 44], ranging-based localization approaches [45–47].

While M-estimation is well-established both theoretically and in practical applications, its effectiveness depends on selecting an appropriately parameterized kernel function. Furthermore, the adapted cost functions used in M-estimators often lose their *convexity*, leading to suboptimal solutions when relying on iterative local solvers. Therefore, this approach may not scale effectively to handle varying outlier patterns in long-term operations.

Recent works have been proposed to address those problems by either introducing a self-adapting mechanism for M-estimators [48, 49], generalized kernels [50–52], or graduated non-convexity [53, 54]. While these self-adapting methods have been shown to enhance the generalization of M-estimators in state estimation problems with time-varying noise levels, they remain limited to simple noise distributions that are unimodal and symmetric.

1.2.2.2 Covariance Matrix Approximation / Tuning

Unlike M-estimators, which achieve robustness by reshaping the cost function without explicitly modeling the appearance of outliers, an alternative approach focuses on quantifying noise behavior by estimating the covariance matrix or approximating the noise distribution.

Robust estimation methods that estimate the covariance matrix have a long history, tracing back to the 1990s in the context of adaptive filtering. In these methods, system noise, often represented by covariance matrices, can be estimated alongside the state estimate using techniques such as dual Kalman filters (e.g., Kalman-Bucy filters) or batch-processing approaches [55, Section 4.7].

Building on the idea that the covariance matrix can be approximated using batches of sampled residuals from ground truth or best-estimate states, approaches leveraging supervised model fitting and maximum-a-posteriori (MAP) *covariance estimation* provide practical strategies for addressing real-world problems. Since tuning covariance matrices or fitting noise models via maximum likelihood (ML) can lead to overfitting, MAP approaches have been widely explored in various research works, where a *conjugate prior distribution* within the exponential families (e.g., inverse-Wishart for multivariate Gaussian) based on empirical assumptions about the noise characteristics is introduced to formulate the MAP problem [56, Chapter 2.4], as demonstrated in studies such as [57, 58].

1.2.2.3 Noise Distribution Approximation

Often, there is limited information about measurement noise, and the data (e.g., measurement residuals) available for fitting noise models cannot be adequately sampled in specific applications. These challenges make it infeasible to formulate a MAP problem for reliable covariance inference.

Recent research efforts have addressed this problem by incorporating robust error modeling into the framework of *Variational Bayes*, which involves re-formulating the MAP problem to maximize the *evidence lower bound* (ELBO) given a predefined latent distribution, typically using the *expectation-maximization algorithm* (EM), as introduced in Bishop [56, Chapter 10.1]. In simpler cases, the latent distribution for measurement noise can be represented using a conjugate prior from the *exponential family*, demonstrating that the noise distribution can be efficiently modeled [59–62].

However, when the true noise distribution is complex—exhibiting long tails, multimodal, or asymmetric shapes that the exponential family cannot capture—an alternative approach utilizes universal function approximations, such as mixture models, to approximate the noise distribution within the problem setting of Variational Bayes. Numerous prior studies have used *Gaussian mixture models* to fit noise models by sampling measurement residuals, demonstrating their effectiveness in accurately identifying noise characteristics [63–66].

Since approximated noise distributions often exhibit complex, highly nonlinear, and non-convex shapes, interpreting the covariance matrix from these distributions for use in a least-squares solver may still yield suboptimal solutions.

Additionally, sampling measurement residuals presents its own challenge, as ground-truth measurement errors are generally unavailable. Sampled measurement residuals may exhibit a distribution shift, especially in dynamic, online applications, such as when a vehicle is in motion. These challenges remain an active area of research investigation at the forefront.

1.2.3 Multi-Hypothesis Methods

Since M-estimators with a single kernel function and fixed tuning parameters often fail to scale across diverse problems associated with multimodal error distributions (e.g., ambiguous loop-closure detection), numerous studies have explored *multi-hypothesis models* within the optimization framework to achieve robust state estimation.

Notably, I refer to multi-hypothesis models as those incorporating multiple noise modalities within the solver (e.g., graph optimization). Sensor observations may exhibit a multi-modal noise distribution, resulting in the inference of a single covariance matrix.

1.2.3.1 Switchable Constraint

Sünderhauf and Protzel [67] first introduced the concept of *switchable constraints* within graph optimization frameworks, where an additional switch variable is assigned to measurement constraints. This approach enables the iterative optimization process to gate outliers by appropriately selecting the variances of the switch variables.

Later, Agarwal et al. [68] advanced the concept of switchable constraints by introducing an analytical solution that dynamically scales the covariance of switch variables, eliminating the need for additional optimization and enabling faster convergence. This work also demonstrated that switchable constraints are equivalent to the M-estimator utilizing the Huber loss function.

However, numerous studies have shown that these approaches still assume well-formed noise distribution shapes, such as symmetry, limiting their effectiveness in challenging scenarios where skewed or multimodal distributions are prevalent [69].

1.2.3.2 Multi-Hypothesis Smoother

Another class of methods seeks to reformulate the optimization problem by incorporating multiple hypothesis measurement constraints derived from a multimodal distribution. These types of algorithms were initially introduced in tracking problems, where tracked objects encounter ambiguous associations [70]. Since ambiguous state constraints also pose challenges in localization methods, Hsiao and Kaess [71] extended the incremental smoother iSAM2 [72] to handle multi-modal measurements. Leveraging an efficient tree-based representation, multi-hypothesis measurements with ambiguous or redundant state constraints are eliminated and pruned through *Chi-square testing*, enhancing performance and robustness, particularly in scenarios with erroneous loop-closure detections [73].

1.2.3.3 Hybrid Optimization

Subsequently, Doherty et al. [74] generalized factor graph optimization for hybrid state estimation problems involving both discrete and continuous state variables. In this approach, the discrete variables to be estimated can assign a semantic cluster to the underlying continuous pose or represent different modes within a multimodal noise distribution, demonstrating effectiveness in SLAM applications with unknown landmark associations [75].

These novel approaches, which introduce alternative methods for solving hybrid inference problems, show considerable potential. However, open questions remain regarding the use of hybrid graph-optimization methods, particularly when initial distributions underlying the measurements are inconsistently modeled. Consequently, there is substantial scope for ongoing research into combining *multi-hypothesis optimizers* with the advanced noise distribution approximations discussed above.

1.2.4 Solver Certification

Although the advanced techniques introduced above have significantly improved the handling of measurement outliers, they still rely on standard least-squares solvers for state estimation. As a result, they may not consistently produce trustworthy, high-quality solutions, particularly when faced with highly *non-convex optimization* objectives. This issue often poses challenges in real-world applications, where reliable initial values are unavailable, or the actual noise distribution differs from the assumed observation model.

Given the solver’s central role in estimation applications—where its decisions on system states impact downstream tasks—research on certifiably correct state estimation has gained significant attention in recent years [76, 77]. The theoretical foundation of this approach is rooted in convex optimization [78], where non-convex optimization objectives can be relaxed within the framework of *semidefinite programming* (SDP). In this context, various algorithms have been developed to either achieve a globally optimal solution through semidefinite relaxations by leveraging the *low-rank structure* [79–81] or to certify the optimality of a local optimizer using *Lagrangian duality* [82–85].

Those algorithms demonstrated their superiority and effectiveness in small-scale applications, solving non-convex problems in the presence of challenging measurement models. However, as research on applying convex optimization techniques from mathematics to robotics is still in its early stages, further investigation is needed into the theory, particularly regarding the loss and recovery of *tightness* in robotic applications [86].

Additionally, the computational efficiency of using SDP relaxations, particularly for large-scale problems, remains a challenge to address. Since solver certification represents the final step in estimation applications, integrating this technique with methods for approximating noise distribution—used as inputs for optimizers—could be a valuable focus for near-term research.

1.2.5 Learning-based Methods

Developing robust estimation methods in the era of machine learning has attracted significant attention for addressing measurement outliers. Environment-dependent outliers can be challenging to identify and exhibit complex noise patterns that are difficult to model using traditional statistical methods. Learning-based approaches², however, offer substantial potential to model these nonlinear functions given sufficient data [56].

²Although many learning-based methods are rooted in the statistical inference introduced in Section 1.2.2 and thus share similar theoretical foundations, I intentionally categorize inference-based and learning-based methods as distinct approaches in this thesis. Specifically, I define learning-based methods as machine or deep learning approaches that rely on extensive offline training pipelines to develop models that cannot be represented using traditional statistical tools, such as conjugate distributions or mixture models.

Recent studies have leveraged universal nonlinear function approximators, including *kernel machines* and *neural networks*, within machine learning frameworks to detect and model measurement outliers. Given sufficient data sets with ground-truth labels, pre-trained models have demonstrated outstanding potential in classifying outliers in range observations [87–89] and predicting observation errors [90, 91].

While these approaches follow the concept of *discriminative modeling* that trains a nonlinear function approximator to directly predict or classify input labels, another class of methods takes a generative view of the learning paradigm, aiming to model the underlying data distribution [92].

Recently, *generative modeling* techniques, such as *normalizing flows*, have been explored to improve the robustness of state estimation algorithms against corrupted measurements [93, 94]. These studies demonstrate that non-Gaussian distributions can be effectively captured within the estimation framework, enhancing resilience to measurement anomalies.

Since the spirit of state estimation aligns closely with statistical analysis, where clear interpretations of state and uncertainty propagation based on data (i.e., measurements) are essential for validating estimation quality, learning-based methods generally fall short in interpretability. Their internal data transformations, used to make inference decisions, often lack transparency, which challenges their reliability. This challenge becomes even more strenuous due to the limited availability of training data, particularly for instances that capture measurement outliers, which are often caused by rare events in real-world applications.

Consequently, the scalability and reliability of learning-based outlier modeling, which is used to determine measurement weighting in state estimation algorithms proposed in recent studies, remain unproven for large-scale applications where out-of-distribution data must be processed.

In contrast, learning-based outlier handling has generally been recognized as a pre-processing step or a source of prior information in state estimation algorithms, where *uncertainty propagation* is not required. Nevertheless, as there is currently no evidence that learning-based methods can generalize across varying scenarios in vehicle localization, further investigation is needed into how learning concepts can be effectively integrated to enhance the robustness of state estimation applications.

1.3 Delimitation

Even though I have provided a brief overview of key concepts and state-of-the-art methods for outlier handling to support robust and consistent vehicle localization, additional ideas and approaches remain that have not yet been covered. Examples include outlier detection

methods leveraging other sensor modalities [95, 96] and end-to-end learning approaches [97–99].

As this thesis focuses specifically on algorithm design for robust state estimation with advanced outlier handling techniques, I seek to **study (a subset of) current algorithms and propose new ideas** within the context of online vehicle localization that hold significant potential for interpretability and scalability, which require

- efficient data structures. Therefore, I do not examine methods that incorporate additional sensor modalities (e.g., camera) for outlier rejection during the inference stage.
- interpretable data processing and uncertainty metrics. Therefore, I do not employ offline-trained models (e.g., neural networks) to infer measurement noise parameters directly within state estimators. Instead, I use and examine such models to infer prior information during the sensor pre-processing stage.
- efficient computation and modular design. Therefore, I do not explore nested and complex system pipelines that rely on a sophisticated engineering stack.

In addition, while accuracy and performance are often regarded as the primary metrics for evaluating state estimation approaches, this thesis does not aim to achieve the **best** possible localization accuracy. Instead, the focus is on robustness and consistency of state estimation in the presence of measurement outliers, as these properties are fundamental to ensuring reliable state estimation performance for long-term autonomy. I have no doubt that a **best-performed** state estimator can always be achieved by carefully fusing multiple sensor modalities for a given test scenario, which is generally an **engineering problem**.

However, understanding the mechanisms for handling measurement outliers, using GNSS-based vehicle localization as a representative case, constitutes a more demanding research challenge toward the development of generalized and sustainable algorithmic advances.

1.4 Research Objectives

Building on the motivation and foundational work introduced above, this thesis focuses on algorithm development for vehicle localization within a multisensor fusion framework, where **GNSS observations serve as the primary source for determining the vehicle’s position**, complemented by an inertial measurement unit (IMU) to enhance the observability of the vehicle’s velocities and orientation. Optical sensors, such as lidar and cameras, are used to provide prior information or maintain state observability when GNSS signals are degraded or unavailable.

1.4.1 Problem Statement

In particular, I investigate **outlier handling algorithms designed explicitly for GNSS code (aka. pseudorange) observations**, which form the basis of a ranging sensor model.

Investigating GNSS *pseudorange observations* for robust vehicle localization offers multiple advantages. Given that positioning signals are readily accessible with a low-cost GNSS receiver, using GNSS measurements for vehicle localization in outdoor environments provides globally referenced state observations that are both effective and essential for many applications.

Furthermore, although the 1-D (pseudo)range measurement provides a simple model, it is often heavily corrupted by atmospheric and environmental interferences, creating challenging noise patterns for least-squares estimators. However, multiple 1-D range measurements can be acquired to establish redundant state constraints through flexible satellite constellations, enabling the study and development of advanced methods for outlier handling.

The principal objective of this thesis is to study and develop well-advanced outlier handling techniques integrated into a GNSS/multisensor state estimator capable of

- delivering accurate vehicle states in a *global reference frame* for *long-term* vehicle localization in *large-scale*³ environments.
- handling GNSS measurement outliers associated with *complex noise patterns* (e.g., asymmetric, long-tail, and multi-modal) in challenging areas, avoiding algorithmic failures;
- supporting a *modular and scalable* system design for efficient computation in online applications.

I start by developing a generalized multisensor state estimator that is more comprehensive than classic Kalman filters. This estimator serves as the fundamental tool for studying outlier handling methods.

Next, I seek to gain an in-depth understanding of M-estimators from both theoretical and practical perspectives tailored to the applications introduced below. Given that such studies are currently absent in the research community, I conduct numerous experiments throughout this work to address our research questions and validate our hypotheses.

Finally, I explore learning-based approaches for outlier detection and prediction of measurement errors, providing an extensive discussion on their advantages and limitations. My objective is to offer valuable guidelines on integrating learning-based methods within probabilistic state estimators.

³As precise definitions for *long-term* and *large-scale* are absent in the literature, our research focuses on experiments derived from extensive measurement campaigns (min. 30 min) encompassing diverse driving scenarios.

1.4.2 Research Questions and Hypotheses

Throughout this thesis, I investigate the following research questions (denoted as Q), each of which is associated with its corresponding hypothesis (denoted as H).

Q1: How should a state estimator be designed to enable seamless and scalable integration of outlier handling algorithms?

H1.1: Compared to filtering-based approaches, graph optimization typically uses iterative nonlinear optimization techniques, allowing robust error models to be incorporated directly into the solver.

H1.2: In addition, graph optimization creates a local structure of vehicle dynamics that encompasses all historical states, measurements, and control variables, providing a natural framework for approximating error models with data-driven methods.

Q2: How should kernel functions for M-estimators be chosen and parameterized for ranging measurements? Additionally, given a proper kernel for range outliers, how can we effectively balance its robustness and efficiency in degraded environments where measurements are both insufficient and highly corrupted?

H2.1: A well-parameterized M-estimator for one scenario may not deliver the same performance in other scenarios, especially when the state estimator faces challenges from both measurement outliers and insufficient state observability.

H2.2: To effectively tune M-estimators, relying solely on statistical information, such as past residuals, is insufficient, as the noise behavior of measurement outliers can dramatically change due to environmental interference.

Q3: How should learning-based methods be integrated into probabilistic state estimators? Specifically, how effectively can learning-based methods generalize to predict measurement errors with limited training samples?

H3.1: Deep neural networks are expected to perform and generalize better than classic machine learning methods (e.g., kernel machines), particularly in handling range outliers that were not encountered during training.

H3.2: Outlier handling with discriminative models should be limited to the measurement pre-processing stage, as these models lack the necessary guarantee of consistent uncertainty prediction required for robust state estimation.

1.5 Contributions

The contribution of this thesis is summarized as follows:

- C1:** I propose a generalized GNSS/multisensor state estimator based on continuous-time factor graph optimization for robust vehicle localization. Additionally, I present comprehensive experiments and comparisons of the proposed approach against a Kalman filter. For details, see Chapter 3.
- C2:** I present an in-depth evaluation and discussion on the use of M-estimators for handling pseudorange outliers, examining various kernel functions and specific parameterizations. Furthermore, I extend this study to explore novel approaches, including graduated-non-convexity and auto-tuned M-estimators, with the goal of providing valuable guidelines for practical applications. For details, see Chapter 4.
- C3:** I propose a novel neural network architecture based on spatiotemporal modeling to detect GNSS non-line-of-sight signals and predict pseudorange errors. The proposed network is evaluated against state-of-the-art learning-based methods, including both classic machine learning models and deep networks. For details, see Chapter 5.

1.6 Thesis Outline

In this chapter, I present the motivation and research background of this thesis, followed by a discussion of related applications and our research objectives. The remainder of this thesis is organized as follows:

Chapter 2 introduces the vehicle localization applications discussed in this thesis, focusing on scenarios with raw sensor measurements containing various outliers in challenging environments. Three datasets are defined from these applications and are used in subsequent chapters to validate and evaluate the proposed methods.

Chapter 3 introduces generalized factor graph optimization with continuous-time trajectory representation, serving as the fundamental component for studying outlier handling techniques in this thesis. Additionally, I provide an in-depth evaluation of an Unscented Kalman Filter, offering valuable discussions and implementation guidelines. The research findings of this chapter have been published in Zhang et al. [M1] and Zhang et al. [M2]. The proposed state estimator, developed in this chapter, serves as the primary testbed for designing and evaluating subsequent algorithms that handle GNSS outliers.

Chapter 4 examines handling GNSS pseudorange measurement outliers using M-estimators. This study provides a comprehensive analysis of various loss functions and parameterizations, with a focus on their robustness and generalization in the presence of complex noise distributions over long-term measurement campaigns.

Chapter 5 addresses pseudorange outlier detection and error prediction using learning-based methods. I begin by applying classic machine learning models to classify non-line-of-sight GNSS signals. Next, I frame a spatiotemporal modeling problem, addressed through a novel neural network, to manage GNSS pseudorange outliers. The implemented learning-based methods are trained and evaluated on real-world datasets from geodetic locations. Additionally, I conduct experiments on model generalization with out-of-distribution data samples and perform ablation studies on model components to support in-depth discussions. The research findings of this chapter have been published in Zhang et al. [M3].

Chapter 6 summarizes the work presented in this thesis. Based on the results and discussions above, I also outline the limitations of this research and suggest areas for future investigation. Additionally, I introduce our open-source software implementations and available datasets, with the intention of contributing valuable resources to the research community. A summary of all research findings presented in this thesis has been published in Zhang [M4] and Zhang [M5].

Appendix A provides in-detail theoretical fundamentals and mathematical elements used throughout this thesis. I begin with an introduction to the probabilistic foundations to underpin state estimation problems, addressed through filtering-based and optimization-based estimators that utilize nonlinear least-squares techniques. Next, I introduce some practical implementations of navigation algorithms and machine learning approaches, which are used in this thesis.

Appendix B provides supplementary materials related to the proposed state estimator introduced in Chapter 3. While not essential for understanding the core content of this thesis, these materials are included to ensure comprehensive documentation.

Appendix C complements the implementation details of the proposed state estimators discussed in Chapter 3 and presents extensive experimental results for vehicle localization across all applications.

Appendix D provides supplementary materials on M-estimators and includes extensive plots of 2-D errors across all evaluated estimators.

Appendix E details the implementation and training protocols for all learning-based approaches. It also provides the remaining experimental results discussed in Chapter 5.

CHAPTER 2

Test Vehicles, Applications, and Datasets

“In theory, there is no difference between theory and practice - in practice, there is.”

– Yogi Berra

2.1	Motivation	19
2.2	Test Vehicles	20
2.3	Applications	22
2.3.1	Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments	22
2.3.2	Application II: GNSS/Multisensor Fusion in Urbanized Areas	23
2.4	Datasets	25
2.4.1	Dataset I: Inland Waterways	25
2.4.2	Dataset II: Urban Areas	25
2.4.3	Dataset III: Machine Learning	26

2.1 Motivation

This chapter will establish a practical perspective to address the research questions outlined in Chapter 1 by presenting our test vehicles, datasets, and application scenarios. My research on algorithms for handling global navigation satellite system (GNSS) measurement outliers in robust vehicle localization is applied to two contexts: (1) autonomous surface vehicles in inland waterways and (2) autonomous street vehicles in urban areas. In both settings, accurate and robust estimation of the vehicle’s state (e.g., pose, velocities) is essential to ensure the system’s fundamental functionality and safety.

While inland waterways generally offer open-sky environments with sufficient satellite visibility, GNSS signals can still encounter significant interference when passing under bridges or through deep-water locks. Conversely, vehicles operating in dense urban areas pose additional challenges for GNSS-based localization, as they must navigate large-scale environments with conditions ranging from open-sky to urban canyons. These diverse driving conditions

significantly strain GNSS-based localization, as measurements may be limited or heavily corrupted by multipath effects and non-line-of-sight receptions.

This chapter is organized as follows: Section 2.2 introduces the test vehicles equipped with the necessary sensors used to record raw measurements for the target applications described in Section 2.3. The recorded raw measurements are then used to generate datasets, the details of which are provided in Section 2.4.

2.2 Test Vehicles

In this thesis, I utilize two test vehicles to collect raw sensor measurements through real-world measurement campaigns, covering extensive routes across diverse driving scenarios in large-scale environments. Both test vehicles utilize the GNSS receivers as the primary sensor to track the vehicle’s position, supplemented by the inertial measurement unit (IMU) to enhance the observability of the vehicle’s velocities and orientation.

Figure 2.1a illustrates the test surface vessel equipped with a NovAtel PwrPark7D-E1 dual-antenna GNSS receiver¹ and a Microstrain 3DM-GX5 IMU². Additionally, a ZED-2i stereo camera³ is installed for monitoring purposes but is not utilized for vessel localization. All data is recorded on a real-time industrial computer, including the pulse-per-second signal (1PPS) at 1 Hz from the GNSS receiver. GNSS raw observations are recorded at 10 Hz, while the IMU measurements are sampled at 100 Hz.

I equipped a regular street vehicle as the second test vehicle, as shown in Figure 2.1b. Our sensor setup included two GNSS receivers, a Microstrain 3DM-GX5 IMU, and a Velodyne VLP-16 lidar⁴. Both the high-grade GNSS receiver (NovAtel PwrPak7D-E1) and the low-grade GNSS receiver (ublox f9p⁵) were equipped with dual GNSS antennas and served with RTK correction data received from a base station. A high-grade optical speed sensor, Correvit S-Motion DTI from Kistler⁶, was mounted on the trailer hitch on the vehicle’s rear side. I have manually calibrated the static transformations between different sensors mounted on the test vehicle’s roof rack. Static transformation of the speed sensor to other sensors was measured using a Leica total station.

The IMU data were acquired at 200 Hz, while lidar point clouds were recorded at 10 Hz. GNSS observations from the NovAtel and ublox receivers were recorded at 10 Hz and 5 Hz,

¹<https://novatel.com/products/gnss-inertial-navigation-systems/combined-systems/pwrpak7d-e1>

²<https://www.microstrain.com/inertial-sensors/3dm-gx5-25>

³<https://www.stereolabs.com/en-de/products/zed-2>

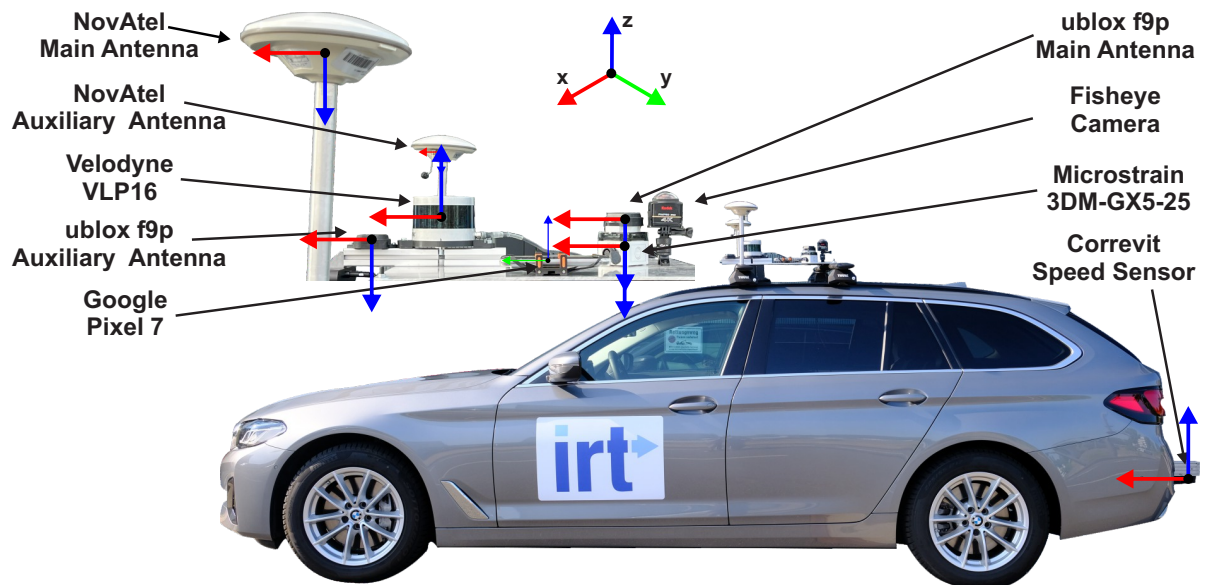
⁴<https://ouster.com/products/hardware/vlp-16>

⁵<https://www.u-blox.com/en/product/zed-f9p-module>

⁶<https://www.kistler.com/>



(a) Test Vehicle 1: Measurement setup on the ship "Datteln" with the primary antenna mounted on the top of IMU box



(b) Test Vehicle 2: Sensor setup and frames.

Figure 2.1: Test Vehicles.

respectively. I used the high-grade GNSS receiver NovAtel PwrPark7D-E1 with a dual constellation of GPS and Galileo satellite systems as a reference source. In addition to the sensor data, I received the pulse-per-second signal (1PPS) from the NovAtel receiver at 1 Hz to calculate the measurement delays. The RTCMv3 (RTK) correction data from the German satellite positioning service *SAPOS*⁷ was also stored at about 1 Hz for GNSS pre-processing.

2.3 Applications

This section introduces three applications to explore outlier handling methods within a novel graph-optimization-based state estimator.

2.3.1 Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments

The first application focuses on vessel localization in inland shipping scenarios by tightly coupling GNSS and IMU observations using two different state estimators. This application provides an ideal test environment for studying and comparing filtering-based and optimization-based state estimators in terms of performance, convergence, and robustness while fusing raw GNSS observations with IMU measurements.

Although inland vessels typically operate under conditions with sufficient satellite visibility, challenging scenarios arise when the vessel passes under bridges or through waterlocks, where GNSS observations can be severely corrupted or even entirely unavailable. Therefore, this application aims to address research question **Q1** and validate hypotheses **H1.1** and **H1.2** (see Chapter 1, Section 1.4.2).

I use the test surface vessel shown in Figure 2.1a and conduct measurement campaigns in the “Dortmund-Ems” water canal between the water lock in Waltrop and Dortmund harbor, as shown in Figure 2.2. Our test run covers a round trip comprising a total route of 30 km within 180 min. Along this route, the waterway passes under 14 bridges, creating multiple temporary GNSS-denied scenarios. Additionally, the GNSS observations are heavily corrupted by non-line-of-sight (NLOS) and multipath effects as the vessel approaches or leaves the bridges.

As the ship moves at a low average speed of approximately 19 km/h, the state observability is significantly challenged during bridge crossings. Consequently, this application provides an excellent testbed for studying different motion priors, as detailed in Chapter 3.

⁷<http://www.sapos.nrw.de>



Figure 2.2: Application I: test run in the Dortmund-Ems water canal.

2.3.2 Application II: GNSS/Multisensor Fusion in Urbanized Areas

In this application, I address the problem of street vehicle localization in urbanized areas using the test vehicle depicted in Figure 2.1b. In contrast to Application I, where the shipping environment does not inherently present challenging scenarios for GNSS-based localization methods, GNSS measurements in highly urbanized areas frequently experience significant signal propagation interference. These disturbances include not only signal loss during tunnel crossings but also complex error dynamics characterized by long-tail, multi-modal, or mixed-shape noise distributions, as analyzed by Hsu [100].

Therefore, Application II focuses on developing a generalized state estimator that leverages multisensor fusion through continuous-time factor graph optimization, building on the findings from Application I. The multisensor setup in this application incorporates a lidar sensor and, optionally, an optical speed sensor alongside GNSS and IMU. The proposed state estimator is sensor-agnostic, enabling robust trajectory estimation even in the presence of sensor degradation. Therefore, this application serves as the fundamental testbed to address research questions Q2 and Q3 (see Section 1.4.2).

The test vehicle equipped with sensors, as shown in Figure 2.1b, is utilized in this application to collect raw sensor data across various cities, including Aachen, Düsseldorf, and Cologne. These measurement campaigns encompass a diverse range of driving conditions, establishing comprehensive databases to validate the proposed state estimator and outlier handling algorithms. Figure 2.3 illustrates all test run routes, with their specifications detailed in Table 2.1.

2 Test Vehicles, Applications, and Datasets

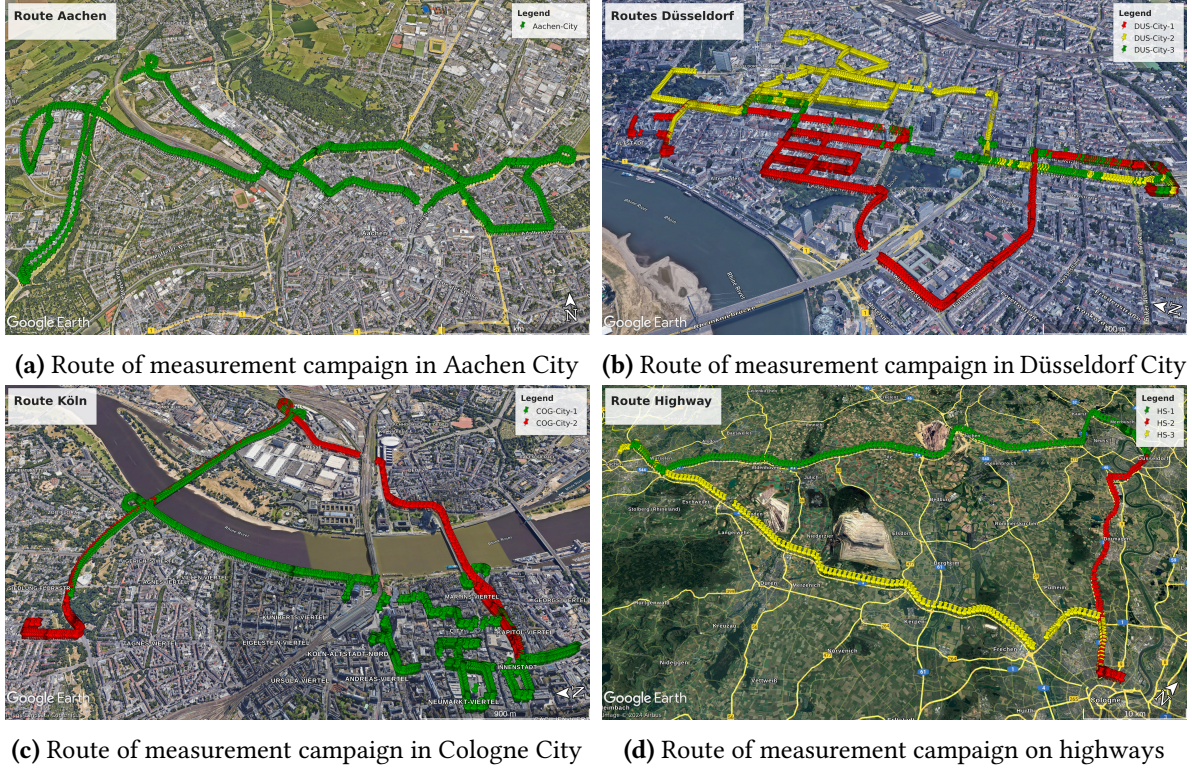


Figure 2.3: Application II: test run in the cities of Aachen, Düsseldorf, Cologne, and on the highways.

Table 2.1: Specification of measurement campaign in Application II. The variable $\bar{v}$ represents the average speed, and the scalar $\bar{n}^{\text{sat.}}$ is the average number of satellites used for a GNSS-PVA solution. I calculated the ratio of RTK-fixed solution and No-Solution due to insufficient GNSS observations, denoted by $R_{\text{fixed}}^{\text{RTK}}$ and R^{NS} , correspondingly. Please note that not all data is used in experiments.

Route	Total Length (km)	Duration (s)	$\bar{v}$ (km/h)	$\bar{n}^{\text{sat.}}$	$R_{\text{fixed}}^{\text{RTK}}$ (%)	R^{NS} (%)
Aachen-City	17.0	2477	27.25	11	76.51	1.7
DUS-City-1	6.9	1717	13.90	8	52.22	3.47
DUS-City-2	9.7	2417	14.72	8	49.87	2.95
DUS-City-3	4.17	1059	8.22	9	42.75	32.76
COG-City-1	14.96	2576	25.92	9	52.44	17.23
COG-City-2	7.45	1197	21.26	12	92.15	1.70
HS-1	84.76	3909	83.35	14	91.5	2.65
HS-2	56.5	2065	59	13	83.5	5.06
HS-3	75.97	2775	100	13	87.5	5.15

2.4 Datasets

Given the extensive dataset recorded in both applications, I extract a subset of the raw sensor measurements for the experimental studies by discarding redundant driving conditions. This process yields two datasets covering inland shipping and driving scenarios. Additionally, I incorporate the dataset from Zhang et al. [90], which includes both raw GNSS measurements and labeled GNSS faults from Hong Kong, to define the third dataset. This dataset is utilized in Chapter 5 to train learning-based models for GNSS fault detection.

2.4.1 Dataset I: Inland Waterways

I use the raw GNSS and IMU measurements from a single trip in the measurement campaign described in Section 2.3.1 to generate the first dataset used in Chapter 3. The characteristics of the dataset are summarized in Table 2.2. Within this dataset, I define four scenarios: the total route, an open-sky canal without bridge crossings, a route with three bridge crossings, and harbor areas.

Table 2.2: Specification of Dataset I. The variable $\bar{v}$ represents the average speed, and the scalar $\bar{n}^{\text{sat.}}$ is the average number of satellites used for a GNSS-PVA solution. I calculated the ratio of RTK-fixed solution and No-Solution due to insufficient GNSS observations, denoted by $R_{\text{fixed}}^{\text{RTK}}$ and R^{NS} , correspondingly.

Sequence	Specification	Total Length (km)	Duration (s)	$\bar{v}$ (km/h)	$\bar{n}^{\text{sat.}}$	$R_{\text{fixed}}^{\text{RTK}}$ (%)	R^{NS} (%)
#1	Total Route	15.62	5531	10.24	12	90.44	1.91
#2	Open-Sky	1.49	750	7.22	14	99.0	0.11
#3	Bridge-Crossing	0.79	350	10.31	10	65.8	16.97
#4	Harbor	0.50	232	7.74	12	100	0

2.4.2 Dataset II: Urban Areas

Our second dataset comprises different driving scenarios: open-sky, semi-urban, and high-speed track, extracted from measurement campaigns in Application II (see Section 2.3.2).

To enable a clear evaluation of the proposed state estimator in Chapter 3, I define various test sequences across multiple measurement campaigns and analyze the driving conditions for each sequence, as detailed in Table 2.3. These test sequences include lengthy runs, with routes extending up to 17 km, to assess the estimator's performance over long-term operations.

Table 2.3: Test sequences definition. I denote the test sequences in Aachen, Düsseldorf, Cologne, and high-speed tracks with “AC”, “DUS”, “C”, and “HS”, respectively. The variable $\bar{v}$ represents the average speed, and the scalar $\bar{n}^{\text{sat.}}$ is the average number of satellites used for a GNSS-PVA solution. I calculated the ratio of RTK-fixed solution and No-Solution due to insufficient GNSS observations, denoted by $R_{\text{fixed}}^{\text{RTK}}$ and R^{NS} , correspondingly.

Sequence	Total Length (km)	Tunnel Length (m)	Duration (s)	$\bar{v}$ (km/h)	$\bar{n}^{\text{sat.}}$	$R_{\text{fixed}}^{\text{RTK}}$ (%)	R^{NS} (%)
AC	17.0	270	2477	27.25	11	76.51	1.7
DUS	5.25	-	1350	13.48	8	52.06	0.9
C01	0.81	276	160	17.89	7	60.8	31.78
C02	1.45	145	390	13.36	7	37.74	11.56
HS	10.6	-	300	124.82	14	94.9	1.47

For urban scenarios, I selected data from areas with varying urbanization levels, including tunnel and bridge crossings, to examine the limitations of the proposed fusion approaches. Additionally, I incorporated open-sky scenarios on high-speed tracks, where the vehicle reached a maximum speed of 170 km/h, inducing significant motion distortion in the lidar point clouds.

2.4.3 Dataset III: Machine Learning

In contrast to the previously mentioned datasets used for vehicle localization, the third dataset consists solely of GNSS measurements and is utilized to train the learning-based models proposed in Chapter 5.

The raw GNSS measurements are categorized into two labels: healthy GNSS measurements and non-line-of-sight (NLOS) GNSS measurements. The Hong Kong dataset includes pre-existing labels derived using fisheye camera images and the ground-truth trajectory obtained from the high-grade GNSS-INS system, NovAtel SPAN-CPT. This dataset comprises four measurement campaigns conducted in highly urbanized areas of Hong Kong, as illustrated in Figure 2.4. I developed a data generation process to label the raw GNSS measurements collected in Aachen, as described in Section 5.3.1. This section focuses on providing details about the dataset and its analysis.

Table 2.4 summarizes the essential data characteristics in this dataset. The data from Aachen (AC) only contains GPS measurements. The Hong Kong data (HK) comprises measurements from multiple satellite constellations, with up to 74 unique satellites observed across different timestamps. These constellations include GPS, Galileo, BeiDou (China), GLONASS (Russia), and QZSS (Japan). Data from Düsseldorf (DUS) with raw GPS and Galileo measurements do not have labels and are therefore only used for testing.

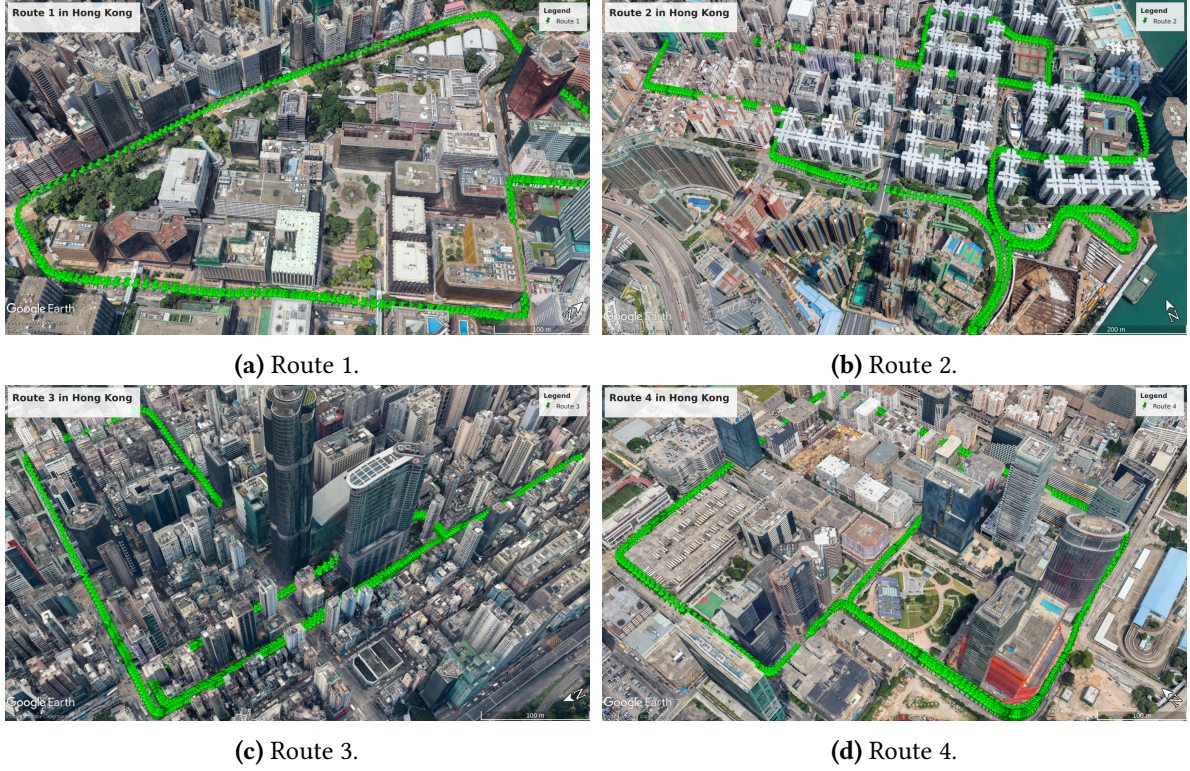


Figure 2.4: Measurement campaigns in Hong Kong.

Table 2.4: Details of Dataset 3. The average, maximal, and minimum number of received satellites are denoted with $n_{\text{avg}}^{\text{sat}}$, $n_{\text{max}}^{\text{sat}}$, and $n_{\text{min}}^{\text{sat}}$. The maximal pseudorange error $\sigma_{\text{max}}^{\rho}$ is inferred with ground-truth positioning. I present the ratio of LOS and NLOS labels in the dataset with R^{LOS} and R^{NLOS} .

Data	Duration (s)	$n_{\text{avg}}^{\text{sat}}$	$n_{\text{max}}^{\text{sat}}$	$n_{\text{min}}^{\text{sat}}$	$\sigma_{\text{max}}^{\rho}$ (m)	R^{LOS} (%)	R^{NLOS} (%)
HK	10573	16	25	4	532.5	60.62	39.38
HK (GPS only)	10573	5	9	1	276.08	60.47	39.53
AC (GPS only)	2366	6	8	0	640.32	77.62	22.38
DUS (GPS+GAL)	810	7	11	2	355.74	-	-

As a modern metropolis, Hong Kong offers a unique urban landscape compared to Aachen, resulting in notable differences in GNSS data. The city is marked by dense building configurations and an abundance of towering skyscrapers, as shown in Figure 2.4. Consequently, GNSS signals transmitted from satellites are more prone to multipath effects, particularly NLOS signals. Compared to the HK data, which can receive more NLOS observations due to its high urbanization rate, the AC data contains only 22.38 % NLOS receptions, resulting in an unbalanced database with significantly more LOS observations.

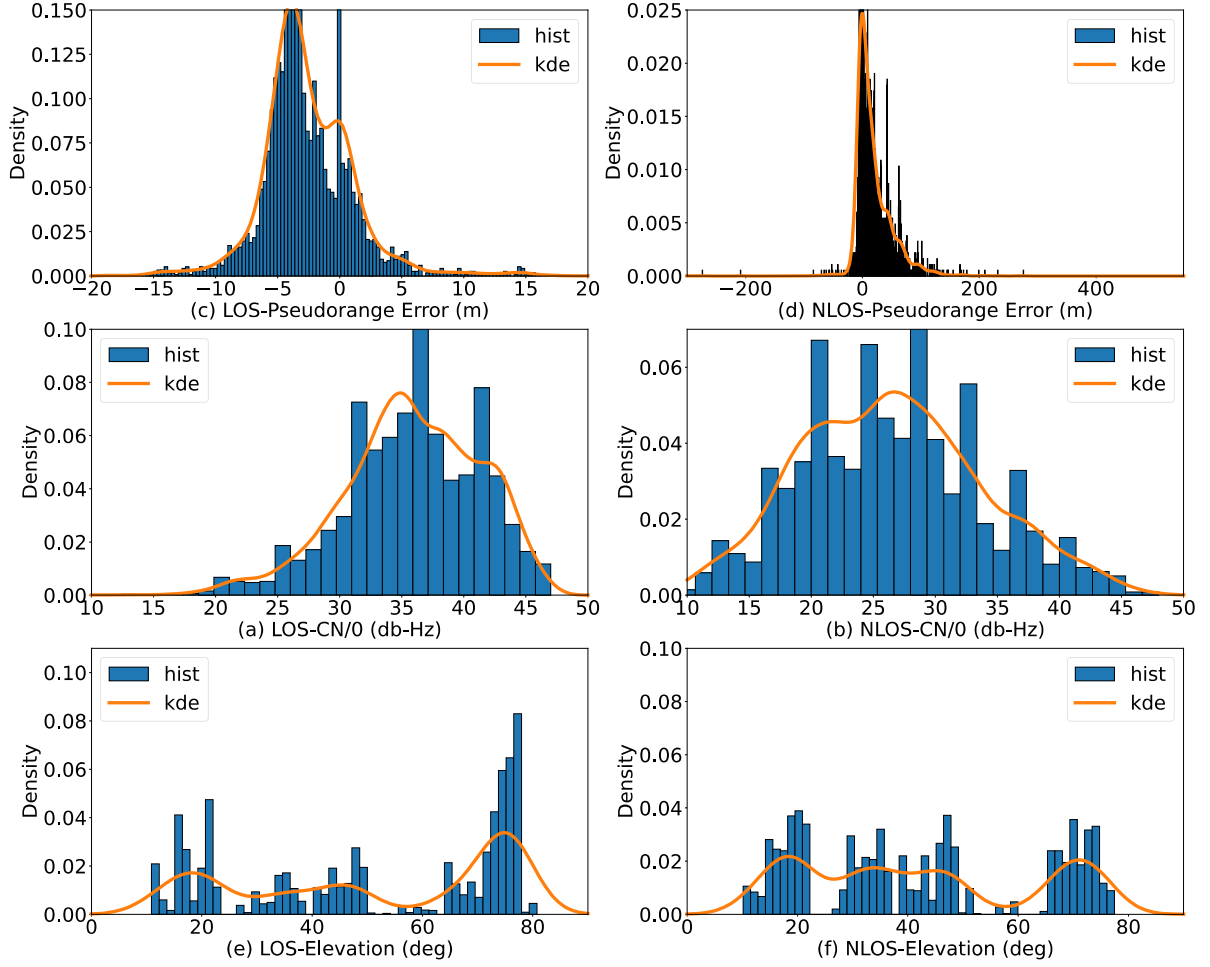


Figure 2.5: GPS data specification from Hong Kong.

To better understand the data from both cities, I plotted key features of GPS measurements over the entire observation period, classifying them into two categories: healthy measurements (LOS) and non-line-of-sight measurements (NLOS). These are presented in Figure 2.5 and Figure 2.6. As supplementary materials, key features for other satellite systems in the Hong Kong data are provided in Appendix E.

The pseudoranges of healthy GNSS measurements (Figure 2.5a) in Hong Kong exhibit higher errors compared to those in Aachen (Figure 2.6.(a)), indicating that the urban environment in Hong Kong introduces more significant challenges to GNSS signal accuracy. Additionally, it can be inferred that the Hong Kong data contains a higher proportion of NLOS measurements (including multipath receptions), as evidenced by the greater number of delays (positive pseudorange errors) observed in Figure 2.5.(b). This pattern is not present in the data from Aachen, as shown in Figure 2.6.(b).

When comparing the Carrier-to-Noise density ratio (C/N_0) from healthy GNSS measure-

ments between the two datasets, it is evident that the Aachen data exhibits higher C/N_0 values (see Figure 2.6a) compared to the data from Hong Kong. This observation highlights the inherent biases in both datasets, which contribute to a distribution shift in the features learned by machine learning models.

Another notable observation arises from the C/N_0 values for unhealthy measurements. As originally discussed in Hsu [101], NLOS or multipath GNSS observations can exhibit higher signal strength (i.e., higher C/N_0), independent of signal reflections in urban canyons. This phenomenon highlights the limitations of traditional fault exclusion methods, which rely solely on C/N_0 values, as they may fail to accurately distinguish between healthy and unhealthy measurements.

A similar conclusion applies to the evaluation of elevation angles, as shown in Figure 2.5f, where NLOS receptions can also originate from satellites positioned directly overhead of the vehicle.

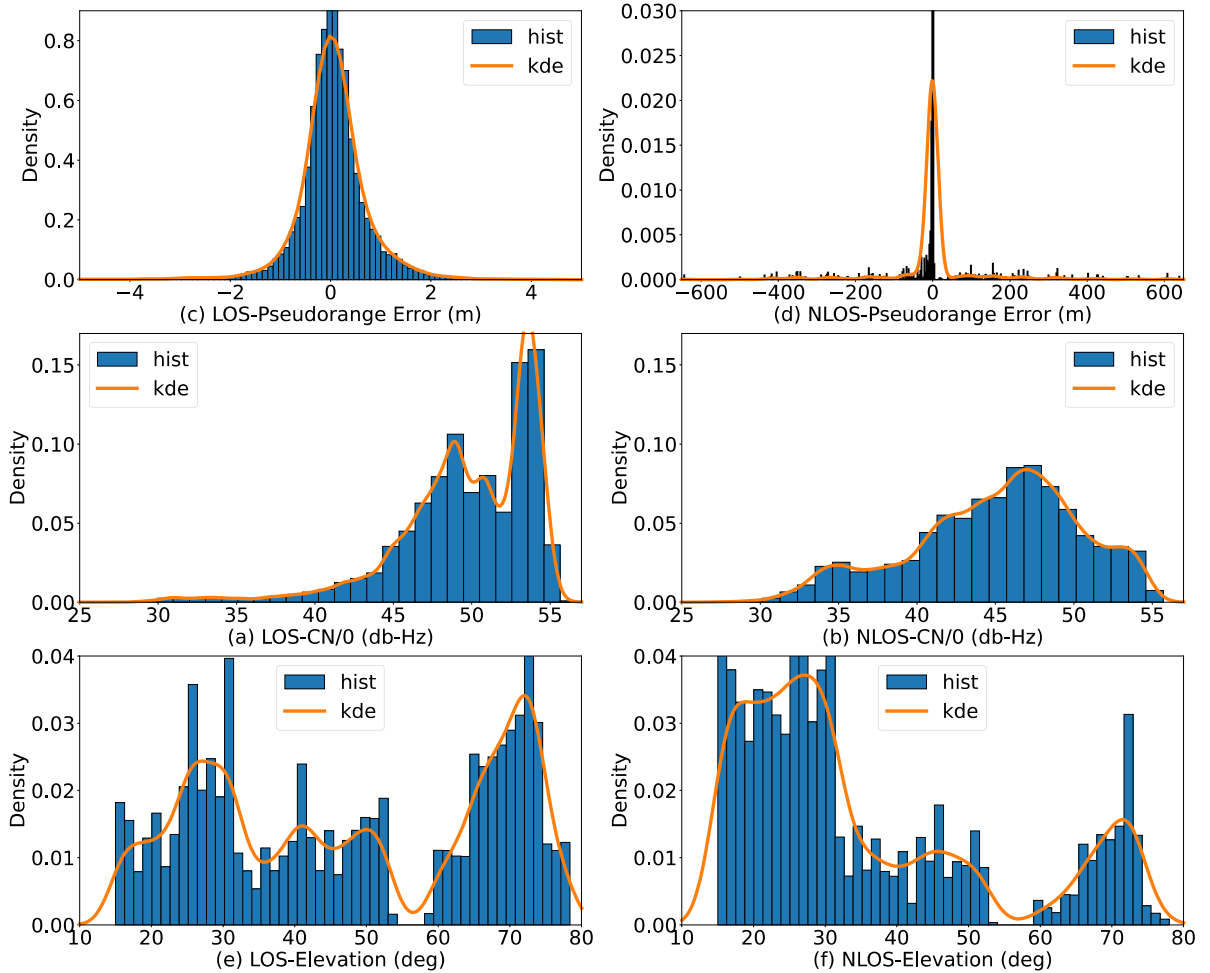


Figure 2.6: GPS data specification from Aachen.

CHAPTER 3

Continuous-Time Factor Graph Optimization

“Factor graphs can represent a wide variety of problems across robotics, expose opportunities to improve computational performance, and are beneficial in designing and thinking about how to model a problem, even aside from performance considerations.”

– Frank Dellaert [102]

3.1	Motivation and Objective	32
3.2	Related Work	33
3.2.1	Graph Optimization for GNSS-based Vehicle Localization	33
3.2.2	Graph Optimization for Multisensor Fusion	34
3.2.3	Continuous-Time Trajectory Representation	35
3.2.4	Modular Estimation Framework	36
3.3	GP-based Continuous-Time Trajectory Representation	36
3.3.1	The Gaussian Process Model	37
3.3.2	The Gaussian Process Motion Prior in $SE(3)$	38
3.3.3	GP-WNOA Motion Prior	39
3.3.4	GP-WNOJ Motion Prior	41
3.4	Time-Centric Factor Graph Optimization	43
3.5	Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments	47
3.5.1	UKF for Tightly Coupled GNSS/INS Fusion	47
3.5.2	FGO for Tightly Coupled GNSS/INS Fusion	49
3.5.3	System Overview	49
3.5.4	Essential Experimental Results	50
3.5.5	Discussion on Application I	56
3.6	Application II: GNSS/Multisensor Fusion for Robust Vehicle Localization .	57
3.6.1	Loosely Coupled Factor Graph Optimization	57
3.6.2	Tightly Coupled Factor Graph Optimization	58
3.6.3	System Overview	59
3.6.4	Essential Experimental Results	60
3.6.5	Discussion on Application II	68

3.1 Motivation and Objective

This section aims to develop a state estimation algorithm that serves as the fundamental test bed for studying outlier handling techniques. Given the comprehensive review of all related technical foundations and my discussions on Kalman filters and factor graph optimization in Section A.2.9, I begin by examining GNSS and IMU integration in temporarily GNSS-denied environments using the classic unscented Kalman filter and continuous-time factor graph optimization. My objective is to comprehensively analyze their respective performance, highlighting their strengths and limitations for GNSS-based vehicle localization in challenging scenarios.

Additionally, I address the degradation problem of GNSS-based localization approaches by translating classic FGO for multisensor fusion into an approach where the graph associated with all to-be-estimated state variables is constructed deterministically based on a priori chosen timestamps. It thus presents a *time-centric factor graph* construction that is independent of any particular reference sensor (e.g., GNSS).

This approach incorporates a Gaussian Process (GP) motion prior using the white-noise-on-jerk (WNOJ) motion model, as proposed initially by Tang et al. [103]. The algorithm independently feeds new observations from each sensor into the factor graph without measurement-to-state synchronization. If a measurement cannot be temporally aligned with any state variable, I query a GP-interpolated state corresponding to the measurement used for the error evaluation.

Consequently, the proposed state estimator generalizes the sensor fusion mechanism, mitigating the effects of sensor degradation and thereby improving the accuracy and robustness of trajectory estimation. Besides, the novel state estimator provides a flexible and extensible data interface for applying algorithms that handle outliers, as presented in the upcoming chapters.

My contributions with associated hypotheses in this chapter are as follows:

1. I compare the performance of the Unscented Kalman Filter (UKF) and Factor Graph Optimization (FGO) with respect to localization accuracy and trajectory smoothness in inland shipping scenarios. This comparison provides the basis for addressing research question **Q1** in Section 1.4.2 and supports my hypothesis that a graph-optimization-based state estimator is preferable for studying the handling of measurement outliers, not only because it offers performance advantages beyond the Markov property, but also because it can incorporate advanced modeling techniques.

2. I propose a flexible, online, continuous-time factor graph optimization framework that can accommodate common multisensor fusion problems. The flexibility comes from the fact that (i) I can accommodate asynchronous measurements, and (ii) I choose estimation timestamps independent of any particular sensor frequency. This latter feature, along with the smoothing effect of a motion prior, provides robustness in the presence of any particular sensor dropout.
3. The proposed method, `gnssFGO`¹, has been extensively studied for vehicle localization in challenging scenarios. It supports both loosely coupled and tightly coupled fusion frameworks, enabling the integration of GNSS measurements with other onboard sensors, and provides comprehensive evaluations and discussions of localization accuracy, robustness, and run-time efficiency. In addition, I evaluated the Gaussian process (GP) motion prior, implemented using the white-noise-on-acceleration (WNOA) and white-noise-on-jerk (WNOJ) models, to analyze the accuracy of interpolated states.

This chapter is organized as follows: Section 3.2 provides a comprehensive review of related work, highlighting their limitations and identifying research gaps. Section 3.3 introduces the theoretical foundations of Gaussian-process-based continuous-time trajectory representation, which is used to implement time-centric factor graph optimization in Section 3.4. Experimental studies and evaluations of the proposed method are presented in Sections 3.5 and 3.6, followed by the discussion of each practical application.

3.2 Related Work

3.2.1 Graph Optimization for GNSS-based Vehicle Localization

Most navigation applications rely on fusing GNSS observations with high-frequency measurements from an inertial navigation system (INS) using Kalman filters. However, even advanced Kalman-filter-based algorithms for state estimation can suffer from degraded trajectory smoothness and robustness when GNSS observations are temporarily unreliable or inconsistently corrupted. In such scenarios, the fundamental assumptions of first-order Markov processes and zero-mean Gaussian error distribution become invalid [104]. This issue commonly arises, for instance, when passing under a broad bridge where GNSS observations are significantly corrupted, as illustrated in Figure 1.1c.

In recent years, factor graph optimization (FGO) has gained significant attention for fusing GNSS observations in robust vehicle localization. Unlike filtering-based approaches, FGO performs batch optimization, where all measurement models are iteratively re-linearized and re-evaluated, yielding more robust state estimation even in the presence of measurement outliers. Previous studies demonstrated the feasibility of robust localization in urban areas by

¹<https://github.com/hz658832/gnssFGO.git>

factoring pseudoranges with robust error models [105, 106]. Later, Wen et al. [104] and Zhang et al. [M1] showed that FGO generally outperforms Kalman filters in terms of trajectory precision and smoothness.

GNSS data can be incorporated into the graph using either a loosely or tightly coupled schema [107]. Loosely coupled fusion integrates GNSS positioning solutions into the graph, enabling rapid convergence and high accuracy when high-quality Real-Time-Kinematic (RTK)-fixed GNSS solutions are available. Conversely, tightly coupled fusion incorporates raw GNSS observations, such as code or carrier-phase measurements, directly as state constraints. This approach establishes multiple state constraints for the received satellites, offering enhanced robustness over loose coupling [104, 108, 109].

Significant advancements in tightly coupled FGO have been demonstrated. Wen and Hsu [110] incorporated double-differenced pseudorange (DDPR) and double-differenced carrier-phase (DDCP) measurements, leading to improved performance. This work was extended to efficiently model carrier-phase constraints across multiple satellite epochs [111]. In another study, time-differenced carrier-phase (TDCP) measurements were combined with cycle-slip estimation to achieve accurate localization with better availability compared to DDCP when satellites are continuously tracked [112]. Congram and Barfoot [113] proposed GPS odometry using TDCP, demonstrating reduced drift compared to visual odometry. However, carrier-phase observations are susceptible to disturbances in dense urban environments, which limits their robustness.

As FGO provides a convenient framework for robust error modeling [114], several works have employed m-Estimators [115] to reject faulty GNSS observations [M1, 106, 112, 116]. Recently, FGO has been extended to address noise distribution identification and adaptive outlier rejection in GNSS-based vehicle localization [117, 118], showing promising results in achieving consistent trajectory estimation.

3.2.2 Graph Optimization for Multisensor Fusion

While the aforementioned works have primarily focused on graph optimization for GNSS observations, they may still experience performance degradation in complex scenarios where GNSS measurements are lost or contain significant outliers. To address this limitation, another research domain focuses on integrating additional sensor modalities alongside GNSS observations into the graph, with many applications rooted in simultaneous localization and mapping (SLAM).

For outdoor scenarios, a pose graph integrating GNSS position measurements and lidar odometry with loop-closure constraints demonstrated improved runtime efficiency and localization performance compared to lidar-only approaches [119]. Similarly, Shan et al. [120] fused feature-based lidar odometry and loop-closure constraints with synchronized GNSS measurements into a factor graph, achieving drift-free pose estimation. This pose estimate was

subsequently incorporated into a second graph optimization process with pre-integrated IMU measurements, enabling high-frequency, real-time state estimation. Beyond odometry, lidar maps have also been utilized for GNSS visibility assessment, helping to secure localization [121].

Camera-centric sensor fusion has gained traction, synchronizing camera data with other sensor modalities, such as GNSS and IMU observations, for graph-based optimization [122–124]. For example, Wang et al. [125] fused camera, lidar, wheel odometry, GNSS positioning solutions, and IMU measurements into a single graph, achieving consistent localization in featureless environments for long-term runs. Meanwhile, works such as Wisth et al. [126] and Shao et al. [127] explored multisensor fusion architectures without GNSS, demonstrating the importance of data synchronization and well-constructed graphs for integrating heterogeneous sensors.

Recent research has introduced multi-graph structures to enable more flexible and compact sensor fusion. For instance, Nubert et al. [128] integrated IMU, GNSS, and lidar observations into separate graphs with a switching mechanism. When GNSS signals were unavailable, the system activated the lidar-centric graph. Similarly, Zhao et al. [129] combined loosely and tightly coupled fusion schemes, associating each sensor modality with a separate graph. These graphs proposed odometry factors to an IMU-centric graph, which provided real-time state estimates. However, multi-graph structures can result in redundant and complex system architectures that demand meticulous engineering and do not fully address challenges posed by inconsistent noise distributions in corrupted sensor data.

Some approaches leverage high-frequency IMU measurements to coordinate multisensor fusion. For example, asynchronous global pose measurements (e.g., GNSS) are propagated to match the timestamps of visual-inertial factors using pre-integrated IMU data [130, 131]. This concept has been extended to a forward-backward IMU pre-integration mechanism, allowing precise association of asynchronous measurements with keyframes [132]. However, these methods remain sensitive to noisy IMU sensors, which introduce additional uncertainty.

3.2.3 Continuous-Time Trajectory Representation

An essential requirement for flexible graph-based multisensor fusion is the ability to query states associated with observations during the iterative optimization process. This requirement can be met by representing the trajectory in continuous time. In Furgale et al. [133], B-splines were introduced as a parametric method for continuous-time trajectory representation, later applied to stereo-inertial odometry [134]. Another approach involves exactly sparse Gaussian process (GP) regression, assuming that system dynamics follow a linear time-varying stochastic differential equation (LTV-SDE) [135]. Typically, the system dynamics are modeled as white noise on acceleration (WNOA). The reliability of this surrogate

dynamics model has been validated in works such as Mukadam et al. [136], Yan et al. [137], and Dong et al. [138].

Recently, Tang et al. [103] proposed an enhanced system dynamics model based on a white-noise-on-jerk (WNOJ) assumption in the LTV-SDE framework. Their work demonstrated that WNOJ could more accurately capture vehicle dynamics, making it suitable for systems with complex dynamics. As highlighted in Johnson et al. [139] and Talbot et al. [140], continuous-time trajectory representation using GPs is particularly advantageous when measurement times align with estimation times. Following this insight, I adopt the GP-WNOJ model proposed in Tang et al. [103] as a motion constraint between states and as a state interpolator to handle asynchronous measurements.

Although continuous-time trajectory representation has been explored for localization and mapping problems, including the extension of incremental smoothing with sparse GP interpolation to reduce computation time [137], the fusion of GNSS observations with multiple heterogeneous sensor measurements for online vehicle localization has not yet been extensively investigated or discussed.

3.2.4 Modular Estimation Framework

As the aforementioned approaches share similar procedures for solving estimation problems, the idea of a modular estimation framework that unifies system design across different applications has emerged. Labbé and Michaud [141] initially proposed a real-time mapping framework to manage internal map memory for loop closure detection. This framework was later extended in Labbé and Michaud [142] to support multisensor fusion and benchmarking. Its modularity and scalability have led to numerous derivative works [143, 144]. Similarly, a plug-and-play SLAM framework was introduced in Colosi et al. [145]. More recently, Solà et al. [146] proposed a tree-based estimation framework, structuring robot entities such as hardware, trajectory, and map management into separate branches. Sensor measurements and prior information are fused using a decentralized strategy, where primary sensors are selected in a configuration file to actively create new keyframes (state variables).

However, none of these modular frameworks incorporates continuous-time trajectory representation, which still requires measurement alignment to keyframes, inevitably leading to a loss of sensor observations.

3.3 GP-based Continuous-Time Trajectory Representation

This section presents a comprehensive introduction of the continuous-time trajectory representation based on a Gaussian process motion prior, which was originally proposed by

Barfoot et al. [147] and serves as the fundamental dynamic model underlying my proposed state estimator for fusing GNSS and multi-sensor measurements in Section 3.4.

3.3.1 The Gaussian Process Model

Barfoot et al. [135] originally proposed a continuous-time trajectory representation using Gaussian process regression, which presents an exactly sparse kernel by assuming the system dynamics follow a linear time-invariant stochastic differential equation (LTI-SDE):

$$\begin{aligned}\dot{\gamma}(t) &= \mathbf{A}\gamma(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{F}\mathbf{w}(t), \\ \mathbf{w}(t) &\sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \cdot \delta(t - t')), \end{aligned} \quad (3.1)$$

where the vector $\gamma(t)$ represents a local state variable. The time-varying system matrices are denoted as $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{F}$, respectively. The input vector $\mathbf{u}(t)$ is set to $\mathbf{0}$. The process noise $\mathbf{w}(t)$ is given as a zero-mean Gaussian process (GP) with the kernel function formulated with the power spectral density matrix $\mathbf{Q}_c \in \mathbb{R}^{6 \times 6}$ and the Dirac delta function, δ [1].

With the general solution of (3.1), we can formulate a Gaussian process, which consists of the following mean and covariance functions as in Barfoot et al. [135]:

$$\mathbf{x}(t) = \boldsymbol{\mu}(t) = \boldsymbol{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^t \boldsymbol{\Phi}(t, s)\mathbf{u}(s) \, ds, \quad (3.2)$$

$$\begin{aligned}\mathcal{K}(t, t') &= \boldsymbol{\Phi}(t, t_0)\mathcal{K}_0\boldsymbol{\Phi}(t', t_0)^T + \\ &+ \int_{t_0}^{\min(t, t')} \boldsymbol{\Phi}(t, s)\mathbf{F}(s)\mathbf{Q}_c\mathbf{F}(s)^T\boldsymbol{\Phi}(t', s)^T \, ds, \end{aligned} \quad (3.3)$$

where the initial states and covariance are denoted as $\mathbf{x}_0$ and $\mathcal{K}_0$, respectively. This Gaussian kernel function $\mathcal{K}(t, t')$ can be further interpreted between two successive timestamps t_i and t_j in discrete time as

$$\mathcal{K}(t_i, t_j) = \begin{cases} \boldsymbol{\Phi}(t_i, t_j) \left(\sum_{n=0}^j \boldsymbol{\Phi}(t_j, t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_j, t_n)^T \right) & t_j < t_i, \\ \sum_{n=0}^i \boldsymbol{\Phi}(t_i, t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_i, t_n)^T & t_j = t_i, \\ \left(\sum_{n=0}^i \boldsymbol{\Phi}(t_i, t_n)\mathbf{Q}_n\boldsymbol{\Phi}(t_i, t_n)^T \right) \boldsymbol{\Phi}(t_j, t_i)^T & t_j > t_i. \end{cases} \quad (3.4)$$

By selecting different timestamps, this Gaussian process can be used for interpolation (i.e., $t_j > t_i$) and extrapolation (i.e., $t_i > t_j$) of vehicle states, utilizing the scaling (whitening) function $\mathbf{Q}_i$, which is computed as

$$\mathbf{Q}_i = \int_{t_{i-1}}^{t_i} \boldsymbol{\Phi}(t_i, s)\mathbf{F}\mathbf{Q}_c\mathbf{F}(s)^T\boldsymbol{\Phi}(t_i, s)^T \, ds, \quad (3.5)$$

where the transition matrix is denoted as $\Phi(\cdot)$ for clarity, instead of $A_d(\cdot)$ as in (A.3). The matrix F represents the same shaping matrix as defined in (A.3). The matrix Q_c serves as the effective scaling parameter in the Gaussian Process (GP)-based continuous-time trajectory representation. It must be appropriately tuned for different applications to accurately capture the real vehicle dynamics under uncertainties². In this thesis, this matrix is manually tuned as

$$Q_c = \begin{bmatrix} q_{\text{roll}} & & & & & \\ & q_{\text{pitch}} & & & & \\ & & q_{\text{yaw}} & & & \\ & & & q_x & & \\ & & & & q_y & \\ & & & & & q_z \end{bmatrix}, \quad (3.6)$$

where the independent scaling parameters of each dimension are denoted as q_{state} .

3.3.2 The Gaussian Process Motion Prior in $SE(3)$

In discrete time following Anderson and Barfoot [148], this state-space model of (3.1) can be furthermore interpreted to interpolate an arbitrary state $\gamma(t_{i+\tau})$ at timestamp $t_{i+\tau} = t_i + \tau$ between two successive local states $\gamma(t_i)$ and $\gamma(t_j)$, where the state timestamps $t_i < t_{i+\tau} < t_j$, using

$$\gamma(t_{i+\tau}) = \Lambda(t_{i+\tau})\gamma(t_i) + \Omega(t_{i+\tau})\gamma(t_j), \quad (3.7)$$

where

$$\Lambda(t_{i+\tau}) = \Phi(\tau) - \Omega(t_{i+\tau})\Phi(t_j - t_{i+\tau}), \quad (3.8)$$

$$\Omega(t_{i+\tau}) = Q(\tau)\Phi(t_j - t_{i+\tau})^T Q^{-1}(\tau). \quad (3.9)$$

The system transition matrix Φ in (3.8) and (3.9) can be defined using different motion matters, which are introduced below. This interpolation provides the mathematical foundation for formulating the Gaussian Process (GP) motion prior, enabling the seamless modeling of continuous-time trajectories.

A GP motion prior in practical applications is often represented on a manifold in $SE(3)$, which can be defined as

$$\begin{aligned} \dot{T}(t) &= \varpi(t)^\wedge T(t), \\ \dot{\varpi}(t) &= w(t), \end{aligned} \quad (3.10)$$

²As discussed in Tang et al. [103], representing a realistic system transition using the GP motion priors requires proper tuning of the power spectral density matrix Q_c . In this work, I assume that Q_c is a constant diagonal matrix defined as $Q_c = \text{diag}(q_c)$ with a 6-D hyperparameter q_c . The q_c was manually tuned for the experiments.

where the vehicle pose in the global frame is denoted as $\mathbf{T}(t)$, which can be calculated as $\mathbf{T}(t) = \exp(\xi(t)^\wedge)$ with local pose $\xi(t) = [\rho(t)^T \phi(t)^T]^T \in \mathbb{R}^6$. The vectors $\rho(t)$ and $\phi(t)$ represent the position and orientation of a local pose (e.g., in the body frame) [149].

A local pose can be converted to $\mathfrak{se}(3)$ by applying the operator $(\cdot)^\wedge$. The operator $(\cdot)^\vee$ is the inverse of $(\cdot)^\wedge$ [1]. The vector $\varpi(t) = [\nu(t)^T \omega(t)^T]^T \in \mathbb{R}^6$ represents the body-centric velocity.

However, the GP motion prior in (3.10) cannot be implemented directly using (3.1) due to the nonlinearity of the system dynamics. To address this problem, Anderson and Barfoot [148] showed that a local linear GP prior can be defined between each state-timestamp pair, t_i and t_{i+1} , by transforming the global pose $\mathbf{T}(t)$ into the local tangent frame, where a local pose $\xi_i(t)$ can be calculated as

$$\xi_i(t) = \ln(\mathbf{T}(t)\mathbf{T}_{t_i}^{-1})^\vee, \quad t_i \leq t \leq t_{i+1}, \quad (3.11)$$

where we consider the pose $\mathbf{T}_{t_i}$ at the timestamp t_i as a fixed parameter while formulating the local pose $\xi_i(t)$ for an arbitrary pose $\mathbf{T}(t)$ for $t > t_i$.

Because the motion between state-timestamp pairs, which are usually associated with high-frequency measurement timestamps (e.g., lidar at 10 Hz), is generally small, this local GP prior approximately represents a linear time-invariant (LTI) SDE, which can be driven from (3.1) by assuming the system matrices remain constant.

Compared to other approaches (e.g., spline-based continuous-time interpolation), trajectory representation (interpolation) using Gaussian process regression effectively incorporates physics-driven models to retrieve realistic vehicle motion by scaling the transition function with the time-varying covariance matrix $\mathbf{Q}$. As the hyperparameter $\mathbf{Q}_c$ can be tuned for different applications [150], this approach can be extended for nonlinear problems (see Section 3.3.4) and enables more accurate state interpolation [M1, 138].

3.3.3 GP-WNOA Motion Prior

Following the GP motion prior introduced above, Barfoot et al. [135] initially proposed the white-noise-on-acceleration (WNOA) motion prior, which assumes that the vehicle moves with approximately constant-velocity dynamics. The system transition matrix $\Phi(\Delta t)$, the control input $\mathbf{u}(t)$ and the noise propagation matrix $\mathbf{F}(t)$ are formulated as:

$$\Phi(\Delta t) = \begin{bmatrix} \mathbf{1} & \Delta t \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \mathbf{u}(t) = \mathbf{0}, \quad \mathbf{F}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \quad (3.12)$$

The covariance matrix $\mathbf{Q}_i$ of the GP-WNOA model is given as:

$$\mathbf{Q}_i = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 \mathbf{Q}_c & \frac{1}{2}\Delta t_i^2 \mathbf{Q}_c \\ \frac{1}{2}\Delta t_i^2 \mathbf{Q}_c & \Delta t_i \mathbf{Q}_c \end{bmatrix}, \quad \text{and} \quad (3.13)$$

$$\mathbf{Q}_i^{-1} = \begin{bmatrix} 12\Delta t_i^{-3} \mathbf{Q}_c^{-1} & -6\Delta t_i^{-2} \mathbf{Q}_c^{-1} \\ -6\Delta t_i^{-2} \mathbf{Q}_c^{-1} & 4\Delta t_i^{-1} \mathbf{Q}_c^{-1} \end{bmatrix}. \quad (3.14)$$

In this model, the state of the GP motion model in a global frame is given as

$$\mathbf{x}(t) = \{\mathbf{T}(t) \boldsymbol{\varpi}(t)\} \in SE(3) \times \mathbb{R}^6, \quad (3.15)$$

and its local state variable is given as

$$\boldsymbol{\gamma}(t) = [\boldsymbol{\xi}(t)^T \dot{\boldsymbol{\xi}}(t)^T]^T, \quad (3.16)$$

and propagated using (3.7) to (3.9). The time derivatives of the local pose can be calculated as

$$\dot{\boldsymbol{\xi}}(t) = \mathcal{J}(\boldsymbol{\xi}_i(t))^{-1} \boldsymbol{\varpi}(t), \quad (3.17)$$

where the matrix $\mathcal{J}$ is the *left Jacobian* of $SE(3)$ [1]. To calculate $\frac{d(\mathcal{J}^{-1})}{dt}$ in closed form for (3.29), we approximately formulate $\mathcal{J}^{-1} \approx \mathbf{1} - \frac{1}{2}\boldsymbol{\xi}^\wedge$. Those computations are presented in the Appendix B.2.

Taking the local state variable of the global state $\mathbf{T}_i$ at the timestamp t_i as the reference, we can project the following local state variable at timestamp t_{i+1} given its global state $\mathbf{T}_{i+1}$ as

$$\boldsymbol{\gamma}_i(t_i) = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\varpi}_i(t) \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\gamma}_i(t_{i+1}) = \begin{bmatrix} \ln(\mathbf{T}_{i+1,i})^\vee \\ \mathcal{J}_{i+1}^{-1} \boldsymbol{\varpi}_{i+1} \end{bmatrix}, \quad (3.18)$$

where the transformation between two successive states $\mathbf{T}_{i+1,i}$ is calculated as projecting the state at timestamp $i+1$ back to i as $\mathbf{T}_{i+1,i} = \mathbf{T}_{i+1} \mathbf{T}_i^{-1}$.

3.3.3.1 GP-WNOA Interpolation

Using the GP-WNOA prior, a state at an arbitrary time $\tau \in (i, i+1)$ can be queried as

$$\mathbf{T}_\tau^{\text{wnoa}} = \exp \left\{ \left[\boldsymbol{\Lambda}_{12}(\tau) \boldsymbol{\varpi}_i + \boldsymbol{\Omega}_{11}(\tau) \ln(\mathbf{T}_{i+1,i})^\vee + \boldsymbol{\Omega}_{12}(\tau) \mathcal{J}_{i+1}^{-1} \boldsymbol{\varpi}_{i+1} \right]^\wedge \right\} \mathbf{T}_i, \quad (3.19)$$

where $\boldsymbol{\Lambda}$ and $\boldsymbol{\Omega}$ are vehicle transition matrices obtained from (3.8) and (3.9). For high-dimensional transition matrices in GP motion models (e.g., $\boldsymbol{\Lambda}(\tau) \in \mathbb{R}^{12 \times 12}$), I denote the subblocks $\boldsymbol{\Lambda}_{mn} \in \mathbb{R}^{6 \times 6}$ associated with different state components for linear state querying.

3.3.3.2 GP-WNOA Between-State Factor

Using (3.18), we can formulate a between-state constraint that serves as a motion prior factor in the factor graph optimization, with the error term as

$$\|e_{i,i+1}^{\text{wnoa}}\|^2 = \left\| \begin{bmatrix} \mathbf{r}_{\Delta\gamma_{i,i+1}}^T & \mathbf{r}_{\Delta\varpi_{i,i+1}}^T \end{bmatrix}^T \right\|_{\Sigma^{\text{wnoa}}}^2, \quad (3.20)$$

where

$$\mathbf{r}_{\Delta\gamma_{i,i+1}} = \ln(\mathbf{T}_{i+1,i})^\vee - (t_{i+1} - t_i)\varpi_i, \quad (3.21)$$

$$\mathbf{r}_{\Delta\varpi_{i,i+1}} = \mathcal{J}_{i+1,i}^{-1}\varpi_{i+1} - \varpi_i. \quad (3.22)$$

3.3.4 GP-WNOJ Motion Prior

Later, Tang et al. [103] introduced a white-noise-on-jerk (WNOJ) prior that presents third-order system dynamics with the system transition function

$$\Phi(\Delta t) = \begin{bmatrix} \mathbf{1} & \Delta t \mathbf{1} & \frac{1}{2} \Delta t^2 \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \Delta t \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}. \quad (3.23)$$

The time-varying covariance matrix $\mathbf{Q}(\Delta t) \in \mathbb{R}^{18 \times 18}$ and its precision matrix $\mathbf{Q}^{-1}(\Delta t)$ are computed as

$$\mathbf{Q}(\Delta t) = \begin{bmatrix} \frac{1}{20} \Delta t^5 \mathbf{Q}_c & \frac{1}{8} \Delta t^4 \mathbf{Q}_c & \frac{1}{6} \Delta t^3 \mathbf{Q}_c \\ \frac{1}{8} \Delta t^4 \mathbf{Q}_c & \frac{1}{3} \Delta t^3 \mathbf{Q}_c & \frac{1}{2} \Delta t^2 \mathbf{Q}_c \\ \frac{1}{6} \Delta t^3 \mathbf{Q}_c & \frac{1}{2} \Delta t^2 \mathbf{Q}_c & \Delta t \mathbf{Q}_c \end{bmatrix}, \quad (3.24)$$

$$\mathbf{Q}^{-1}(\Delta t) = \begin{bmatrix} 720 \Delta t^{-5} \mathbf{Q}_c^{-1} & -360 \Delta t^{-4} \mathbf{Q}_c^{-1} & 60 \Delta t^{-3} \mathbf{Q}_c^{-1} \\ -360 \Delta t^{-4} \mathbf{Q}_c^{-1} & 192 \Delta t^{-3} \mathbf{Q}_c^{-1} & -36 \Delta t^{-2} \mathbf{Q}_c^{-1} \\ 60 \Delta t^{-3} \mathbf{Q}_c^{-1} & -36 \Delta t^{-2} \mathbf{Q}_c^{-1} & 9 \Delta t^{-1} \mathbf{Q}_c^{-1} \end{bmatrix}. \quad (3.25)$$

The state of the GP-WNOJ model in a global frame is presented as

$$\mathbf{x}(t) = \{\mathbf{T}(t) \ \varpi(t) \ \dot{\varpi}(t)\} \in SE(3) \times \mathbb{R}^{12}. \quad (3.26)$$

And its local state variable of GP-WNOJ prior in $SE(3)$ can be defined as

$$\gamma(t) = [\xi(t)^T \ \dot{\xi}(t)^T \ \ddot{\xi}(t)^T]^T, \quad (3.27)$$

and propagated using (3.7) to (3.9). The time derivatives of the local pose can be calculated as

$$\dot{\xi}(t) = \mathcal{J}(\xi_i(t))^{-1} \dot{\varpi}(t), \quad (3.28)$$

$$\ddot{\xi}(t) = -\frac{1}{2}(\mathcal{J}(\xi(t))^{-1} \dot{\varpi}(t))^\wedge \dot{\varpi}(t) + \mathcal{J}(\xi(t))^{-1} \ddot{\varpi}(t). \quad (3.29)$$

The operator $(\xi)^\wedge$ represents the adjoint of $\xi^\wedge \in \mathfrak{se}(3)$ [1], which can be calculated as

$$\xi^\wedge = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho^\wedge \\ 0 & \phi^\wedge \end{bmatrix}. \quad (3.30)$$

Because the left Jacobian requires several matrix calculations, it can be approximated as an identity matrix $\mathbf{1}$ over small intervals to improve the computation efficiency³ [151].

3.3.4.1 GP-WNOJ Interpolation

Using the GP-WNOJ prior, a state at an arbitrary time $\tau \in (i, i+1)$ can be queried as

$$\begin{aligned} \mathbf{T}_\tau^{\text{wnoj}} = \exp \bigg\{ & \left[\mathbf{\Lambda}_{12}(\tau) \varpi_i + \mathbf{\Omega}_{13}(\tau) \dot{\varpi}_i + \mathbf{\Sigma}_{11}(\tau) \ln(\mathbf{T}_{i+1,i})^\vee + \right. \\ & + \mathbf{\Omega}_{12}(\tau) \mathcal{J}_{i+1}^{-1} \varpi_{i+1} + \\ & \left. + \mathbf{\Omega}_{13}(\tau) \left(-\frac{1}{2} (\mathcal{J}_{i+1}^{-1} \varpi_{i+1})^\wedge \varpi_{i+1} + \mathcal{J}_{i+1}^{-1} \dot{\varpi}_{i+1} \right) \right]^\wedge \bigg\} \mathbf{T}_i, \end{aligned} \quad (3.31)$$

where $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ are vehicle transition matrices obtained from (3.7) to (3.9).

3.3.4.2 GP-WNOJ Between-State Factor

Given a local state variable that represents the origin system state for each state-timestamp pair, we can retrieve the GP-WNOJ motion model for two successive local state variables in the local frame as

$$\begin{aligned} \gamma_i(t_i) &= [\mathbf{0} \ \varpi(t)^T \ \dot{\varpi}(t)^T]^T, \\ \gamma_i(t_{i+1}) &= \begin{bmatrix} \ln(\mathbf{T}_{i+1,i})^\vee \\ \mathcal{J}_{i+1}^{-1} \varpi_{i+1} \\ -\frac{1}{2} (\mathcal{J}_{i+1}^{-1} \varpi_{i+1})^\wedge \varpi_{i+1} + \mathcal{J}_{i+1}^{-1} \dot{\varpi}_{i+1} \end{bmatrix}. \end{aligned} \quad (3.32)$$

³I implement this trick as a configuration in the proposed gnssFGO.

I implement the GP-WNOJ motion model as between-state factors, similar to Yan et al. [137]. The error function was originally given in Tang et al. [103] using (3.32). We can summarize this error function for convenience as

$$\|e_{i,i+1}^{\text{wnoj}}\|^2 = \left\| [r_{\Delta\gamma_{i,i+1}}^T \ r_{\Delta\varpi_{i,i+1}}^T]^T \right\|_{\Sigma^{\text{wnoj}}}^2, \quad (3.33)$$

where

$$r_{\Delta\gamma_{i,i+1}} = \ln(\mathbf{T}_{i+1,i})^\vee - (t_{i+1} - t_i)\varpi_i - \frac{1}{2}(t_{i+1} - t_i)^2\dot{\varpi}_i, \quad (3.34)$$

$$r_{\Delta\varpi_{i,i+1}} = \mathcal{J}_{i+1,i}^{-1}\varpi_{i+1} - \varpi_i - (t_{i+1} - t_i)\dot{\varpi}_i. \quad (3.35)$$

In this thesis, I used the measured accelerations of the IMU in the GP motion models⁴. Thus, only the 6-D pose and the 6-D velocity are evaluated in GP-WNOJ motion factors, so that $e_{i,i+1}^{\text{wnoj}} \in \mathbb{R}^{12}$. The analytical Jacobians of the GP motion models are presented in Appendix B.2.

3.4 Time-Centric Factor Graph Optimization

In this section, I finally present my proposed multisensor state estimator by leveraging the GP-based trajectory representation introduced in the last section (Section 3.3). The core algorithm of this state estimator can be interpreted as a *time-centric factor graph optimization*, in which the state variables are deterministically instantiated in the graph and connected to all associated nodes (i.e., unary factors or landmark states).

To the best of my knowledge, existing work has not explored continuous-time factor graph optimization for online fusion of GNSS with multiple heterogeneous sensor measurements compared to the original implementation.

Generally, fusing multiple heterogeneous sensor observations into a state estimator incorporates different timestamps due to asynchronous measurements and unpredictable delays. In this work, I assume that the state estimator and all measurements have the same clock. Compared to the estimated states in continuous time, all sensor observations are sampled and processed in asynchronous timestamps, as illustrated in Figure 3.1. Here, I use the variable $\tau = t_{\text{meas.}} - t_{x_i}$ to define the time offset between a non-delayed measurement and the last state variable x_i prior to this measurement. If a measurement is delayed with the timestamp $\check{t}_{\text{meas.}}$, I use the given time delay t_d to calculate the non-delayed measurement timestamp $t_{\text{meas.}} = \check{t}_{\text{meas.}} - t_d$.

⁴Unlike [103], I do not estimate 6-D accelerations in GP motion models to reduce the dimension of the state vector. Instead, I consider the vehicle accelerations measured by the IMU as inputs to the WNOJ model.

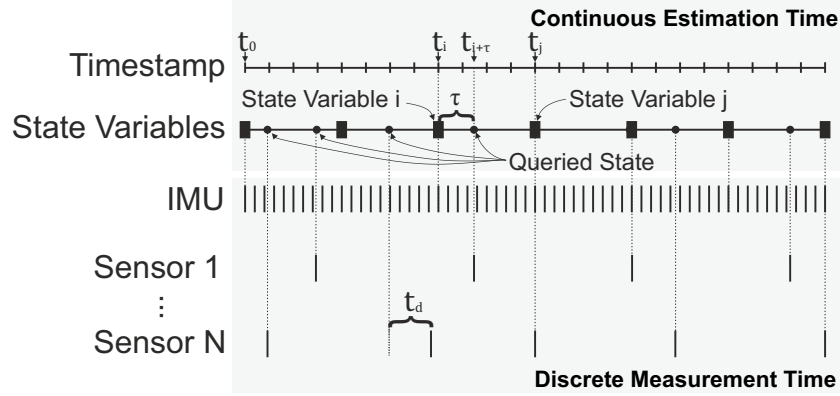


Figure 3.1: Continuous-time state estimation with asynchronous measurements. A time offset τ can be calculated with respect to a former state variable x_i at timestamp t_i for each asynchronous measurement. The variable t_d denotes a measurement delay that is assumed to be given (reproduced from [M2] with permission of IEEE).

I employ GP motion priors that enable a continuous-time trajectory representation. In this way, constructing a factor graph can be deterministic and time-centric, bypassing asynchronous sensor frequencies and timing issues. I show the general structure of a time-centric factor graph in Figure 3.2, where the to-be-estimated state variables x_t are presented in solid line circles on a continuous-time trajectory. Queried states in dashed line circles are not to-be-estimated state variables and, therefore, are only queried between two successive state variables using the time offset τ between the sensor timestamp and the previous state variable.

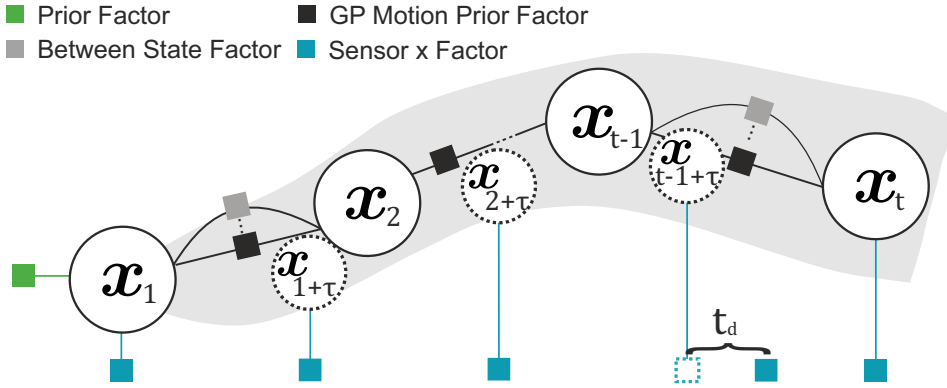


Figure 3.2: A general time-centric factor graph. The state variables x_t are created and constrained with GP motion prior factors on time, while all asynchronous measurements are fused by querying a state with a time offset τ between the measurement and the former state variable. The queried states (in dashed circles) are thus not to-be-estimated state variables. I assume that the measurement delay t_d is known to correct the measurement timestamp for querying a state (reproduced from [M2] with permission of IEEE).

This procedure yields a state estimation algorithm, as presented in Algorithm 1, which explains one optimization procedure from graph construction to iterative optimization.

Assuming that the time-centric factor graph is extended with n new to-be-estimated state variables in each procedure. I extend the graph with n state variables and create GP motion prior factors that constrain the relative state transitions between two successive state variables. By doing so, the timestamps of all state variables are chosen deterministically. While solving the iterative optimization problem, an initial prediction $\mathbf{x}_k^- \in \mathcal{X}^-$ must be provided for each state variable. These predictions can be acquired using prior motion models (e.g., GP state extrapolation [152]). In this work, I utilize state propagation using IMU measurements to calculate the initial estimate of future states at high frequency.

As new sensor observations are received at different timestamps in parallel to estimation times, I retrieve the cached m observations from each sensor $s \in \mathcal{S}$ in a second loop. I define a time threshold t_{sync} for state-observation alignment to query the index of related state variables. If state variables can be associated with sensor observations within this threshold, normal sensor factors are added to the graph. Otherwise, I construct the measurement factors by querying a GP interpolated state aligned with the measurement timestamp. In this case, two successive state variables $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are obtained with a time offset τ between the measurement and the former state $\mathbf{x}_i$.

After graph construction, I employ a Gauss-Newton-like optimizer to solve the MAP problem [153]. The optimized state $\mathbf{x}_k^+$ and marginalized uncertainties $\mathbf{P}_k^+$ are returned for further state propagation, as introduced in Section A.2.7.

Algorithm 1: Time-centric factor graph optimization (reproduced from [M2] with permission of IEEE).

Input : Last state id and timestamp pair $(x_{\text{id}}^-, x_{\text{ts}}^-)$
Initial state $(\mathbf{x}_0^-, \mathbf{P}_0^-)$ or Propagated states $\mathbf{x}_k^- \in \mathcal{X}^-$, $k = 1 \dots n$
List of sensor measurements $s \in \mathcal{S}$

Output: Current state id and timestamp pair $(x_{\text{id}}^+, x_{\text{ts}}^+)$
Optimized state $\mathbf{x}_k^+$ and uncertainties $\mathbf{P}_k^+$

```

1  $\mathcal{G} \leftarrow \text{initGraph}(\mathbf{x}_0^-, \mathbf{P}_0^-)$ ;
2 List of state id and timestamp pairs  $\mathcal{P} = \emptyset$ ;
3  $x_{\text{id}} = x_{\text{id}}^-$ ;
4 for  $k = 1 : n$  do
5    $x_{\text{id}}^+ = x_{\text{id}} + 1$ ;
6    $x_{\text{ts}}^+ = \text{updateTimestamp}(x_{\text{ts}}^-)$ ;
7    $\mathcal{G} \leftarrow \text{NewStateVariable}(x_{\text{id}}, x_{\text{ts}}, \mathcal{X})$ ;
8    $\mathcal{G} \leftarrow \text{GPMotionFactor}(x_{\text{id}-1}, x_{\text{id}})$ ;
9    $\mathcal{P} \leftarrow (x_{\text{id}}, x_{\text{ts}})$ ;
10 end for
11 for Each sensor  $s \in \mathcal{S}$  do
12   for Each observation  $k = 1 : m$  do
13      $(x_i^{\text{id}}, x_i^{\text{ts}}, \tau, \text{type}) \leftarrow \text{queryStateInfo}(\text{timestamp}_k^s, \mathcal{P})$ ;
14     if  $\text{type}$  is DROPPED then
15       /* Measurements in the past. */;
16        $\text{discardMeasurement}(\mathbf{o}_k^s)$ ;
17     else if  $\text{type}$  is SYNCHRONIZED then
18        $\mathcal{G} \leftarrow \text{SensorFactor}(x_{\text{id}}, \mathbf{o}_k^s)$ ;
19     else if  $\text{type}$  is INTERPOLATED then
20        $\mathcal{G} \leftarrow \text{GPSensorFactor}(x_i^{\text{id}}, x_{i+1}^{\text{id}}, \tau, \mathbf{o}_k^s)$ ;
21     else if  $\text{type}$  is CACHED then
22       /* Measurements in the future. */  $\text{cacheMeasurement}(\mathbf{o}_k^s)$ ;
23   end for
24 end for
25  $\{\mathbf{x}_k^+, \mathbf{P}_k^+\} \leftarrow \text{doOptimizationAndMarginalization}(\mathcal{G})$ ;
26 return  $\{(x_{\text{id}}^+, x_{\text{ts}}^+), \mathbf{x}_k^+, \mathbf{P}_k^+\}$ 

```

3.5 Application I: GNSS/INS Fusion in Temporarily GNSS-Denied Environments

As introduced in Section 2.3.1, this application focuses on vessel localization in temporarily GNSS-denied environments through GNSS/INS integration. The objective of this application is to comprehensively study the classical Kalman filter and continuous-time factor graph optimization for fusing GNSS and IMU measurements under temporary GNSS outages. Thus, this study establishes a fundamental understanding of the limitations of filtering-based methods, as well as the differences arising from the use of various motion priors and error models in graph optimization, which is necessary to address the research question **Q1**.

In this study, I adopted the unscented Kalman filter⁵ from prior work [157, 158], which has served as a strong baseline for fusing GNSS and IMU measurements in maritime environments. I explored both classic and continuous-time factor graph optimization (FGO), which implements the proposed Algorithm 1 in an offline process, to evaluate their performance and limitations, with and without incorporating various between-state constraints (i.e., motion priors).

To the best of my knowledge, the use of FGO for GNSS/INS integration using continuous-time trajectory representation has not been thoroughly investigated in the research community. Furthermore, recent studies rarely address solver settings and runtime efficiency for FGO in large-scale outdoor applications. Thus, this application focuses on tightly coupled FGO for fusing inertial measurements with GNSS pseudorange and Doppler velocity observations in the context of inland shipping.

3.5.1 UKF for Tightly Coupled GNSS/INS Fusion

I followed the previous work by Gehrt et al. [159] to implement an unscented Kalman filter (UKF), as shown in Figure 3.3.

In the prediction step, the raw Inertial Measurement Unit (IMU) data were fed into the strap-down algorithm described in Algorithm 8 in Section A.3.4.1 to propagate the state and uncertainties. The raw GNSS measurements from both the main and auxiliary antennas underwent pre-processing to eliminate delays caused by satellite clock errors and atmospheric

⁵Although advanced and novel filtering techniques for sensor fusion exist—such as the iterative Kalman filter [154], invariant Kalman filter [155], and equivalent filter [156]—this study intentionally adopts the classical unscented Kalman filter for two reasons. First, I employed a classical Kalman filtering framework to develop a deep understanding and to enable a fair and transparent comparison between Kalman filtering and graph optimization. Second, advanced filtering techniques remain out of scope for this thesis, as they still suffer from the same fundamental limitation imposed by the Markov property, as discussed in Appendix A.2.9. Consequently, even novel filtering-based methods cannot meet my requirements for handling GNSS outliers throughout this thesis.

3 Continuous-Time Factor Graph Optimization

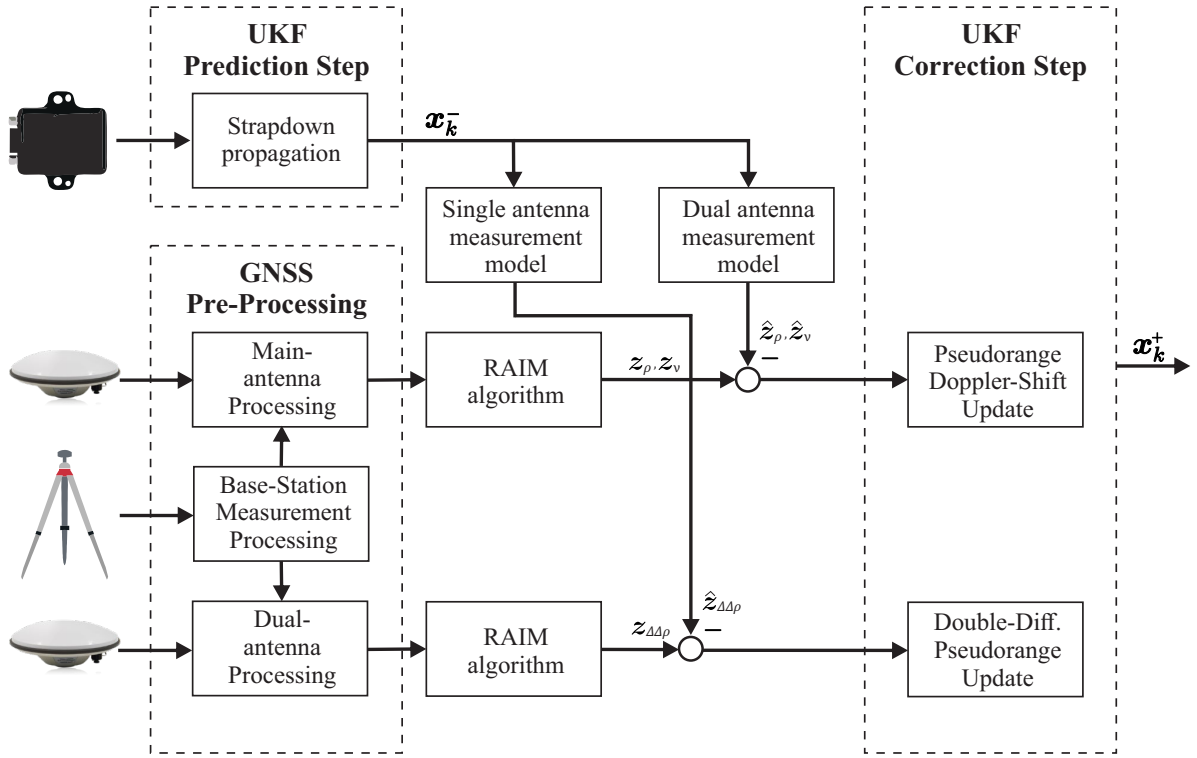


Figure 3.3: Flowchart of the UKF-based GNSS/INS fusion following Gehrt et al. [159].

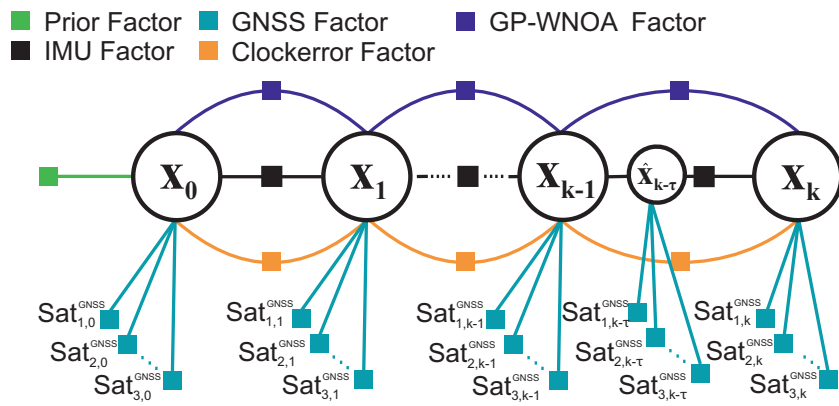


Figure 3.4: Factor graph proposed in Application I (reproduced from [M1] with permission of IEEE).

influences. I utilized differential correction data from a base station provided by a local positioning service (SAPOS) to calculate ionospheric and tropospheric delays. Additionally, the pre-processed GNSS observations were processed using the Receiver Autonomous Integrity Monitoring (RAIM) algorithm to identify and exclude faulty GNSS observations from the measurement updates. In the correction step, I computed two Kalman innovations based on both pseudorange/Doppler shift and double-differenced pseudorange observations.

The UKF generated state estimate at a frequency of 100 Hz, while the measurement updates were incorporated at 10 Hz. For further implementation details, refer to Appendix C.

3.5.2 FGO for Tightly Coupled GNSS/INS Fusion

Following Algorithm 1 in an offline process, the factor graph implemented in this application is shown in Figure 3.10. Once a new state variable is created, all relative factors, such as IMU factor, motion prior factor, and receiver clock error factor, are constructed between the new state and the last successive state. If the GNSS observation, which is expected at a fixed rate, is delayed, a GP-interpolated GNSS factor is added to the interpolated state with the delayed time τ . For GP related factors, I took $\mathbf{Q}_c$ as a diagonal matrix with parameter Q_c .

The MAP formulated based on the factor graph is defined in (3.36), where each error term is defined in Appendix B.3. I used the Levenberg–Marquardt optimizer to solve this MAP problem.

$$\hat{\mathbf{X}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left[\sum_{i=1}^k \sum_{s=1}^n \hat{e}_{s,i}^{\text{PrDo}} + \sum_{i=1}^k \|\mathbf{e}_i^I\|_{\Sigma_I}^2 + \sum_{i=1}^k \|\mathbf{e}_i^C\|_{\Sigma_c}^2 + \sum_{i=1}^k \|\mathbf{e}_i^{GP}\|_{\Sigma_K}^2 \right]. \quad (3.36)$$

3.5.3 System Overview

Figure 3.5 shows a system overview of the implemented state estimator with the proposed FGO. I acquired IMU measurements at 100 Hz and deterministically extended the state variables with related IMU factors at 10 Hz on the graph. The GNSS observations were expected to be integrated at a rate of 10 Hz.

I utilized the PPS signal from the GNSS receiver to calculate the time delay of the pre-processed GNSS observations and construct either the GP-interpolated GNSS factor on time delay or the normal GNSS factor without time delay on the graph. The RTCMv3 correction data for the GNSS pre-processing were received from the SAPOS server via the internet at about 1 Hz.

After each optimization, the estimated IMU biases and recalibrated gravity were fed back to the IMU pre-integrator. To provide the navigation solution at a high frequency, I propagated

3 Continuous-Time Factor Graph Optimization

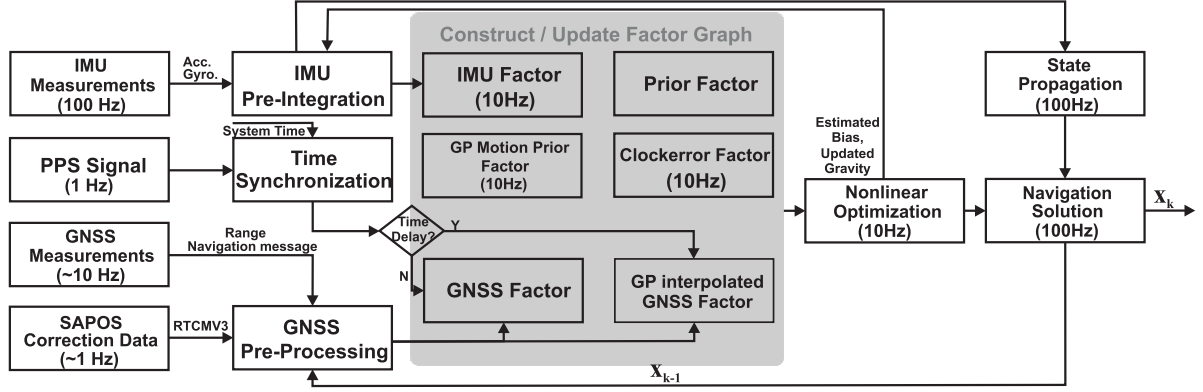


Figure 3.5: Flowchart of the system for tightly coupled GNSS/INS integration with continuous-time FGO in Application I (reproduced from [M1] with permission of IEEE).

the state between two optimization processes on each new IMU measurement, achieving the state estimation at 100 Hz.

3.5.4 Essential Experimental Results

This section presents the essential experimental results of Application I. For implementation details and more extensive results, I refer readers to Appendix C.

3.5.4.1 General Error Metrics

The estimation results obtained using the UKF as the baseline and FGO with various configurations are summarized in Table 3.1, where the RTK-fixed PVT solution from the GNSS receiver serves as the ground-truth reference. In addition, a traditional discrete-time FGO without GP-WNOA motion prior factors is evaluated to highlight the performance improvements achieved by the proposed continuous-time FGO.

In general, FGO demonstrates superior localization accuracy compared to UKF, particularly in challenging scenarios such as bridge crossings. However, exceptions are observed in route #2 (open-sky) and route #4 (harbor area), where the UKF performs equally well or slightly better than FGO. I attribute this behavior to the Kalman filter's ability to converge rapidly due to complete marginalization at the previous state, provided that its underlying assumptions hold, as discussed in Section A.2.9. Despite this, FGO consistently produces smoother trajectories.

For the discrete-time (classic) FGO, only IMU pre-integration, GNSS observations, receiver clock error, and constant-velocity motion prior are represented as factors on the graph. In

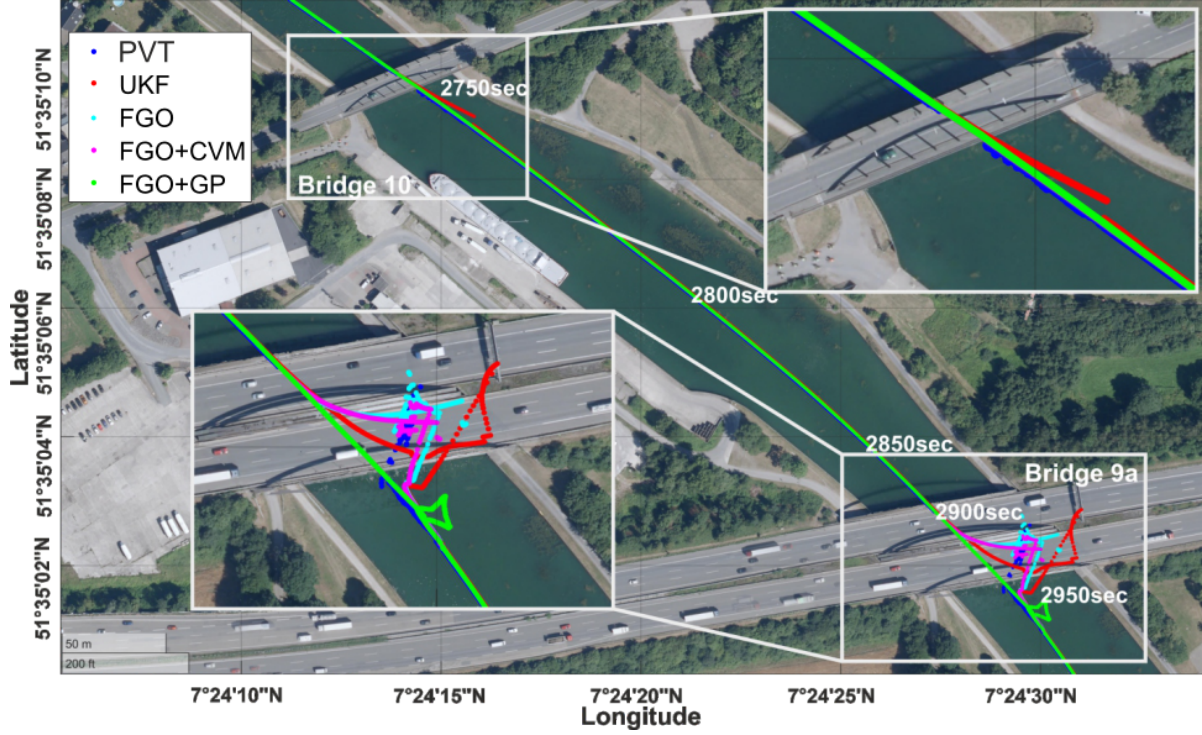


Figure 3.6: Trajectory plot by crossing the bridges 10 and 9a in Dortmund-Ems canal (reproduced from [M1] with permission of IEEE).

contrast, the FGO with continuous-time trajectory representation using the GP-WNOA motion prior demonstrates superior performance, particularly in terms of trajectory smoothness.

However, the position error between these two approaches shows only minor differences. This observation aligns with my expectations, as the primary motivation for investigating the continuous-time trajectory representation is to enable more flexible sensor fusion and to more effectively restrict trajectory drift, as discussed in Section 3.5.4.3.

Figure 3.7 illustrates the 2D position error, estimated velocity, and orientation in the navigation frame. It is evident that the UKF diverges significantly in all estimations during bridge crossings, whereas FGO maintains overall smoother and more reliable estimations. Notably, the 2D position error was not computed during bridge crossings because the ground-truth PVT data was unavailable.

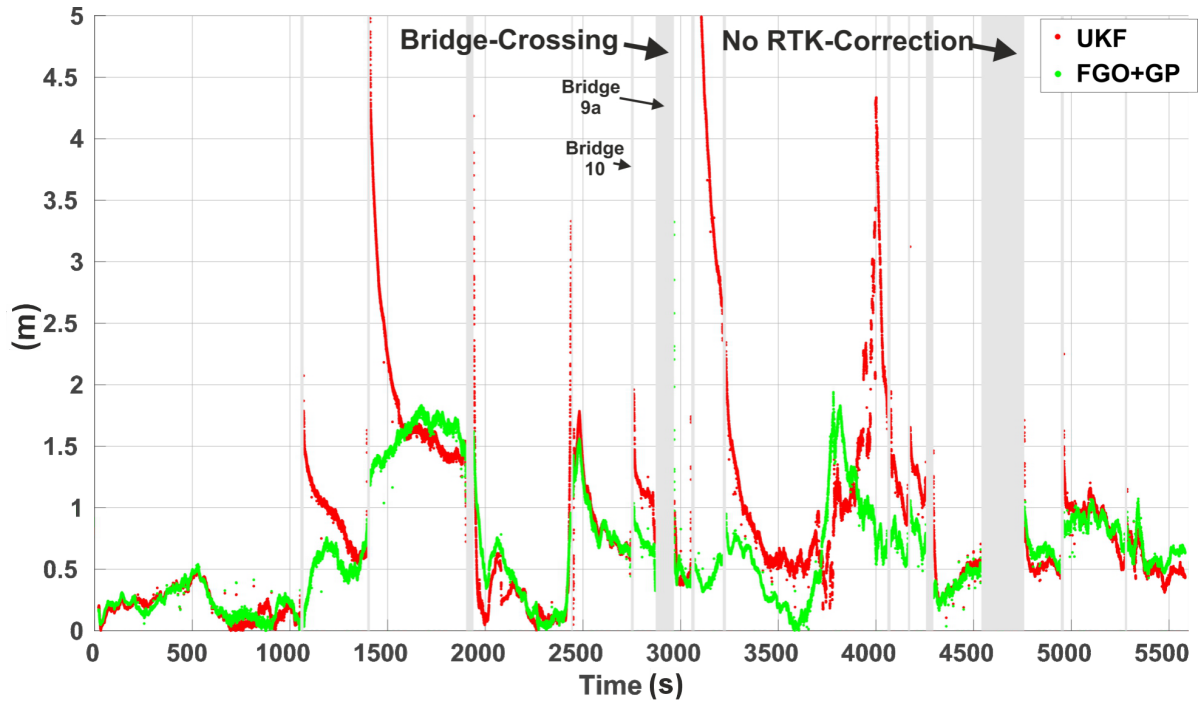
3.5.4.2 Trajectory Smoothness

To gain deeper insight into the robustness, I evaluated the smoothness $\bar{s}$, as introduced in Appendix A.3.7, of the estimated trajectories using the smoothness metrics introduced in Appendix A.3.7.

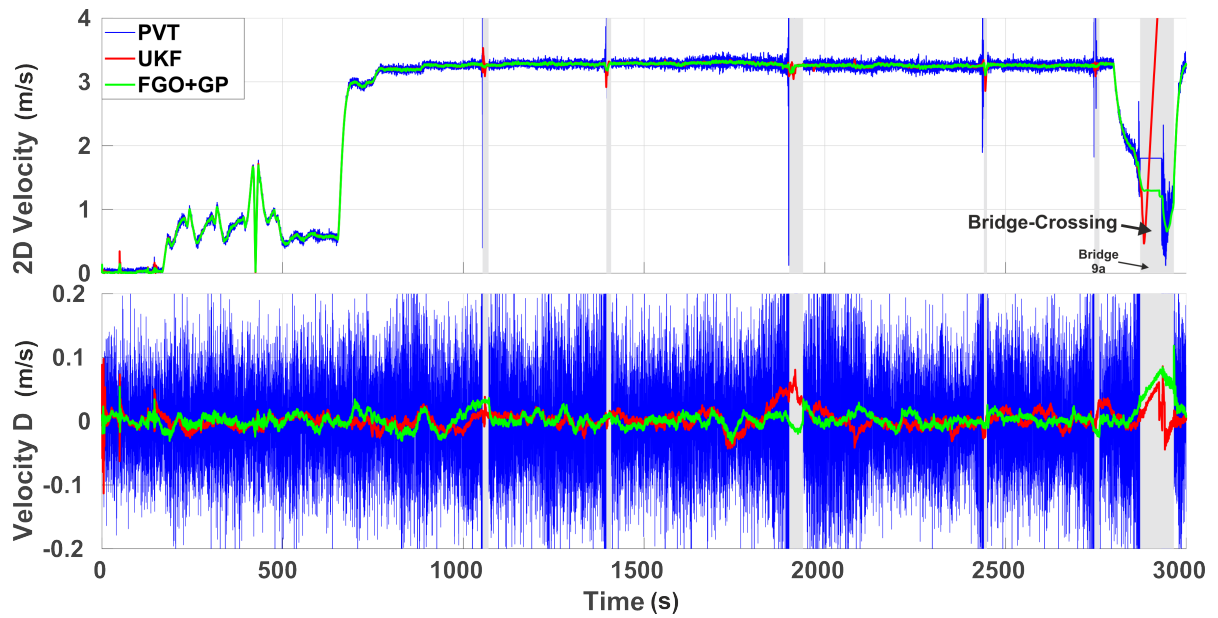
3 Continuous-Time Factor Graph Optimization

Table 3.1: State Estimation Performance of Application I. #1: Total route (0-5583.3s), #2: open-sky without bridge-crossing (300-500s), #3: three bridges crossing (2740-3090s), #4: harbor (5300-5550s) (reproduced from [M1] with permission of IEEE). The variant $\bar{s}$ denotes the smoothness metric introduced in Appendix A.3.7.

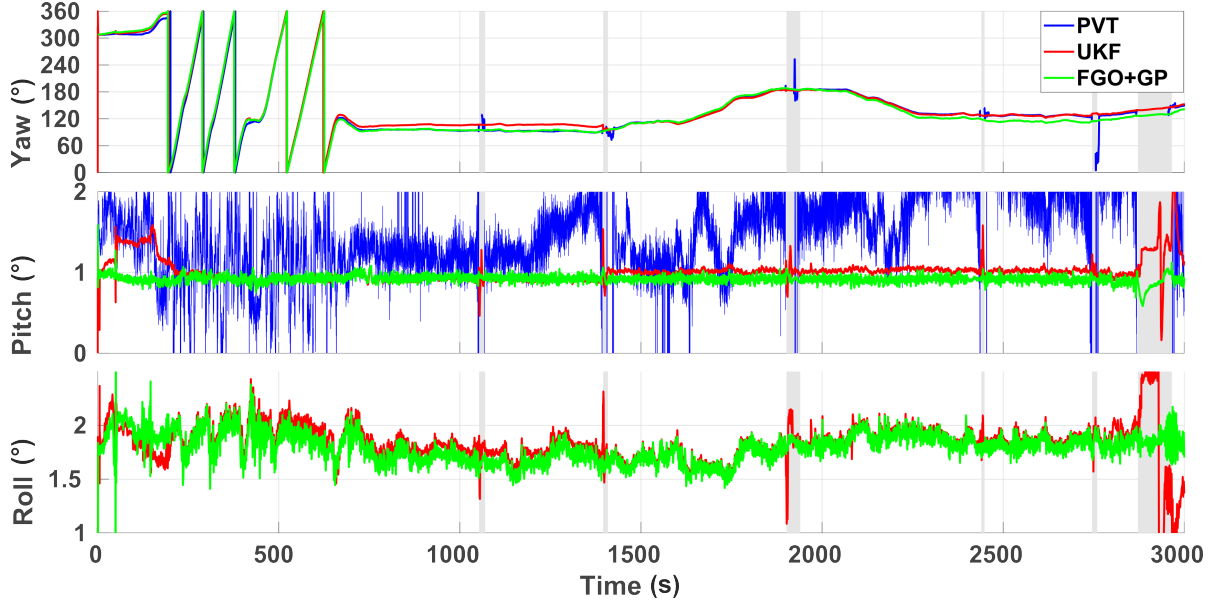
Method & Setting		Sequence	Mean 2D Pos. Error (m)	Mean 3D Pos. Error (m)	Mean 2D Vel. Error (m/s)	$\bar{s}$	
UKF		#1	0.8757	1.9437	0.0420	3179.87	
		#2	0.3333	0.4314	0.0263	59.46	
		#3	1.3230	1.5000	0.0605	-	
		#4	0.5323	1.7595	0.0432	15.24	
Discrete-Time FGO	Batch Lag 0.5 s	#1	0.6532	1.7244	0.0421	744.05	
		#2	0.3200	0.3925	0.0263	0.44	
		#3	0.5564	0.9683	0.0609	-	
		#4	0.6666	1.2940	0.0439	558.55	
	iSAM2	#1	0.6942	1.7408	0.0420	792.26	
		#2	0.4156	0.5025	0.0263	0.39	
		#3	0.5956	1.0443	0.0605	-	
		#4	0.6649	1.3556	0.0437	36.16	
	iSAM2 Lag 0.5 s	#1	0.6422	1.6916	0.0422	969.48	
		#2	0.3333	0.3760	0.0264	0.5423	
		#3	0.5787	1.0315	0.0609	-	
		#4	0.6591	1.2966	0.0439	318.44	
	Continuous-Time FGO	Batch Lag 0.5 s	#1	0.6458	1.7124	0.0421	345.79
			#2	0.3155	0.3902	0.0262	0.40
			#3	0.5680	0.9569	0.0609	-
			#4	0.6200	1.2802	0.0439	2.57
iSAM2		#1	0.6876	1.7245	0.0420	1156.98	
		#2	0.4127	0.4723	0.0262	0.45	
		#3	0.6191	1.0351	0.0606	-	
		#4	0.6692	1.3212	0.0437	437.01	
iSAM2 Lag 0.5 s		#1	0.6370	1.6831	0.0421	267.66	
		#2	0.3249	0.3711	0.0262	0.49	
		#3	0.6073	1.0415	0.0608	-	
		#4	0.6141	1.2976	0.0439	1.01	
iSAM2 Lag 0.5 s w/o. M-estimator		#1	0.8538	2.0091	0.0419	675.52	



(a) 2D Position Error



(b) Estimated Velocity in the NED frame



(c) Estimated Yaw, Pitch, and Roll Angle in the NED frame

Figure 3.7: Estimation Results of Application I (reproduced from [M1] with permission of IEEE).

The results, presented in Table 3.1, show that the trajectories estimated by FGO are generally smoother (lower smoothness metric values) compared to those from the UKF. This higher trajectory smoothness with FGO is also qualitatively evident in Figure 3.6, especially in scenarios with strongly corrupted GNSS observations.

It is important to note that the smoothness for route #3 cannot be accurately calculated in all experiments due to trajectory drifts observed while crossing three bridges, as depicted in Figure 3.6.

3.5.4.3 Impact of Motion Prior Factors

In scenarios with strongly corrupted GNSS observations, we can observe that integrating motion prior factors in FGO can significantly constrain trajectory drift. To evaluate this effect, my experiments assess FGO performance with the constant velocity model (CVM) and the Gaussian Process (GP) motion prior model using different hyperparameters, with no prior factor serving as the baseline for comparison.

Figure 3.8 illustrates the trajectories generated by different motion prior models while crossing bridge 9a, which is presented in Figure 3.6. The GP motion prior outperforms the CVM prior by leveraging the scaled covariance $\mathbf{Q}_i$ to adapt the system transition matrix, as defined in (3.14). However, the hyperparameter q_c must be carefully tuned for different scenarios to ensure optimal performance.

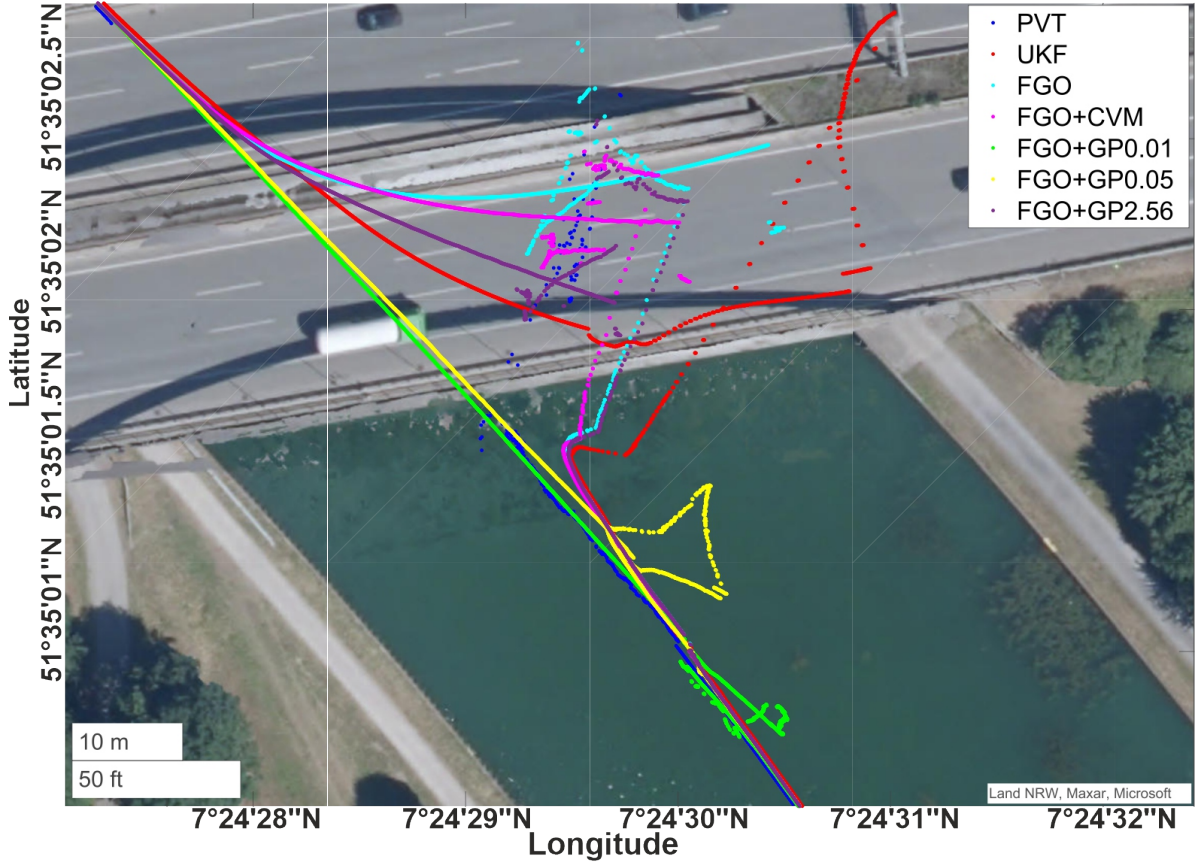


Figure 3.8: FGO with different motion prior models of Application I (reproduced from [M1] with permission of IEEE).

Additionally, since the GP motion prior incorporates the constant velocity model in the transition matrix for state and error propagation, it may be insufficient during periods of acceleration or deceleration. This limitation is evident in Figure 3.6 after the ship crosses bridge 9a, where a slight drift appears in the estimated trajectory of FGO using GP with $q_c = 0.05$. This issue was not observed when crossing other bridges, where the ship maintained a consistent speed.

3.5.4.4 Query the States on Delayed GNSS Observations

As demonstrated in Section 3.5.4.3, the use of GP-WNOA motion prior factors significantly enhances trajectory estimation. Furthermore, the continuous-time trajectory representation with the GP-WNOA motion model offers notable flexibility for sensor integration, enabling state queries at arbitrary times.

Table 3.2 compares the results of normal linear interpolation and GP-WNOA-based state

3 Continuous-Time Factor Graph Optimization

interpolation for delayed GNSS observations. The results show that state interpolation using the GP-WNOA motion prior generally outperforms normal linear interpolation, offering improved accuracy and consistency.

Table 3.2: State Estimation Performance with Linear Interpolated States on GNSS Delay in Application I (reproduced from [M1] with permission of IEEE).

Method	Batch		iSAM2		
	Lag Size 0.5s	Lag Size 10s	w/o. Lag	Lag Size 0.5s	Lag Size 10s
2D Error #1 (m)	0.6528	1.8877	0.6920	0.6400	0.6456
2D Error #2 (m)	0.3199	0.2676	0.4430	0.3339	0.3183
2D Error #3 (m)	0.5561	0.6228	0.6105	0.5680	0.6204

3.5.5 Discussion on Application I

In this application, I used the proposed state estimator for GNSS/INS integration to enhance both localization accuracy and trajectory smoothness in outdoor shipping environments where GNSS observations are temporarily unavailable or heavily corrupted.

My results demonstrated that FGO generally outperformed the UKF in terms of accuracy and trajectory smoothness when using the same measurement data and noise parameters. Although the Fixed-Lag-iSAM2 solver in FGO has a computation time approximately 30 % longer than that of the UKF, this is considered acceptable for online applications on standard computing platforms.

To better understand the impact of different settings and hyperparameters on FGO performance, I conducted experiments varying solver types, fixed-lag sizes, measurement noise levels, and motion prior models. The findings indicated that the Fixed-Lag-iSAM2 solver with a lag size of 0.5s provided the optimal configuration. Furthermore, incorporating GP-WNOA motion priors with a well-tuned hyperparameter q_c significantly improved performance.

3.6 Application II: GNSS/Multisensor Fusion for Robust Vehicle Localization

Building on the outcomes of Application I, this application implemented the proposed Algorithm 1 for online and real-time multisensor fusion in challenging urban environments. Compared to the inland shipping scenario, this application considered a more demanding setting for GNSS observations in highly urbanized areas, where GNSS measurements are frequently corrupted or even unavailable.

Unlike Application I, which focused on a preliminary study of fusing raw GNSS observations with IMU measurements across different estimator types, this application investigates the performance and robustness of a graph-optimization-based state estimator in challenging environments.

To achieve robust trajectory estimation under such conditions, additional sensor modalities, including a lidar sensor and an optical speed sensor, were integrated into the state estimation pipeline. Moreover, the integration of multiple sensor modalities was necessary to study the scalability and flexibility of the proposed state estimator.

3.6.1 Loosely Coupled Factor Graph Optimization

Although the loosely coupled fusion with GNSS and IMU measurements has been shown to be less performant compared to tight coupling [104], I implemented a loosely coupled fusion of sensor observations, including the 2-D speed sensor and the lidar odometry, to i) study the performance gain by fusing multiple sensor observations; ii) evaluate the loosely and tightly coupled fusion for GNSS-based vehicle localization in challenging areas; iii) demonstrate the flexibility and scalability of the proposed method.

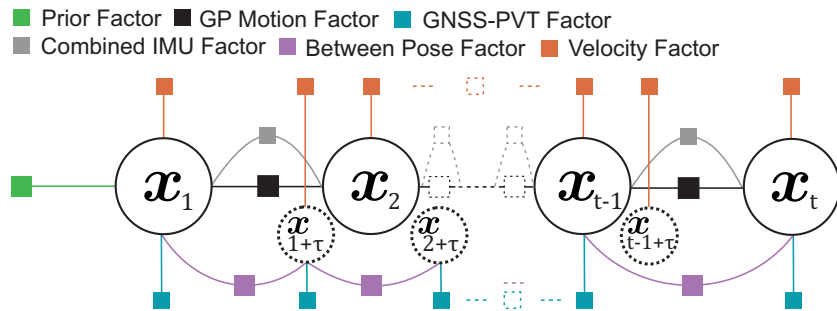


Figure 3.9: A general graph of loose coupling in gnssFGO (reproduced from [M2] with permission of IEEE).

The proposed factor graph is shown in Figure 3.9. The states $x_{1:t}$ are created deterministically on the graph independently of any measurement. If a measurement cannot be

associated with any state variable, a state $\hat{\mathbf{x}}_{i+\tau}$ between two state variables $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_{i+1}$ (where $t_i < \tau < t_{i+1}$) is queried for the error evaluation.

The optimization problem can then be formulated as

$$\begin{aligned} \hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \bigg(& \|e^0\|_{\Sigma_0}^2 + \sum_{i=1}^M \|e_i^{\text{imu}}\|_{\Sigma^{\text{imu}}}^2 + \sum_{i=1}^M \|e_i^{\text{gp}}\|_{\Sigma^{\text{gp}}}^2 + \\ & + \sum_{i=1}^N \|e_i^{\text{vel}}\|_{\Sigma^{\text{vel}}}^2 + \sum_{i=1}^K \|e_i^{\text{pvt}}\|_{\Sigma^{\text{pvt}}}^2 + \sum_{i=1}^J \|e_i^{\text{bp}}\|_{\Sigma^{\text{bp}}}^2 \bigg), \end{aligned} \quad (3.37)$$

where the error term e^0 represents the prior factor obtained at initialization or from marginalization. Because sensor observations are received asynchronously and not necessarily at the estimation timestamps M , I use different index notations N , K , J to denote the number of sensor observations in (3.37). The detailed mathematical foundations of the error terms are provided in Appendix B.3.

3.6.2 Tightly Coupled Factor Graph Optimization

In contrast to the loosely coupled fusion approach, a tightly coupled fusion of raw GNSS observations contributes more constraints with multiple observed satellites to state variables, as illustrated in Figure 3.10.

Unlike the loosely coupled fusion presented in Figure 3.9, I include the pseudorange and Doppler-shift factors in the graph, providing redundant constraints to each state variable. To improve robustness when GNSS observations are degraded or lost in challenging areas, I incorporate lidar odometry as between-state constraints to enhance the consistency of the estimated trajectory. The receiver clock error factor is also added to the graph.

In this fusion mechanism, I do not fuse the measurements from the 2-D velocity, which is not commonly used in vehicle localization approaches, aiming to highlight the robustness of the tightly coupled fusion (see discussion in Section 3.6.4.2 and Section 3.6.4.3).

The optimization problem with sensor observations from different time domains becomes

$$\begin{aligned} \hat{\mathbf{X}} = \operatorname{argmin}_{\mathbf{x}} \bigg(& \|e_i^0\|_{\Sigma_0}^2 + \sum_{i=1}^M \|e_i^{\text{IMU}}\|_{\Sigma^{\text{IMU}}}^2 + \sum_{i=1}^M \|e_i^{\text{gp}}\|_{\Sigma^{\text{gp}}}^2 + \sum_{i=1}^N \|e_i^{\text{bp}}\|_{\Sigma^{\text{bp}}}^2 + \\ & + \sum_{i=1}^M \|e_i^{\text{cbd}}\|_{\Sigma^{\text{cbd}}}^2 + \sum_{i=1}^J \sum_{s=1}^K \|e_{s,i}^{\text{PrDo}}\|_{\Sigma^{\text{PrDo}}}^2 \bigg), \end{aligned} \quad (3.38)$$

where all error terms are defined in Appendix B.3.

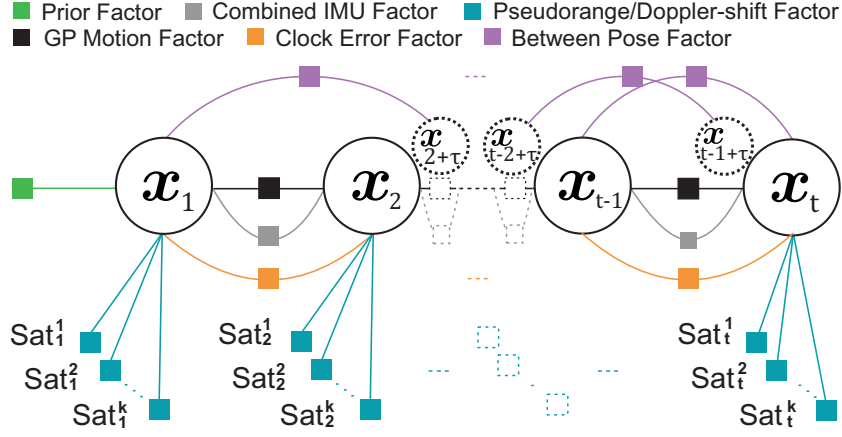


Figure 3.10: A general graph of tight coupling in gnssFGO with k raw satellite observations at each timestamp t denoted as Sat_t^k (reproduced from [M2] with permission of IEEE).

3.6.3 System Overview

The system overview, including the implementation of Algorithm 1 and all data interfaces, is shown in Figure 3.11. The sensor data are received and pre-processed in separate processes. I construct the time-centric factor graph in a two-stage process, as introduced in Algorithm 1.

The first stage (lines 4-10 of Algorithm 1) includes between-state factors and delay-free IMU factors to construct a deterministic graph over time. Subsequently, asynchronous sensor observations are fused into the deterministic graph by aligning the timestamps between the measurement and the state variables (lines 11-24 of Algorithm 1).

For measurements that cannot be aligned with any state, two successive state variables are queried to construct a GP-interpolated state for measurement evaluation in optimization procedures. The time-centric graph can be optimized using a fixed-lag batch optimizer [160] or a fixed-lag incremental smoother, i.e., iSAM2 [161], at a lower frequency.

In the experimental results, the estimated trajectories in the error metrics are optimized using iSAM2. I also evaluate both smoothers with respect to both estimator performance and computation efficiency, as presented in Section C.3.2. After each optimization procedure, I forward the optimized state variables to a state publisher and sensor pre-processing modules. The state publisher is associated with the IMU sensor and provides high-frequency state estimates at a rate of 200 Hz.

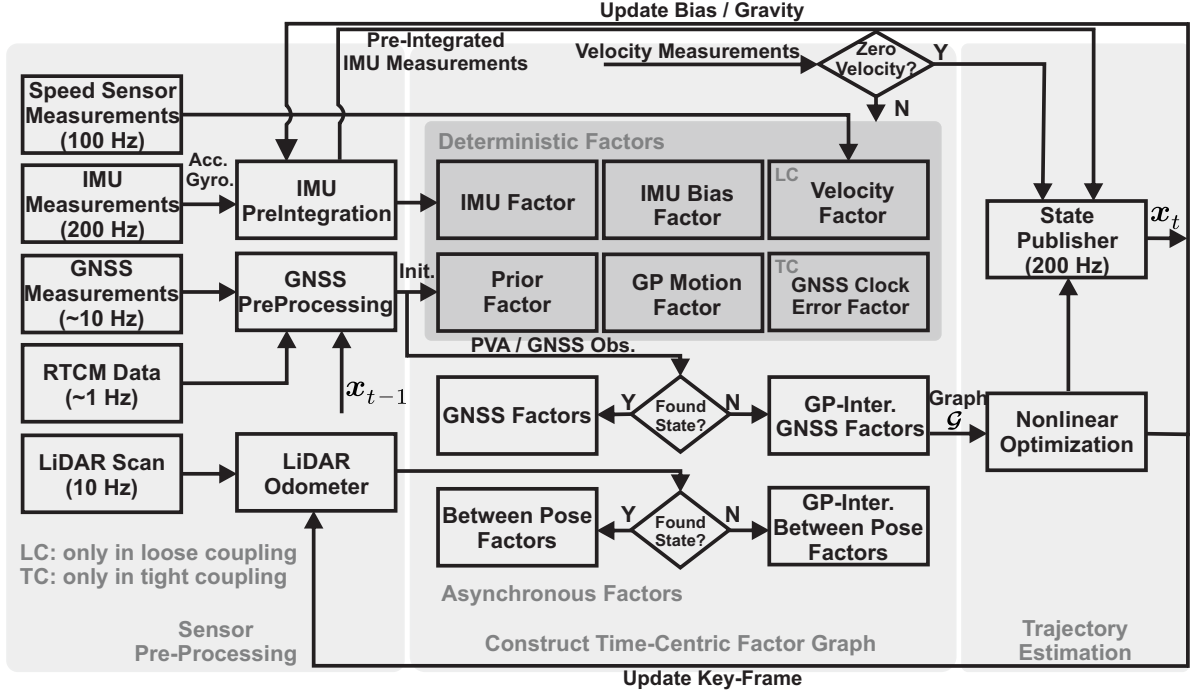


Figure 3.11: System overview showing all data interfaces and factor types by implementing Algorithm 1 (reproduced from [M2] with permission of IEEE).

3.6.4 Essential Experimental Results

This section presents the essential experimental results of Application II. For implementation details and more extensive results, I refer readers to Appendix C. With pre-defined test sequences in Table 2.3, I present the general error metrics for all experiments in Table 3.3 by taking the RTK-fixed GNSS-PVA solution as the ground truth.

Because an RTK-fixed solution is not available in challenging areas, I denote the solution rate used as a ground-truth reference to calculate the error metrics of each test sequence as a percentage in the column "Seq.". Due to limited space, figures cannot be presented at full scale; I thus upload all interactive figures to GitHub⁶.

3.6.4.1 Lidar-Centric Fusion

As shown in Table 3.3, the lidar-centric SLAM approach LIO-SAM failed in several test sequences even when the same factors with robust error modeling were used, and loop-closure

⁶[Online] https://github.com/rwth-irt/gnssFGO/tree/ros2/online_fgo/plots_tro

detection was enabled (see video demonstration⁷).

The most frequent reason is that scan registration fails due to an invalid feature association, which is observed in all failed test sequences. Another possible reason for the failure may be associated with corrupted GNSS observations that exhibit inconsistent noise values, leading to divergence in the optimization.

Figure 3.12b demonstrates this result, where the estimate diverges and cannot be recovered after the vehicle enters a tunnel. In Seq. HS, the lidar-centric approach cannot even be properly initialized while the vehicle moves very fast, which was not observed in the proposed gnssFGO. Furthermore, the estimated velocities and orientations using LIO-SAM show large variation and therefore less robustness compared to both fusion mechanisms in gnssFGO (see Figure 3.14c and Figure 3.14d).

Discussion on Lidar-Centric Fusion

We observe that graph construction triggered by the lidar odometer in [120] requires strict timestamp synchronization of GNSS measurements with lidar timestamps; therefore, asynchronous GNSS measurements are dropped.

Although lidar odometry and detected loop closures are still available, dropping GNSS measurements results in a loss of effective state constraints, so graph optimization becomes more sensitive to inaccurate scan registration. Therefore, the trajectory's smoothness and the estimate's accuracy are dramatically penalized (see Table 3.3).

This hypothesis is supported in the test sequences DUS and C01 (see Figure 3.14 and Figure 3.15), where the estimated height, orientation, and velocities were frequently diverted. Therefore, it can be observed that trajectory drift cannot be effectively eliminated using the classic sensor-centric localization approach LIO-SAM. Even worse, the robustness and reliability of sensor-centric approaches cannot be guaranteed in challenging areas once the primary sensor is compromised.

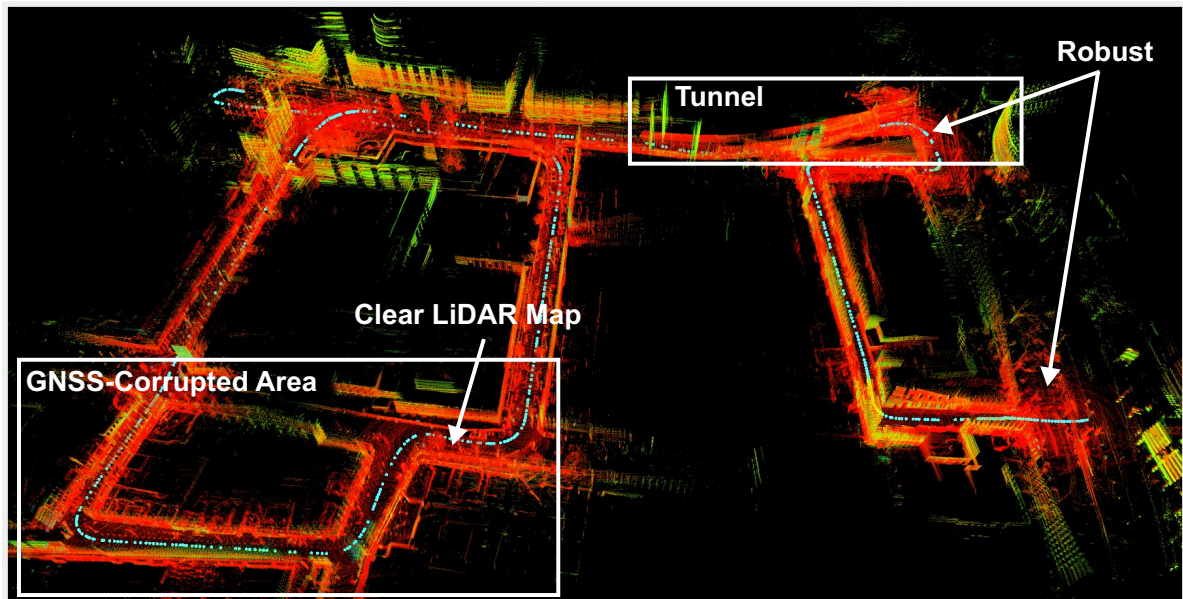
As online applications raise computation time and resource requirements, sensor degradation due to, e.g., insufficient data processing, becomes non-trivial. The proposed gnssFGO presents an effective workaround for fusing multiple sensors, thereby eliminating dependence on a single sensor and enabling the fusion of lossless information, which improves the robustness of the estimate in cases where sensor failure can be expected.

⁷[Online] <https://youtu.be/JhxJc1NFN7g>

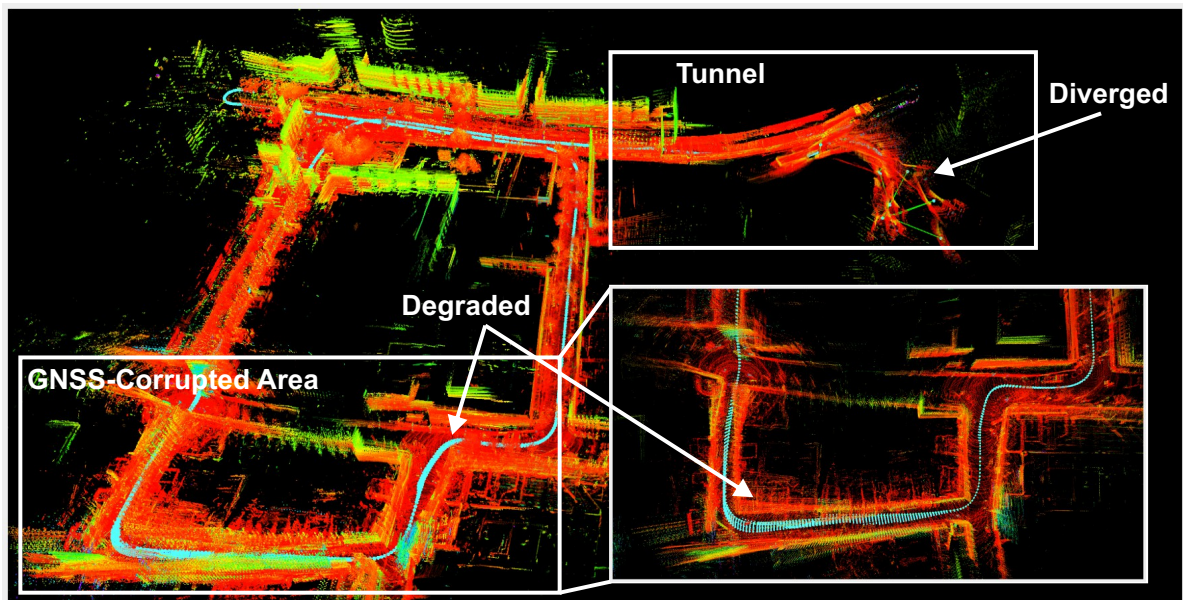
3 Continuous-Time Factor Graph Optimization

Table 3.3: General Trajectory Estimation Metrics in Application II. A test run is classified as failed if the algorithm diverges. We show the RTK-fixed PVA solution ratio in percent from the high-grade NovAtel GNSS receiver under each sequence’s name in column Seq. (e.g., AC (52 %)) that is used to calculate the error metrics. Calculating the smoothness as a relative performance metric is independent of the ground-truth trajectory (reproduced from [M2] with permission of IEEE).

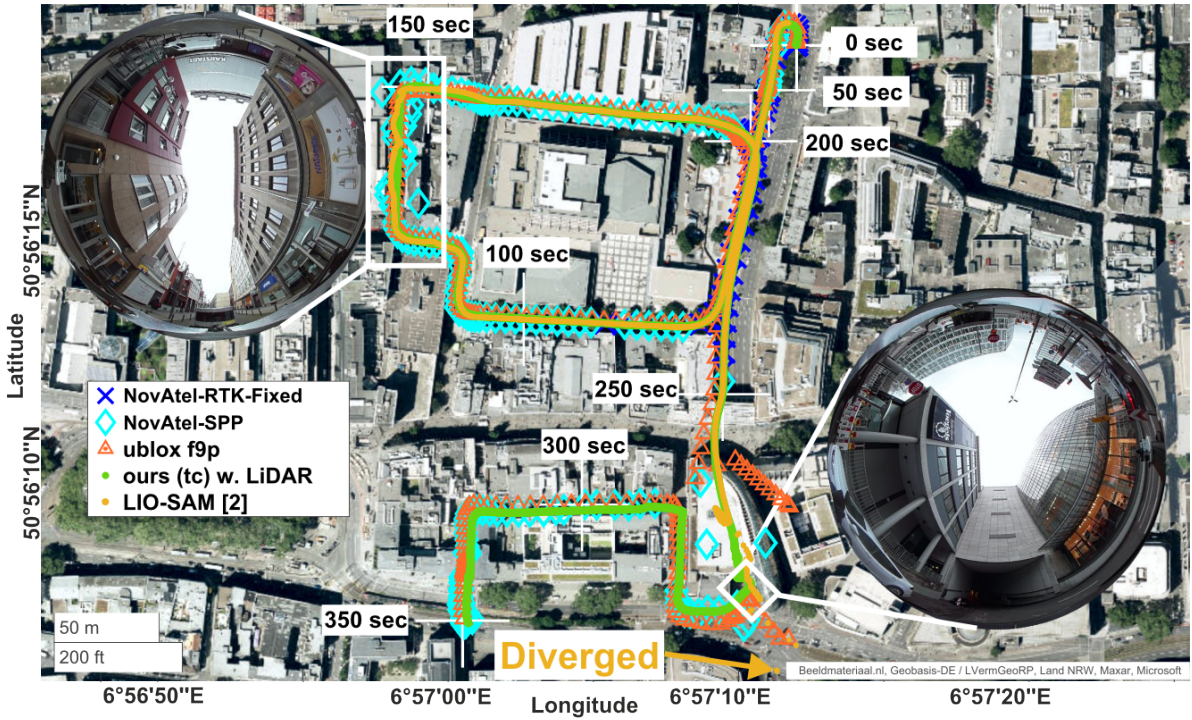
Seq.	Configuration	Mean 2D Pos. Err. (m)	2D Pos. STD (m)	Max. 2D Pos. Err. (m)	Mean 3D Pos. Err. (m)	3D Pos. STD (m)	Max. 3D Pos. Err. (m)	Mean Yaw Err. (°)	Yaw STD (°)	Max. Yaw Err. (°)	Smooth. _s
AC (52%)	ublox-f9p	0.889	±0.237	12.905	99.424	±0.293	263.24	6.651	±37.54	359.9	1.07e+6
	LIO-SAM [2]	Failed (diverged occasionally)									
	ours (LC) w/o. MultiSensor	0.732	±0.152	12.378	0.918	±0.217	12.619	42.984	±4.57	90.0	7301.52
	ours (LC) w. MultiSensor	0.770	± 0.149	15.299	0.960	± 0.1964	15.770	44.365	±2.14	89.99	1439.82
	ours (TC) w/o. LiDAR	0.936	±0.650	5.877	1.276	±1.392	6.127	2.76	±2.68	88.53	1706.63
	ours (TC) w. LiDAR	0.476	±0.738	1.605	1.057	±1.088	2.343	1.18	± 2.13	85.12	982.40
DUS (18%)	ublox-f9p	3.441	±2.413	19.284	76.939	±3.121	276.001	1.956	±62.44	9.368	8250.5
	LIO-SAM [2]	0.736	-	9.129	1.024	-	17.391	1.758	-	80.44	2.15e+5
	ours (LC) w/o. MultiSensor	4.138	±1.228	19.544	9.256	±1.813	36.166	40.202	±5.41	89.745	1.02e+7
	ours (LC) w. MultiSensor	2.631	±0.462	9.402	7.668	± 0.950	39.806	39.697	±1.33	89.980	1.19e+9
	ours (TC) w/o. LiDAR	1.123	±1.081	4.415	1.461	±2.524	7.141	0.473	±3.01	81.32	3595.72
	ours (TC) w. LiDAR	1.074	± 0.822	4.570	2.170	±1.890	4.789	0.511	± 2.40	6.018	1510.48
C01 (57%)	ublox-f9p	3.126	±52.01	21.583	106.731	±52.872	259.399	76.175	±40.81	258.040	1.580e+4
	LIO-SAM [2]	3.339	-	8.186	3.478	-	8.463	3.774	-	34.260	6.30e+5
	ours (LC) w/o. MultiSensor	4.290	±2.350	23.662	10.085	±3.307	41.421	82.340	±10.1	89.830	435.89
	ours (LC) w. MultiSensor	7.697	±1.258	17.487	12.221	±2.050	45.104	87.164	± 1.77	88.384	1.20e+5
	ours (TC) w/o. LiDAR	0.727	±1.52	1.125	0.78	±3.194	1.179	0.935	±3.209	6.534	5.38e+3
	ours (TC) w. LiDAR	0.402	± 0.846	0.807	0.468	± 1.892	0.841	0.888	±2.76	7.025	122.358
C02 (27%)	ublox-f9p	3.632	±14.69	6.745	76.043	±16.053	256.092	2.4326	±91.67	357.521	1.12e+4
	LIO-SAM [2]	Failed (diverged occasionally)									
	ours (LC) w/o. MultiSensor	7.421	±1.350	33.417	25.919	±2.020	88.612	28.084	±7.86	40.242	562.92
	ours (LC) w. MultiSensor	4.029	± 0.479	20.393	18.227	± 1.058	53.803	10.673	±4.87	58.392	7.29e+6
	ours (TC) w/o. LiDAR	0.933	±1.111	1.933	1.188	±2.836	2.104	0.927	±4.15	3.811	198.03
	ours (TC) w. LiDAR	0.264	±1.039	0.499	0.328	±2.402	0.565	0.496	± 2.40	9.716	218.42
HS (78%)	ublox-f9p	0.645	± 0.217	1.361	33.486	± 0.273	66.942	0.977	± 0.236	2.453	0.004
	LIO-SAM [2]	Failed (diverged on initialization)									
	ours (LC) w/o. MultiSensor	1.354	±0.520	6.010	2.167	±0.729	11.873	87.287	±3.63	89.156	0.248
	ours (LC) w. MultiSensor	3.136	±1.299	15.571	8.051	±1.634	41.482	88.902	±0.98	89.954	0.300
	ours (TC) w/o. LiDAR	0.404	±0.509	2.265	0.822	±1.08	4.833	0.993	±1.69	10.05	0.139
	ours (TC) w. LiDAR	0.475	±0.672	3.121	1.400	±1.252	3.609	0.699	±0.94	3.778	0.060



(a) Lidar map and trajectory using tightly coupled gnssFGO (ours). The proposed method provides robust trajectory estimation and clear lidar maps in GNSS-corrupted areas.



(b) Lidar map and trajectory using LIO-SAM [120]. This approach fused with GNSS positioning failed due to faulty scan registrations while crossing a tunnel and degrades dramatically with corrupted GNSS measurements.



(c) Lidar map and trajectory using tightly coupled gnssSFGO (ours). The proposed method provides robust trajectory estimation and clear lidar maps in GNSS-corrupted areas.

Figure 3.12: Demonstration of multisensor fusion for vehicle localization of test sequence C02 in the city of Cologne (reproduced from [M2] with permission of IEEE).

3.6.4.2 Loosely Coupled (LC) Fusion

It can be observed in Table 3.3 that the accuracy of the low-grade GNSS receiver, especially in the vertical dimension (height), is dramatically degraded compared to the high-grade GNSS receiver. Furthermore, this receiver does not characterize its noise values, so the standard deviations provided are strongly inconsistent with the real noise. Therefore, fusing the GNSS PVT solution (see Appendix A.3.4.2) from the low-grade GNSS receiver significantly downgrades the performance of the loosely coupled fusion in all test sequences, even when multiple sensor observations are fused.

In a less challenging environment (Seq. AC, see Figure 3.13), the proposed loosely coupled fusion generally outperforms the original PVT solution. However, the same improvement cannot always be expected in challenging environments when comparing the error metrics of other test sequences. The primary reason for this result is caused by inconsistent noise values and highly inaccurate height measurements in the GNSS-PVT observations.

Another interesting phenomenon is that the loosely coupled fusion of the GNSS-PVT solution with other sensors can present a degraded performance (see Seq. C01 and Seq. HS).

Due to inaccurate height measurements that present high variations (see Figure 3.14b and Figure 3.15b), the pose of the lidar keyframes cannot be stably optimized, so the keyframes, which are used to calculate the relative motion of each scan, present different height values. In such cases, the estimated lidar odometry is associated with incorrect relative motion increments, which degrade graph optimization (see height in Figure 3.15b and Vd in Figure 3.15d).

To improve the robustness of the estimation by acquiring redundant state constraints, I utilized a high-grade speed sensor in this fusion approach, expecting it to effectively constrain the unobserved states when GNSS solutions are compromised. However, my experiments indicate that the 2-D velocity measurements provided by the 2-D speed sensor cannot sufficiently constrain the state space, especially when vehicle orientation cannot be observed.

Discussion on LC-Fusion

Although the loosely coupled fusion using the proposed gnssFGO did not fail in the test sequences, we can observe the same conclusion as shown in [104] that loosely coupled fusion cannot serve as a robust state estimator in challenging areas where GNSS solutions present inconsistent uncertainties.

However, I indicate that this fusion mechanism can present fast estimation convergence as long as accurate GNSS-PVT solutions are available. This is because the state variables (position and velocity) can be directly observed in the GNSS-PVT solution, where the measurement model does not present high nonlinearity, and thus, effective state constraints are presented.

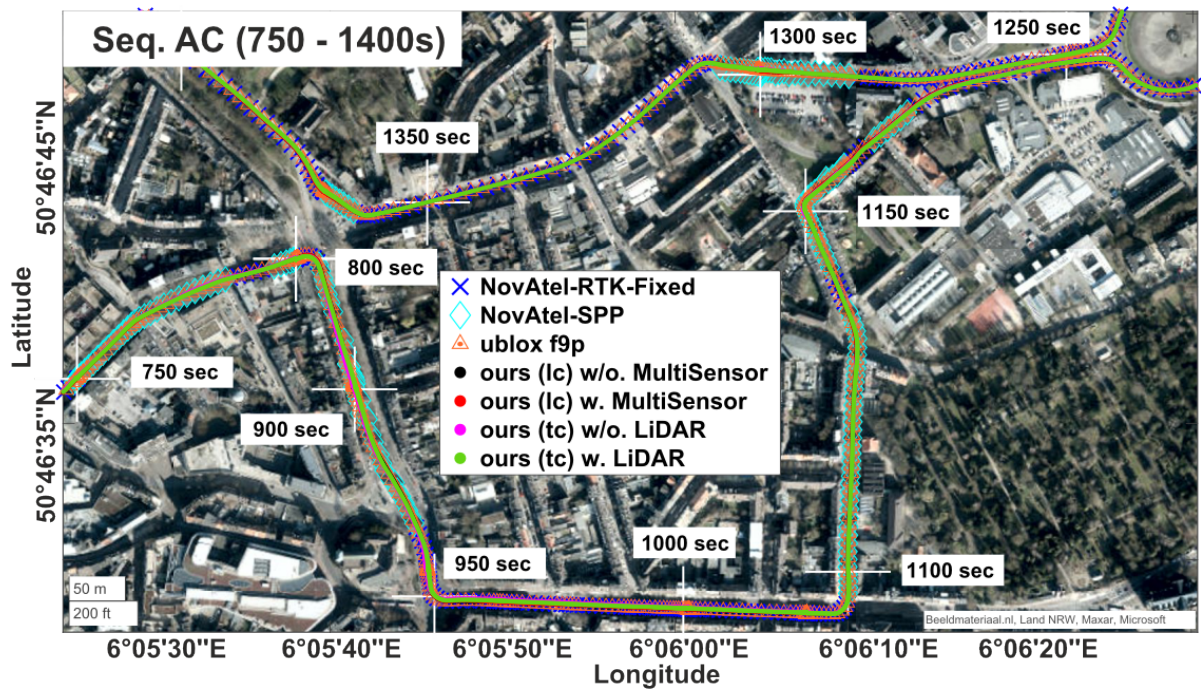
3.6.4.3 Tightly Coupled (TC) Fusion

Compared to loosely coupled sensor fusion, fusing pre-processed GNSS observations in a tight coupling contributes to redundant state constraints. Thus, the tightly coupled fusion generally presents more robust trajectory estimations with a smaller maximum position error and greater trajectory smoothness in lengthy runs, except in the high-speed scenario (Seq. HS).

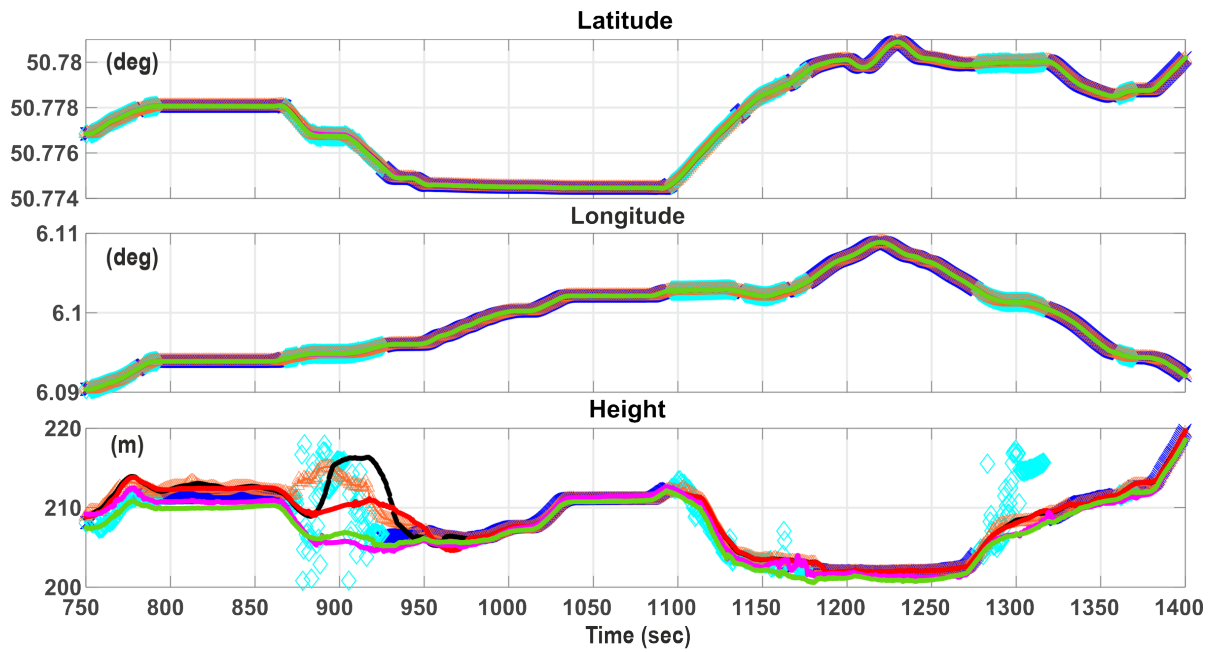
In challenging urban areas such as Seq. C01 and C02, by fusing the lidar odometry as between-state constraints, generally improve the estimation performance and trajectory smoothness. This conclusion can also be drawn when referring to Figure 3.14b and Figure 3.15b, where more accurate height and velocity estimations can be observed by fusing lidar odometry in the graph.

In high-speed scenarios, lidar scans suffer from serious motion distortion, and insufficient features can be extracted compared to urban areas. Therefore, a limited performance improvement can be observed by fusing lidar odometry in the graph.

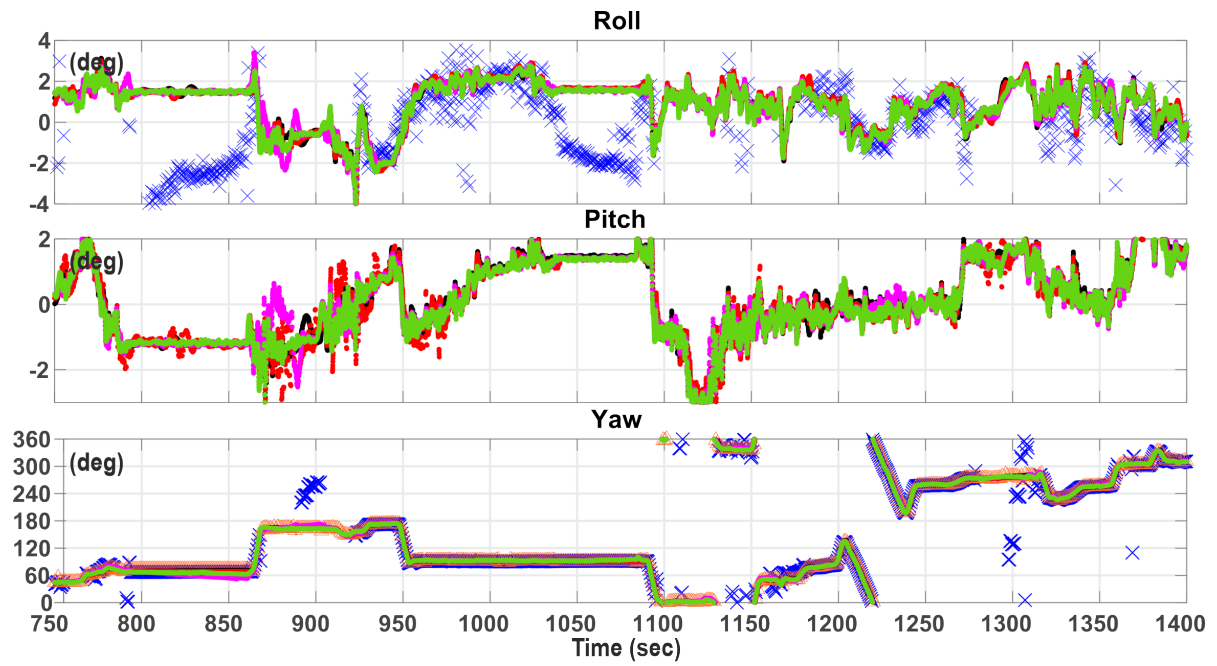
3 Continuous-Time Factor Graph Optimization



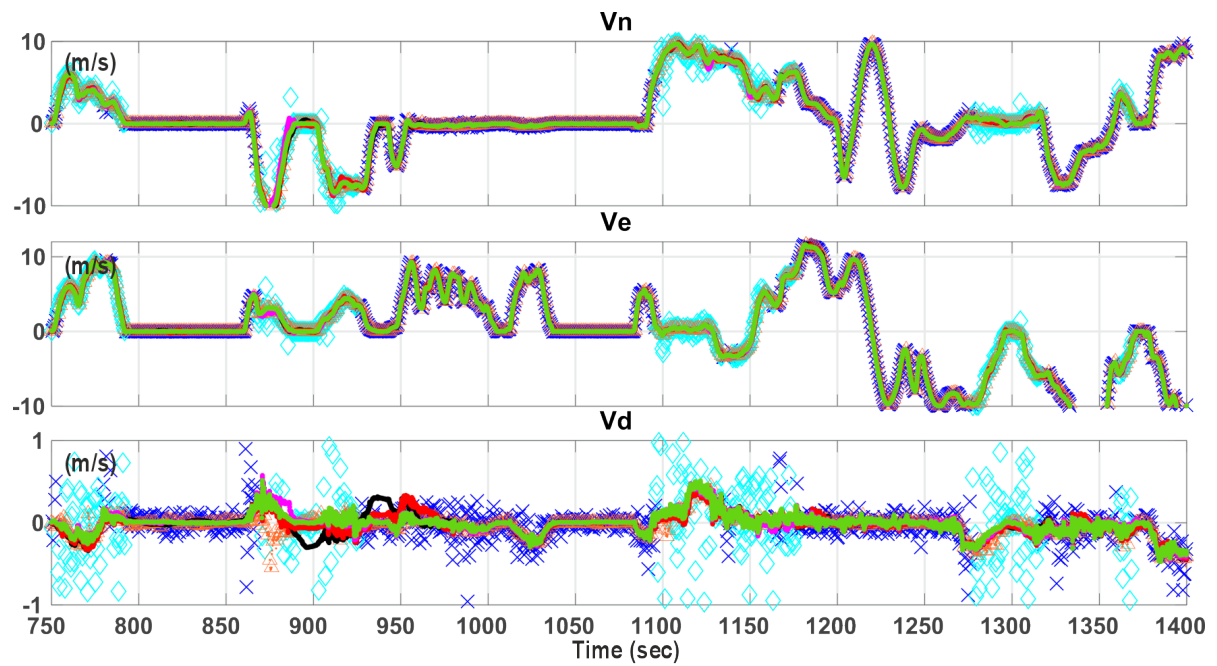
(a) Trajectories of Seq. AC.



(b) Coordinates in the WGS84 frame.



(c) Estimated rotation in the NED frame.



(d) Estimated velocity in the NED frame.

Figure 3.13: Trajectory plot (700 s - 1400 s) in urban areas in Aachen. I plot the GNSS single point position (SPP) if the RTK-fixed solution is unavailable (reproduced from [M2] with permission of IEEE).

Discussion on TC-Fusion

Based on the experimental results presented above, a robust trajectory estimation can be achieved in challenging scenarios using the proposed approach by fusing multiple sensor measurements in a tight coupling, which supports my hypothesis proposed in Section 3.1. In contrast to the loose coupling, the tightly coupled multisensor fusion presents a more robust trajectory estimation in my experimental studies.

The same conclusion has also been shown in Wen et al. [104] and Cao et al. [123]. However, acceptable accuracy cannot be achieved, especially in dense urban scenarios. For instance, although all estimated trajectories using the proposed gnssFGO in Figure 3.12 remain consistent, a large drift is presented using the proposed sensor integration.

Possible reasons to explain this phenomenon can be attributed to lidar degradation, insufficient GNSS observations, and inconsistent sensor noise models, which are influenced by the presence of outliers. As gnssFGO provides a flexible fusion mechanism, this problem can be addressed by integrating more effective state constraints into the graph.

3.6.5 Discussion on Application II

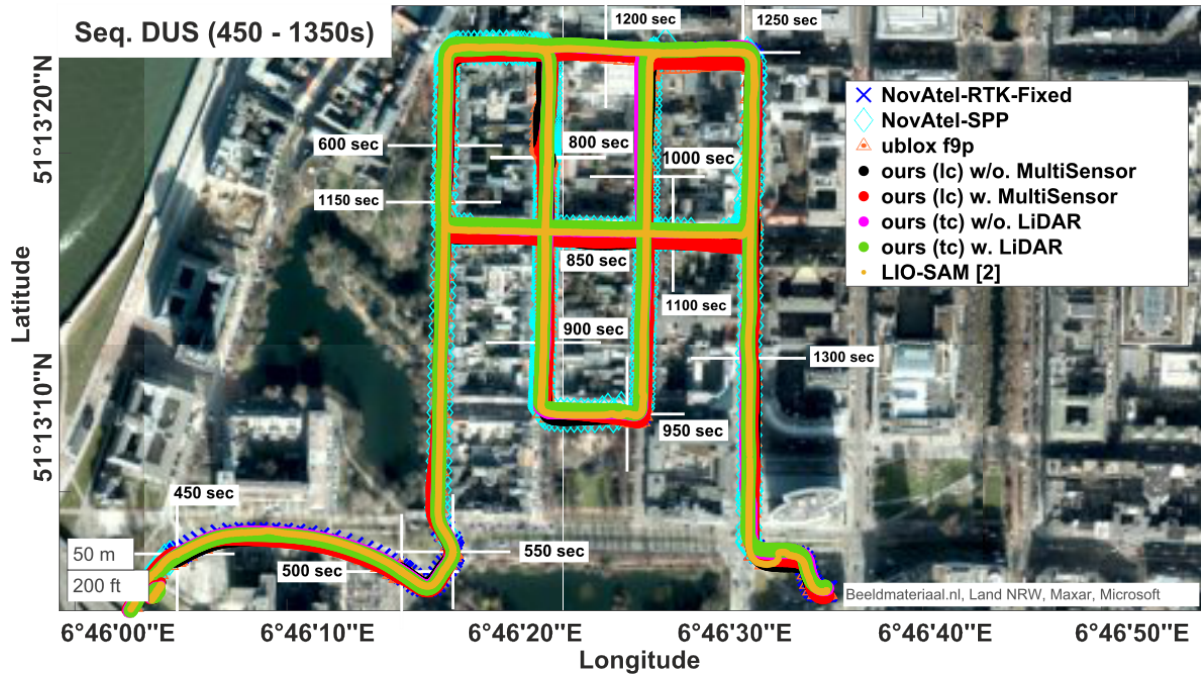
In this application, I successfully integrated asynchronous sensor measurements into the proposed framework, achieving robust vehicle localization in challenging environments.

Experimental studies demonstrate that the proposed method is robust, flexible, and accurate, operating online across multiple datasets collected in diverse and challenging scenarios, including urban areas and high-speed tracks. Notably, all FGO configurations succeeded across all test sequences, whereas the classic state-of-the-art lidar-centric method [120] encountered failures in some situations. The experimental results further highlight that the GP-WNOJ motion prior enables accurate continuous-time trajectory representation when its hyperparameters are properly tuned.

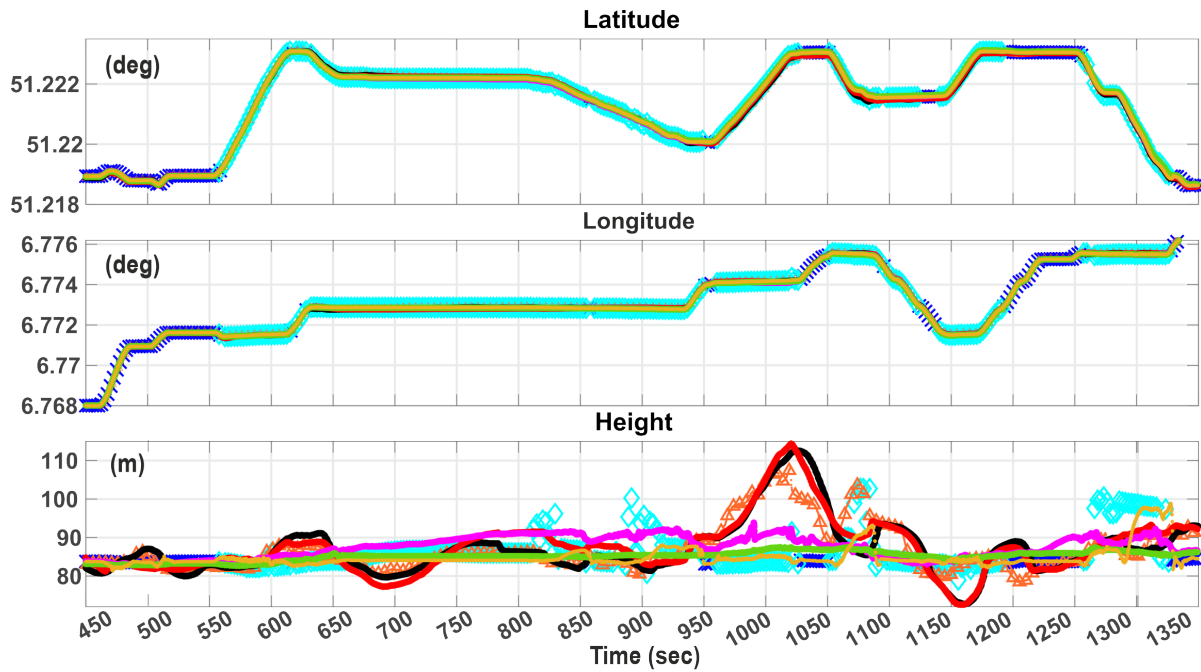
However, I did not fully exploit the available GNSS observations—particularly the carrier-phase measurements—which require sophisticated techniques for ambiguity resolution and cycle slip detection.

Additionally, advanced methods for mitigating multipath effects and excluding non-line-of-sight GNSS signals could be integrated into my state estimator framework. I also overlooked hyperparameter tuning of the GP-WNOJ model and the identification of sensor noise characteristics, both of which can be addressed online using learning-based approaches.

Furthermore, incorporating additional sensor modalities, such as visual odometry, into the fusion framework could further enhance performance. These directions collectively define the scope of future work.

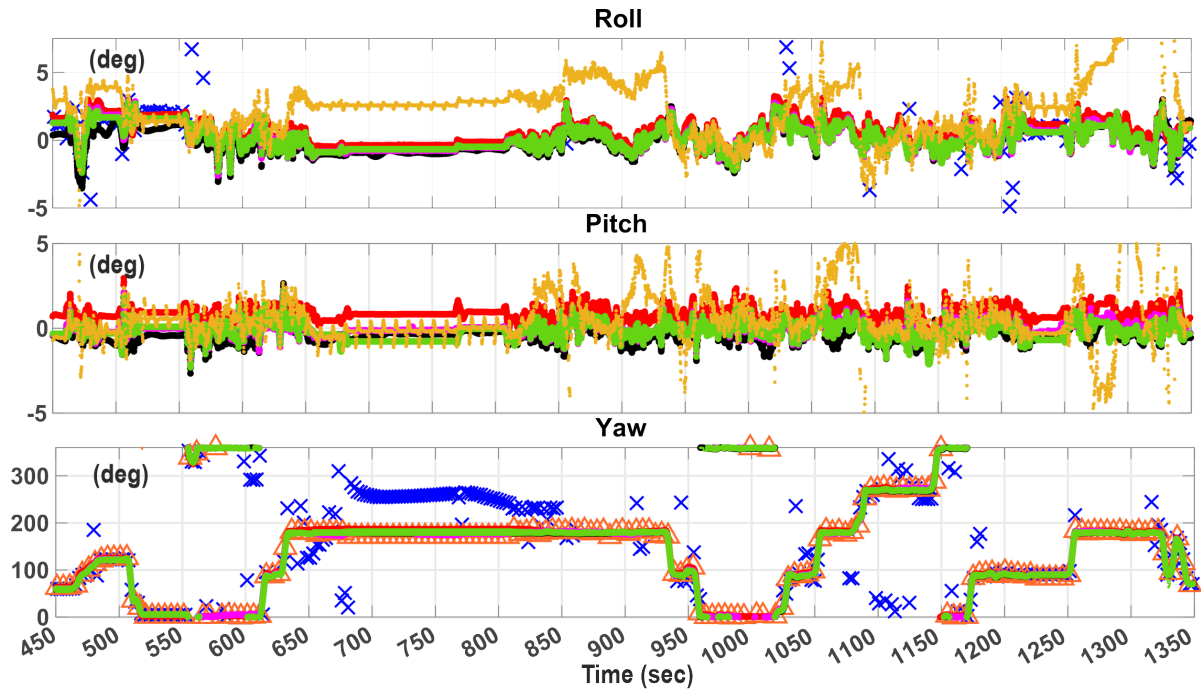


(a) Trajectories of Seq. DUS.

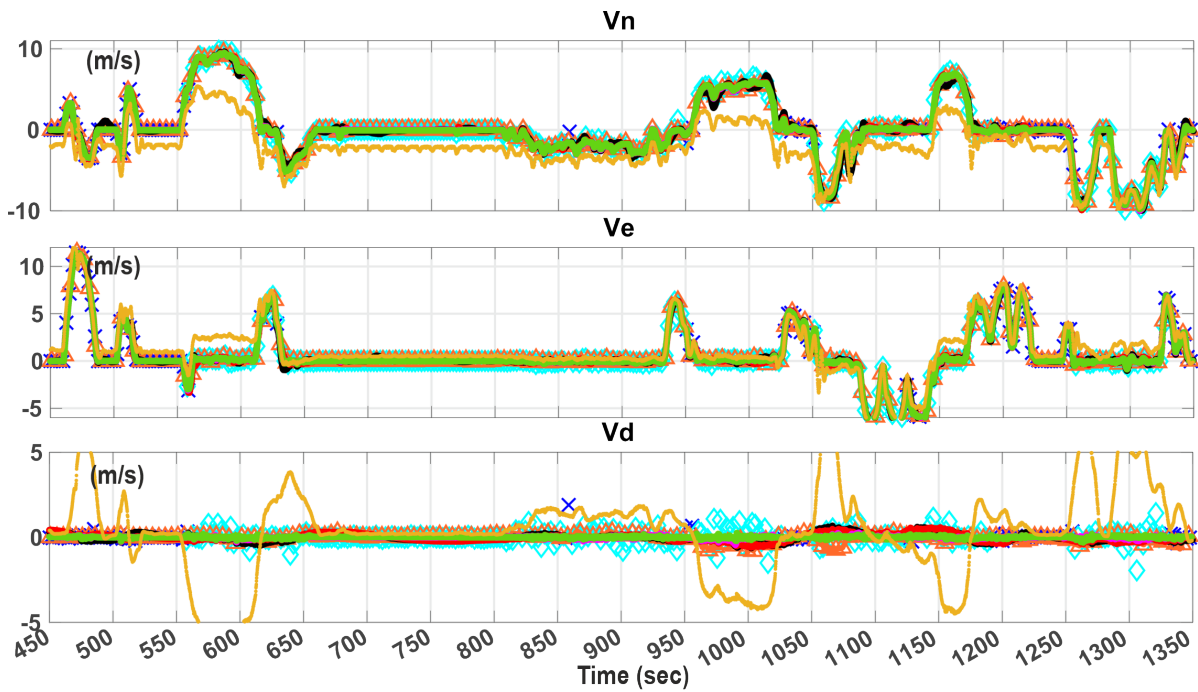


(b) Coordinates in the WGS84 frame.

3 Continuous-Time Factor Graph Optimization

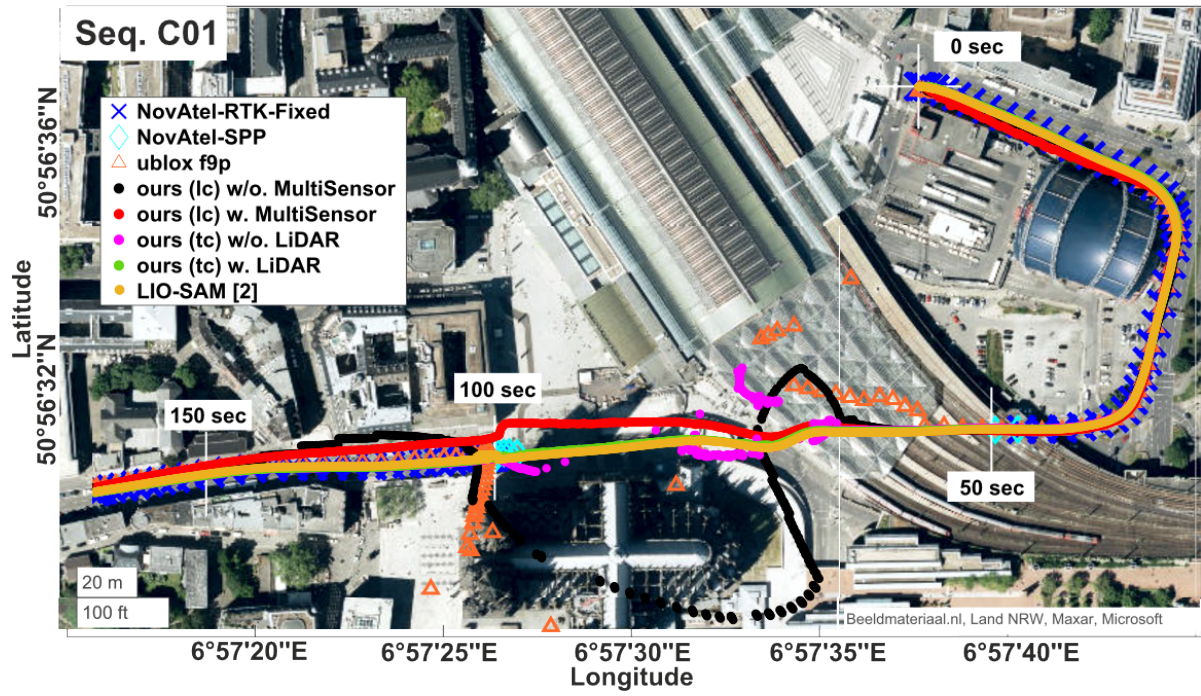


(c) Estimated rotation in the NED frame.

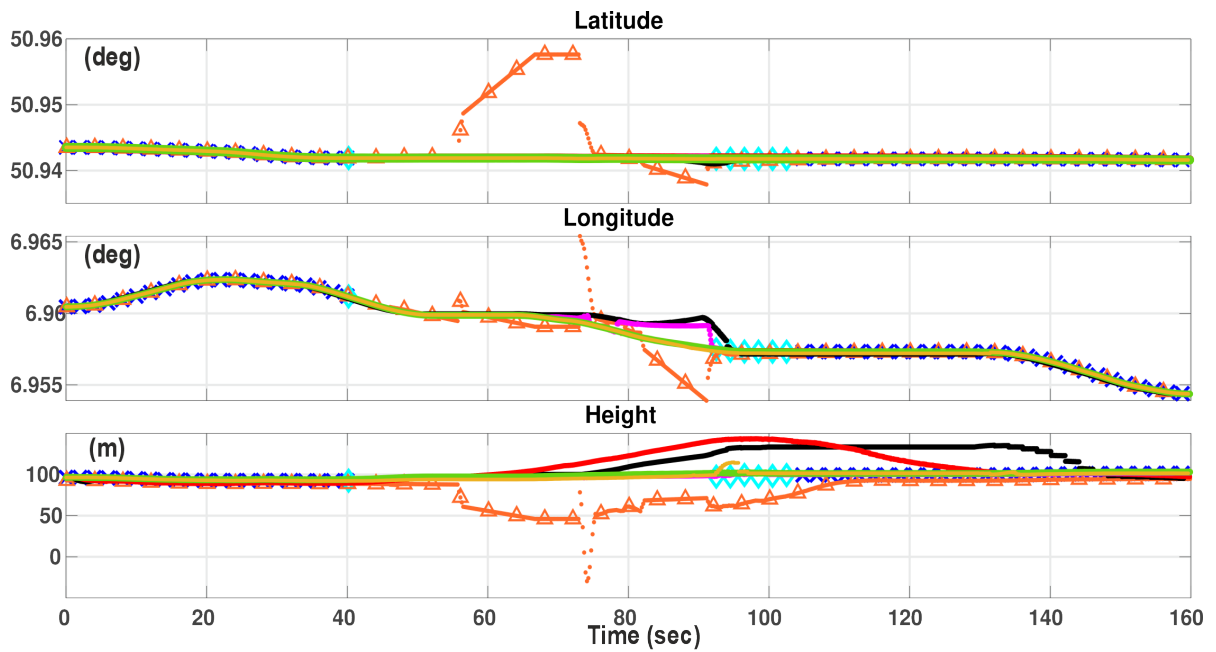


(d) Estimated velocity in the NED frame.

Figure 3.14: Trajectory plot (450 s - 1350 s) in challenging areas in Düsseldorf (reproduced from [M2] with permission of IEEE).

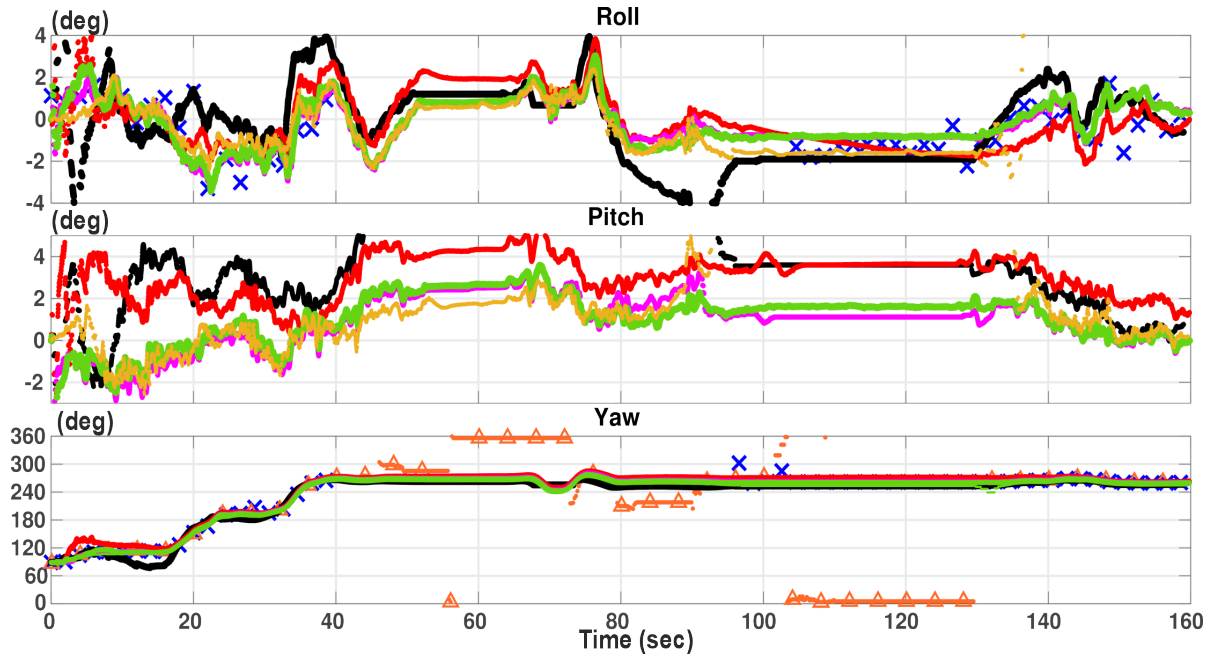


(a) Trajectories of Seq. C01.

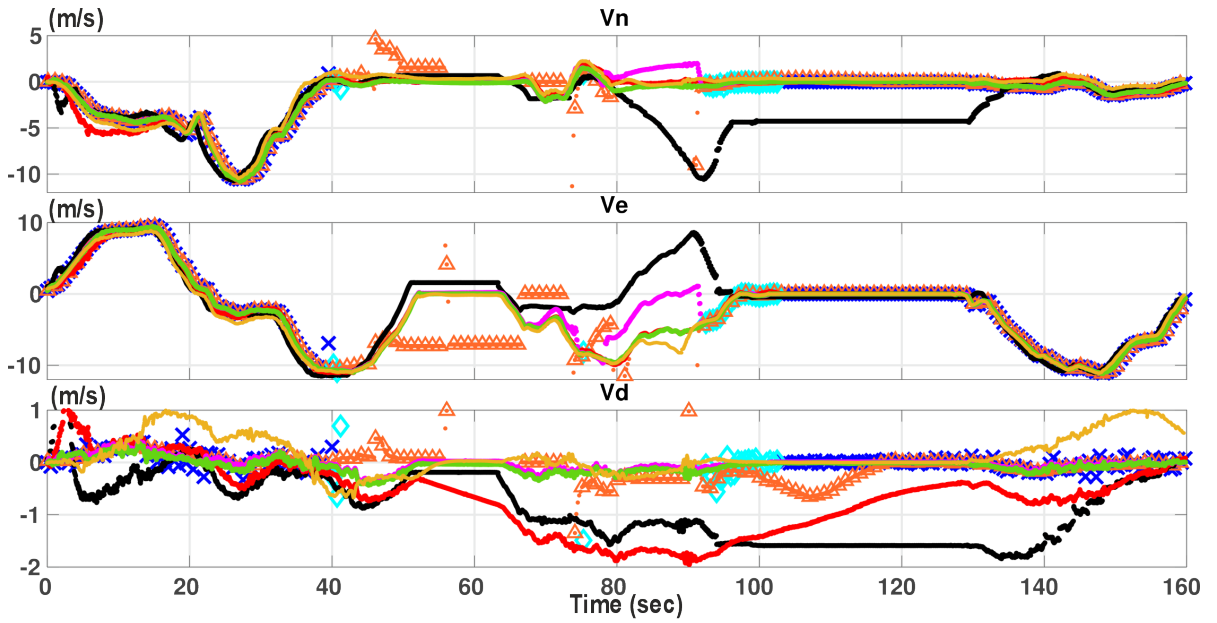


(b) Coordinates in the WGS84 frame.

3 Continuous-Time Factor Graph Optimization



(c) Estimated rotation in the NED frame.



(d) Estimated velocity in the NED frame.

Figure 3.15: Trajectory plot near the central station of Cologne (Seq. C01). For the tightly coupled fusion, I omitted the near-zero velocity detection to present trajectory drifting while the receiver clock error is unobservable (reproduced from [M2] with permission of IEEE).

CHAPTER 4

Handling Outliers using Classic M-Estimators

“Far better an approximate answer to the right question, than an exact answer to the wrong question.”

– John Tukey

4.1	Motivation and Objective	73
4.2	Related Work	74
4.2.1	M-Estimation for GNSS-based Localization	74
4.2.2	Self-Tuning M-Estimators	75
4.3	M-Estimation	76
4.3.1	Robustness and Relative Efficiency	77
4.3.2	Kernel Functions	78
4.3.3	Influence Analysis	79
4.3.4	Breakdown Point	82
4.4	Kernel Evaluation	82
4.4.1	Experiment Setup	82
4.4.2	Essential Experimental Results	83

4.1 Motivation and Objective

M-estimation, originally derived from robust statistics [162], has been extensively applied in state estimation problems over the past decades to address measurement outliers without explicitly identifying their uncertainties. Unlike traditional least-squares estimation, which assumes zero-mean Gaussian noise processes, M-estimation minimizes a modified loss function with configurable kernel functions, effectively reducing the influence of large residuals. This adaptability makes M-estimation particularly effective in practical scenarios

where sensor measurements are corrupted by non-Gaussian noise or other anomalies (see Section 1.2.2).

However, the performance of M-estimation heavily depends on the choice of kernel functions and an appropriately tuned scaling parameter, which balances the estimator's *robustness* against outliers and its *efficiency* in utilizing valid measurements. Most practical applications utilizing M-estimators select the kernel function and its hyperparameter based on limited ad-hoc experiments. These studies lack a systematic understanding of the effectiveness of different kernel functions in balancing robustness and efficiency, leaving the choice of kernel largely unexplored and context-dependent.

Therefore, this chapter focuses on the study of M-estimators in the context of range-based vehicle localization, evaluating their performance across a set of widely used kernel functions. My motivation is driven by a more fundamental question:

“what should be considered and analyzed from a theoretical perspective when deploying an M-estimator to handle GNSS outliers?”

I believe that addressing such theory-grounded questions is essential before investigating more advanced techniques, such as adaptive M-estimators.

Accordingly, this chapter aims to empirically analyze selected kernel functions through comprehensive experiments conducted across diverse scenarios. The chapter begins with an introduction to the theoretical foundations of M-estimation, providing insight into the principles underlying kernel configuration. Based on the experimental findings, I present guidelines for selecting or designing suitable kernel functions and for tuning their parameters appropriately, thereby addressing Research Question **Q2** defined in Section 1.4.2.

4.2 Related Work

In Chapter 1, Section 1.2.2, I briefly introduced M-estimation as a sub-approach for robust error modeling in state estimation problems. This section examines the literature related to the application of M-estimation in GNSS-based localization.

4.2.1 M-Estimation for GNSS-based Localization

The use of M-estimation to reject GNSS measurement outliers has garnered significant attention over the past decades. Early studies analyzed the effectiveness of M-estimation for GPS-only positioning methods and demonstrated its satisfactory performance [163, 164].

Subsequently, numerous studies have incorporated M-estimators into Kalman filters, confirming their effectiveness in handling GNSS measurement outliers. This integration has significantly improved the accuracy and robustness of vehicle positioning [165–167].

Additionally, research has investigated the performance of M-estimators for outlier rejection in harsh environments, where GNSS observations are highly corrupted due to multipath or non-line-of-sight receptions [168–170]. These works provide valuable insights into leveraging M-estimators to enhance traditional state estimators for robust GNSS-based vehicle localization.

Recently, Crespillo et al. [171] presented studies on designing M-estimators using Huber loss for robust positioning in urban environments, focusing on mitigating multipath and non-line-of-sight signals through various pseudorange models.

However, all these studies predominantly focused on commonly used kernel functions within the category of hard-redescending estimators, such as Huber and Tukey. None have explored the potential of other kernels, particularly soft-redescending kernel functions, for addressing robust GNSS-based vehicle localization.

Furthermore, these studies also fail to provide insights into parameter tuning and the trade-off between robustness and efficiency across different M-estimators. In addition, previous studies, such as Zhang [38], De Menezes et al. [39], and MacTavish and Barfoot [40], which evaluate M-estimation with various kernels and parameterizations, are not specifically tailored for range-based vehicle localization.

Given these reasons, a comprehensive investigation of M-estimators across different categories, focusing on their mathematical properties, becomes essential to better understand their performance and applicability in robust GNSS-based vehicle localization.

4.2.2 Self-Tuning M-Estimators

GNSS-based vehicle localization typically requires robust state estimators capable of long-term operation in large-scale environments, where GNSS observations exhibit diverse noise patterns. Consequently, M-estimators with fixed parameterization often experience degraded performance under such varying conditions.

To address this issue, recent studies have focused on adaptive M-estimation techniques, which aim to dynamically select the optimal parameterization based on current error statistics. Ding et al. [49] proposed a learning-based approach that formulates the selection of the *Huber loss* hyperparameter as a regression problem using the residuals of GNSS observations. This method demonstrated a slight improvement in performance when using the self-learning Huber estimator.

Another recent study by Das and Gross [172] focused on tuning the hyperparameter of the generalized kernel function proposed by Barron [50] within a two-stage optimization framework, resembling the expectation-maximization algorithm. Their approach utilized past measurement observations to infer outlier behavior, which could be mitigated by selecting an appropriate kernel based on the online-tuned hyperparameter.

However, as my hypothesis **H2.2** suggests, tuning the hyperparameter for M-estimators solely based on past measurement residuals may be shortsighted when noise patterns suddenly change. Moreover, in GNSS-based vehicle localization, the efficiency of the estimators plays a crucial role alongside robustness, especially when satellite observations are severely limited.

4.3 M-Estimation

The goal of robust statistics is to develop techniques replicating traditional statistical methods while remaining relatively unaffected by outliers or small deviations in the data [162]. Robust estimators are broadly categorized into three classes: *L-estimates*, *R-estimates*, and *M-estimates*. L- and R-estimators are particularly useful in robust descriptive statistics and statistical education, but require significant computational effort to calculate the full statistical distribution for specific problems.

On the contrary, M-estimators offer an alternative approach to addressing outlier problems. They are preferred due to their fixed functional form, simplifying implementation. Additionally, M-estimators allow for a more straightforward incorporation of prior information than L- and R-estimates. Unlike sequential regression-based techniques, M-estimators can simultaneously address outlier detection during the estimation process [39].

We recall the nonlinear least-square optimization problem (A.33) introduced in Section A.2.3, where the cost function takes the quadratic form under the Gaussian assumption

$$J(\mathbf{e}) = \|\mathbf{e}\|^2 \triangleq \|\mathbf{z} - \mathbf{h}(\mathbf{x})\|_{\Sigma}^2. \quad (4.1)$$

This cost function is commonly referred to as the *L2* loss.

Since the quadratic *L2* loss is sensitive to outliers, M-estimators aim to reshape the quadratic loss by introducing a symmetric, positive-definite function $\rho(\cdot)$ with a unique minimum at zero. This modifies the original nonlinear least-square optimization problem introduced in Section A.2.3 as follows

$$\hat{\mathcal{X}}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{k=0}^K \rho(\mathbf{e}_k). \quad (4.2)$$

The new loss function $\rho(\cdot)$ in (4.2) is typically designed to reduce the influence of large residuals $\mathbf{e}$, which are often caused by outliers.

To solve the optimization problem in (4.2), we equivalently update the to-be-estimated state variables in each iteration given

$$\sum_{k=0}^K \Psi(e_k) \frac{\partial e_k}{\partial x} = \mathbf{0}, \quad (4.3)$$

where the function $\Psi(e_k) = \frac{d\rho(e_k)}{de_k}$ is referred to as the *score function*. The score function approximately measures the influence of the residual, and thus it is a crucial tool for evaluating the robustness of an M-estimator. It quantifies the impact of an *infinitesimal contamination* at a specific point e on the estimate, normalized by the mass of the contamination. This interpretation provides insight into how sensitive an estimator is to small deviations or outliers in the measurements, making it a fundamental concept in robust statistics.

Since the cost function $\rho(e)$ of M-estimators does not follow the traditional “least-square” form, it can be reformulated into a sub-quadratic form

$$\rho(e_k) = w(e_k^*) \cdot e_k^2, \quad (4.4)$$

where we introduce the *weighting function* $w(e_k^*)$, defined as

$$w(e_k^*) = \frac{\Psi(e_k^*)}{e_k^*}, \quad (4.5)$$

based on the residual e_k^* from the optimal state (often the previous residual). This reformulation enables the iterative computation of robust estimates by adapting the influence of residuals dynamically through the weighting function [38]. Therefore, M-estimators can be directly integrated into the solving procedures of the batch-optimization state estimators introduced in Section A.2.7, formulating an *iterated reweighted (nonlinear) least-square* problem.

4.3.1 Robustness and Relative Efficiency

Robust statistics introduces two fundamental concepts, *robustness* and *relative efficiency*, to evaluate the performance of statistical estimators. Robustness refers to an estimator’s ability to remain effective and reliable in the presence of deviations from model assumptions, such as outliers, heavy-tailed distributions, or other anomalies in the data. On the other hand, relative efficiency measures the trade-off between robustness and the precision of the estimator under ideal conditions. It can be calculated as

$$E_{\text{ff}} = \frac{V_{\text{ref}}}{V} \quad (4.6)$$

using the *asymptotic variance* of the reference estimator (e.g., $L2$ -estimator), V_{ref} , and the *asymptotic variance* V of the target M-estimator. We can compute the asymptotic variance using the following equation

$$V[\Psi(x), f(x)] = \frac{\int_{-\infty}^{+\infty} \Psi^2(x) f(x) dx}{(\int_{-\infty}^{+\infty} \Psi(x) f'(x) dx)^2}, \quad (4.7)$$

where the function $f(x)$ denotes the true error distribution.

As noted by Albuquerque and Biegler [173]: “**The more robust an estimator is, the less efficient it is**”. A more efficient estimator requires fewer input data or observations than a less efficient one to achieve the Cramér–Rao bound (see Appendix A.2.2.3), which serves as a benchmark for the “best” possible performance that an unbiased estimator can achieve.

To understand the concept of relative efficiency, consider two estimators with variances v_1 and v_2 . Assuming both estimators have roughly normal distributions with variances v_1/n and v_2/n , relative efficiency compares the amount of data needed by each estimator to achieve the same variance. For instance, if $v_1 = 3v_2$, the first estimator requires three times as many observations as the second to reach the same level of variance, highlighting the second estimator’s higher efficiency.

In this thesis, I select the tuning parameters of all kernel functions for the same efficiency level. The tuning parameters are calculated using the Equations (4.6) and (4.7) given a fixed efficiency level. For more details, I refer the readers to De Menezes et al. [39].

4.3.2 Kernel Functions

Table 4.1 lists all the kernel functions used in the M-estimators investigated in this thesis. The kernel functions and their influence functions at the same efficiency level $E_{\text{ff}} = 95\%$ are illustrated in Figure 4.1 and Figure 4.2, respectively.

I focus on the most commonly used kernel functions, classified based on the mathematical properties of their respective influence functions into the following categories

- **non-robust** includes functions that offer no resistance to outliers, such as the $L2$ loss.
- **robust-monotonous** includes functions like the Fair kernel, which reduce the influence of large residuals in a gradual, monotonic manner.
- **robust-soft-redescending** includes functions like the Welsh, GM, and Cauchy kernels, which allow the influence of residuals to decrease smoothly and eventually plateau as residuals grow larger.

Table 4.1: Robust kernels used in this thesis with their loss function $\rho(\cdot)$, influence function $\Psi(\cdot)$, and weighting function $w(\cdot)$. The parameter c is the tuning parameter that controls the trade-off between the robustness and efficiency of the M-estimator.

Kernel \ Function	$\rho(x)$	$\Psi(x)$	$w(x)$
Fair	$c^2 \left[\frac{ x }{c} - \log \left(1 + \frac{ x }{c} \right) \right]$	$\frac{x}{1 + x /c}$	$\frac{1}{1 + x /c}$
Cauchy	$\frac{c^2}{2} \log(1 + (x/c)^2)$	$\frac{x}{1 + (x/c)^2}$	$\frac{1}{1 + (x/c)^2}$
Geman-McClure (GM)	$\frac{(c^2 x^2)/2}{c^2 + x^2}$	$\frac{c^4 x}{(c^2 + x^2)^2}$	$\left(\frac{c^2}{c^2 + x^2} \right)^2$
Welsch	$\frac{c^2}{2} [1 - \exp(-(\frac{x}{c})^2)]$	$x \exp(-(\frac{x}{c})^2)$	$\exp(-(\frac{x}{c})^2)$
Huber $\begin{cases} \text{if } x \leq c \\ \text{if } x \geq c \end{cases}$	$\begin{cases} x^2/2 \\ c(x - c/2) \end{cases}$	$\begin{cases} x \\ c \cdot \text{sgn}(x) \end{cases}$	$\begin{cases} 1 \\ c/ x \end{cases}$
Tukey $\begin{cases} \text{if } x \leq c \\ \text{if } x \geq c \end{cases}$	$\begin{cases} c^2/6(1 - [1 - (x/c)^2]^3) \\ c^2/6 \end{cases}$	$\begin{cases} x[1 - (x/c)^2]^2 \\ 0 \end{cases}$	$\begin{cases} [1 - (x/c)^2]^2 \\ 0 \end{cases}$

- **robust-hard-redescending** includes functions like the Huber and Tukey kernels, where the influence of residuals decreases to zero beyond a certain threshold, effectively ignoring extreme outliers.

The details of each kernel function are provided in Appendix D.

4.3.3 Influence Analysis

Huber and Roncetti [162] summarizes the usage of the influence function: it assesses the relative influence of individual observations on an estimate or test statistic. If unbounded, outliers can cause issues. In addition, the influence curve provides a quick and intuitive way to assess the asymptotic properties of an estimator, often offering a heuristic basis for guessing its asymptotic variance, which other methods can rigorously verify.

Given the influence functions, we can derive three necessary quantities, as illustrated in Figure 4.3, to characterize the kernel function's behavior in robust estimation [174]:

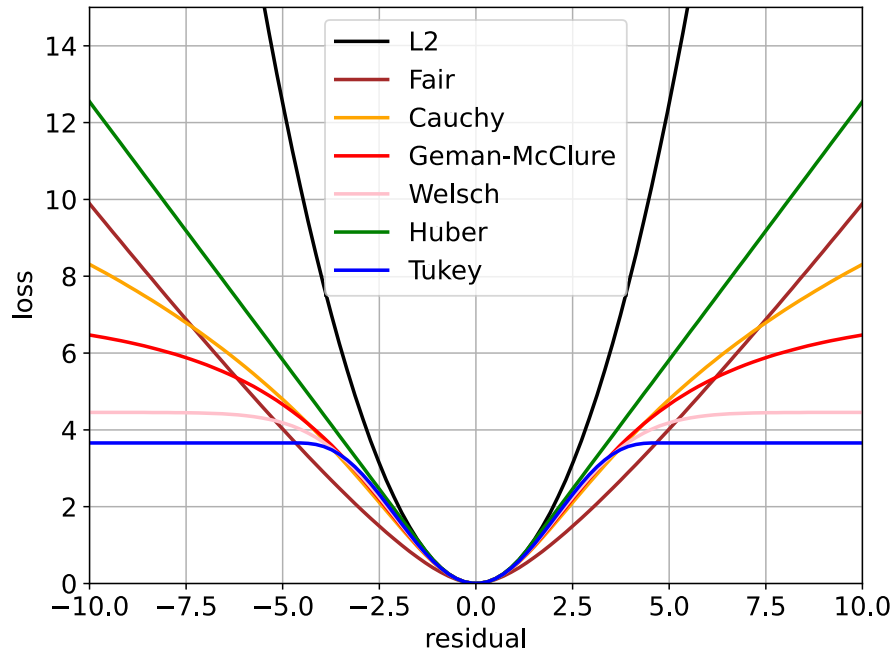


Figure 4.1: Plot of loss values using different kernel functions at $E_{\text{ff}} = 95\%$.

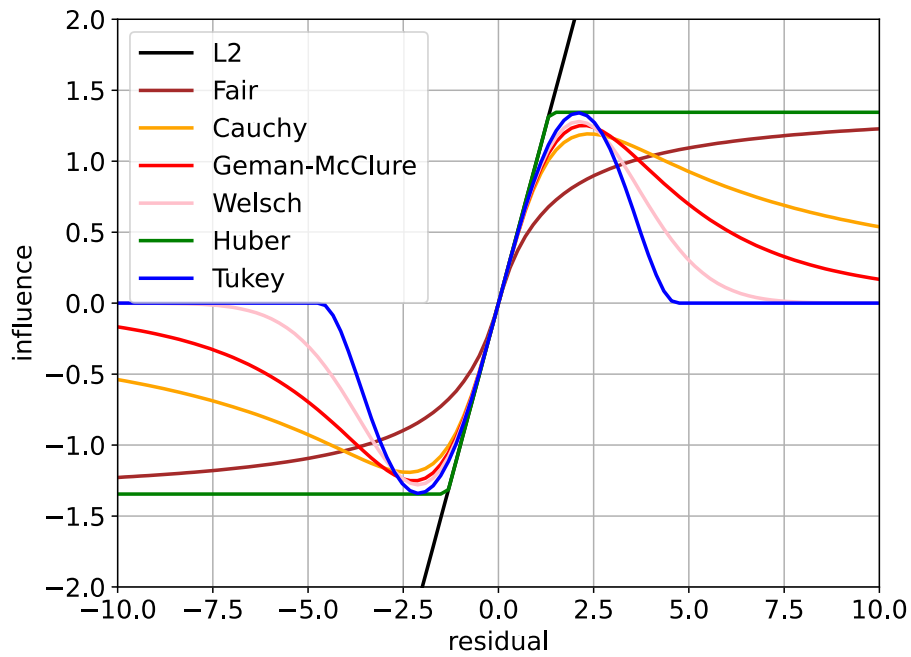


Figure 4.2: Plot of influence values using different kernel functions at $E_{\text{ff}} = 95\%$.

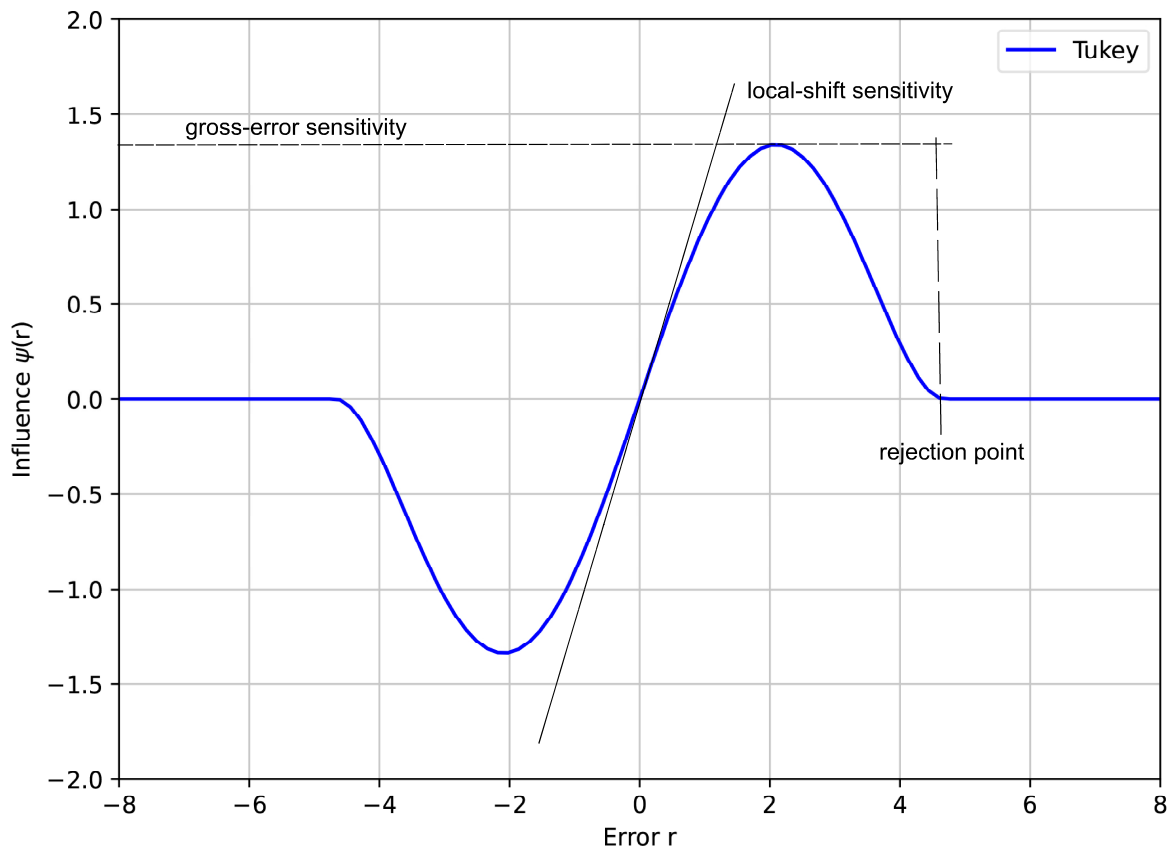


Figure 4.3: Analytical quantities of the Tukey estimator.

- **Gross-error sensitivity** γ : The maximum absolute value of the influence function, γ , measures the worst-case impact that a small number of outliers of fixed size can have on the estimator. It represents the upper bound on the estimator's asymptotic bias. Typically, there is a trade-off between achieving low gross-error sensitivity and maintaining high efficiency or low asymptotic variance.
- **Local change sensitivity** λ : Defined as the slope of the influence function at a specific point, λ quantifies the worst (approximate and standardized) effect of "wiggling" an observation slightly from one point x to a neighboring point y .
- **Rejection point** r_p : The point where the influence function descends to zero, indicating that outliers beyond this point have no effect on the estimator. This is a defining feature of hard-re-descending estimators, which completely ignore extreme outliers.

4.3.4 Breakdown Point

The concept of the Breakdown Point (BP) was introduced by Hampel et al. [174] to evaluate the quantitative global robustness of an estimator. The breakdown point of an estimator θ is the maximum proportion of contamination (percentage of atypical points) in the data that the estimator can tolerate while still retaining meaningful information about the parameter θ . It represents the smallest percentage of gross errors that can cause the estimator to fail entirely.

There must intuitively be more "typical" than "atypical" points for reasonable estimators, meaning the highest possible BP is less than 0.5. For example, the BP of the arithmetic mean is 0, while the BP of the median is 0.5. While the BP measures quantitative robustness, it is distinct from qualitative robustness, though the two are closely related.

Empirically, linear extrapolation using the influence function provides a reasonably accurate estimate of robustness for M-estimators in simple location models. It is exceptionally reliable for contamination levels up to one-quarter of the BP and remains usable for levels up to half of the BP. By quantifying the robustness of an estimator, the breakdown point is a key metric for assessing its reliability and performance in the presence of outliers.

4.4 Kernel Evaluation

In this section, I present the results of kernel evaluations using the selected kernel functions with hyperparameters tuned for different efficiencies. The data from Aachen City is utilized for this analysis, as it provides straightforward scenarios with sufficient satellite observations (four GPS satellites received), enabling a clear assessment of kernel performance.

4.4.1 Experiment Setup

I divide the data from Aachen City into four sequences, as shown in Table 4.2. For each sequence, the proposed state estimator based on continuous-time factor graph optimization (described in Chapter 3) is used to fuse IMU data with raw GNSS pseudorange and Doppler-shift observations. In the factor graph formulation, the original residual term $e_{s,i}^{\text{PrDo}}$ associated with the GNSS pseudorange and Doppler measurements is replaced by the M-estimator cost function (4.4) in the overall optimization objective (3.36). I solve the resulting nonlinear optimization problem iteratively using the Gauss–Newton–based solvers introduced in Appendix A.2.8.

I conduct experiments for each M-estimator across all test sequences. Table 4.3 presents the tuning hyperparameters c for all kernel functions used in the experiments. The hyperparameters of all M-estimators are configured based on their relative efficiency, calculated using

Table 4.2: Sequence definition containing four scenarios in the dataset collected in Aachen.

Sequence	Type	Location	Start	End	Specification
1	Open-Sky	Campus Melaten	0 s	150 s	no GNSS blockage
2	Urban	Kaiser Platz	750 s	980 s	heavy GNSS blockage
3	Sub-Urban	Europaplatz	1200 s	1400 s	slight GNSS blockage
4	Sub-Urban	Eurogress	1430 s	1580 s	slight GNSS blockage
5	Total route	-	0 s	2240 s	-

Table 4.3: Configured tuning parameter c of six M-estimators at four efficiencies.

Kernel	Efficiency	95%	90%	85%	80%	Kernel Category
Fair		1.3998	0.6351	0.3333	0.1760	monotonous
Cauchy		2.3849	1.7249	1.3737	1.1385	soft-redescending
Geman-McClure (GM)		3.8557	2.8937	2.5731	2.2926	soft-redescending
Welsch		2.9846	2.3831	2.0595	1.8383	soft-redescending
Huber		1.3450	0.9818	0.7317	0.5294	hard-redescending
Tukey		4.6851	3.8827	3.4437	3.1369	hard-redescending

(4.6), which leverages the asymptotic variances relative to the reference noise model (i.e., Gaussian). Tuning the M-estimators based on their efficiency ensures a balanced trade-off between robustness and accuracy.

Instead of identifying the best kernel function and hyperparameters, the primary goal of the experiments is to gain a comprehensive understanding of the effectiveness of various kernel functions operating at different efficiencies, particularly in the context of range-based vehicle localization. Given this knowledge, I am equipped to address research question **Q2** and validate my hypotheses concerning the application and effectiveness of M-estimation.

4.4.2 Essential Experimental Results

The error metrics of the mean 2-D localization errors in my experiments are presented in Table 4.4. Recalling the definitions of robustness and efficiency, the error metrics reveal that the robustness of all M-estimators improves as efficiency decreases. Consequently, the positioning error decreases, provided that sufficient observations remain available to prevent the state estimator from diverging.

Figure 4.4 illustrates the GNSS observation weights assigned by all M-estimators at different efficiency levels for the entire route. As robustness increases, more sensor observations are de-weighted, potentially leading to insufficient constraints on the state estimator and indirectly

4 Handling Outliers using Classic M-Estimators

Table 4.4: Mean 2-D error of all sequences is presented. Values marked in red indicate the worst accuracy, while those in green highlight the best results.

Eff.	Kernel	Seq.	#1 (Open Sky)	#2 (Urban)	#3 (Light-Urban)	#4 (Light-Urban)	#5 (Total Route)
95%	Fair		0.35	4.09	1.74	3.08	2.51
	Cauchy		0.35	2.12	1.05	1.60	1.38
	GM		0.50	0.67	0.38	0.78	0.99
	Welsch		0.47	NaN	NaN	NaN	3.75
	Huber		0.38	3.26	1.62	2.77	2.09
	Tukey		0.38	0.82	0.95	0.69	1.11
90%	Fair		0.35	2.97	1.42	2.78	2.02
	Cauchy		0.22	0.76	0.44	2.26	1.10
	GM		0.30	1.80	0.41	1.80	1.20
	Welsch		0.50	0.50	NaN	NaN	0.99
	Huber		0.35	2.78	1.33	2.54	1.91
	Tukey		0.38	0.82	0.95	0.69	1.11
85%	Fair		0.30	1.90	1.19	2.18	1.59
	Cauchy		0.45	1.13	0.51	0.98	1.07
	GM		0.31	1.05	1.25	0.64	1.15
	Welsch		0.38	0.82	0.95	0.69	1.11
	Huber		0.34	2.30	1.37	2.45	1.75
	Tukey		0.38	0.82	0.94	0.69	1.11
80%	Fair		0.34	1.72	1.05	1.84	1.41
	Cauchy		0.44	0.73	0.53	0.92	1.04
	GM		0.36	0.47	0.69	0.46	1.09
	Welsch		70.38	NaN	NaN	NaN	70.38
	Huber		0.31	1.58	1.25	2.08	1.49
	Tukey		0.38	0.82	0.95	0.69	1.11

reducing localization accuracy. This conclusion is supported by the positioning accuracy in open-sky areas (Seq. #1), where fewer constraints result in a noticeable decline in performance despite the absence of significant outliers.

Similarly, in urban areas, M-estimators with higher robustness do not necessarily outperform the same estimators configured for higher efficiency. This is evident when comparing results at 90 % efficiency to those at 80 % efficiency. The underlying reason for this result lies in the extensive discarding of GNSS observations at lower efficiencies. While this may enhance the consistency of the estimation by mitigating the impact of outliers, it can also degrade accuracy due to the reduced availability of valid measurements.

In the following, I analyze and interpret the experimental results of all M-estimators, categorized according to the properties of their respective influence functions. To maintain clarity and focus, additional experimental results are provided in Appendix D.

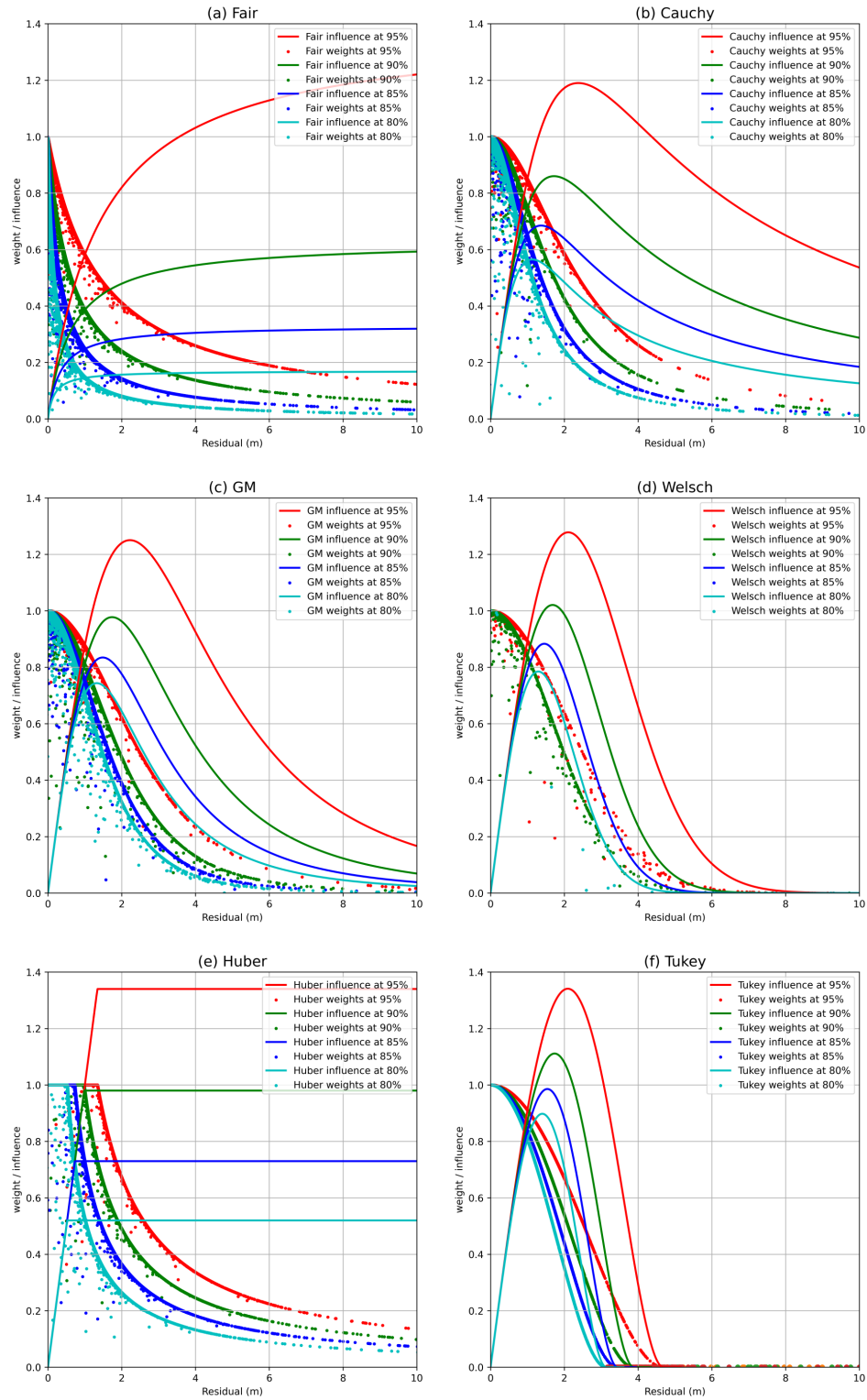


Figure 4.4: Plot of all kernels' influence function and **unwhitened** residual weights. As the loss functions are symmetric, I only plot the weights and kernel influence for positive residuals.

4.4.2.1 Monotonous M-Estimator

As introduced earlier, monotonous M-estimators, such as the M-Fair estimator, present monotonically decreased influence values. Compared to other estimators, the influence function of the Fair estimator increases rapidly for small errors and grows more slowly for errors exceeding its tuning parameter. As a result, the Fair estimator significantly downweights observations that are only slightly corrupted. This characteristic leads to reduced positioning accuracy, as evident in the 2-D error plots from Figure 4.5 to Figure 4.8, where the Fair estimator consistently demonstrates worse performance across all experiments.

Additionally, the Fair's influence function does not descend to zero, which means outliers still exert some influence on the estimates, thereby reducing its robustness to, for example, hard-redescending estimators like Tukey's.

In summary, the Fair estimator fails to achieve an optimal trade-off between robustness and accuracy in range-based vehicle localization approaches, as its tendency to overly downweight slightly corrupted observations negatively impacts positioning performance.

4.4.2.2 Soft-Redescending M-Estimator

Unlike the Fair estimator, soft-redescending M-estimators, such as the Cauchy, Geman-McClure (GM), and Welsch, exhibit influence functions with distinct characteristics. These estimators rapidly increase the loss for small measurement errors, then reduce the growth as the error approaches the tuning parameter. For errors exceeding the tuning parameter, the influence decreases smoothly, allowing for continuous and gradual downweighting of large residuals. This property allows soft-redescending M-estimators to handle outliers more effectively while maintaining stability in the presence of significant measurement errors.

Therefore, soft-redescending M-estimators generally present better performance and well-balanced robustness, so that the positioning errors are smaller at all efficiencies compared to other M-estimators, as shown in Table 4.4.

However, the Welsch estimator, unlike the other two, demonstrated instability during my experiments, leading to divergence in the state estimation. A possible explanation for this observation is the Welsch estimator's exponential weighting function, which may introduce numerical instability, particularly when measurement outliers dominate the received data. This behavior is corroborated by the results in Table 4.4, which show that the Welsch estimator is more prone to failure in urban areas compared to open-sky areas. To further support this explanation, the geometric shapes of all loss functions at different efficiency levels are illustrated in Figures D.1, D.2, and D.3. These figures show that, although the Welsch estimator is formally classified as a soft-redescending kernel, its loss profile closely resembles that of the Tukey estimator from the hard-redescending category, providing insight into its similar failure characteristics.

The Cauchy and GM estimators, on the other hand, effectively down-weight measurements that are more likely to be outliers. As shown in Figure 4.4, even smaller (unwhitened) residuals are assigned lower weights, as they may be corrupted by a larger variance. It can also be observed that these estimators maintain a balanced response to large residuals, avoiding excessive down-weighting of measurements. Compared to the Fair or Huber estimators, soft-redescending estimators produce fewer large residuals, thereby reducing the likelihood of true-negative outliers being incorrectly identified (see Figure 4.4).

Another interesting observation with the soft-redescending estimators aligns with my earlier conclusion: tuning these estimators for lower efficiency does not necessarily lead to improved performance.

In summary, the Cauchy and GM estimators generally achieve lower positioning errors, as shown in Figures 4.5–4.8. This makes them more suitable for range-based vehicle localization approaches.

4.4.2.3 Hard-Redescending M-Estimator

Although hard-redescending estimators, such as the Huber estimator, are widely used in various applications, they do not perform as effectively as soft-redescending estimators in the specific applications.

Because the influence of residuals in hard-redescending estimators decreases to zero once the residuals exceed the tuning parameter, measurements with large residuals are effectively ignored. This behavior is particularly evident in the Tukey estimator, where such measurements are entirely disregarded (see Figure 4.4). It is beneficial only if the tuning parameter is consistently and appropriately selected.

Using a suboptimal tuning parameter in these estimators can lead to overly aggressive outlier handling, potentially resulting in the exclusion of valid measurements. This is a key reason why the positioning error of the Tukey estimator does not surpass that of the soft-redescending estimators. However, since many outliers—especially those with inconsistent noise models—cannot be perfectly identified in real-world applications, using a hard-redescending estimator poses significant challenges in delivering consistent performance over long-term and large-scale environments.

Compared to the Tukey estimator, the Huber estimator does not exhibit superior properties for handling outlier measurements in range-based localization. Unlike the Tukey estimator, the Huber loss cannot directly "reject" outlier measurements. Instead, when measurement residuals exceed the tuning parameter, the Huber estimator behaves similarly to the Fair estimator, gradually down-weighting potential outliers rather than eliminating their influence outright.

4 Handling Outliers using Classic M-Estimators

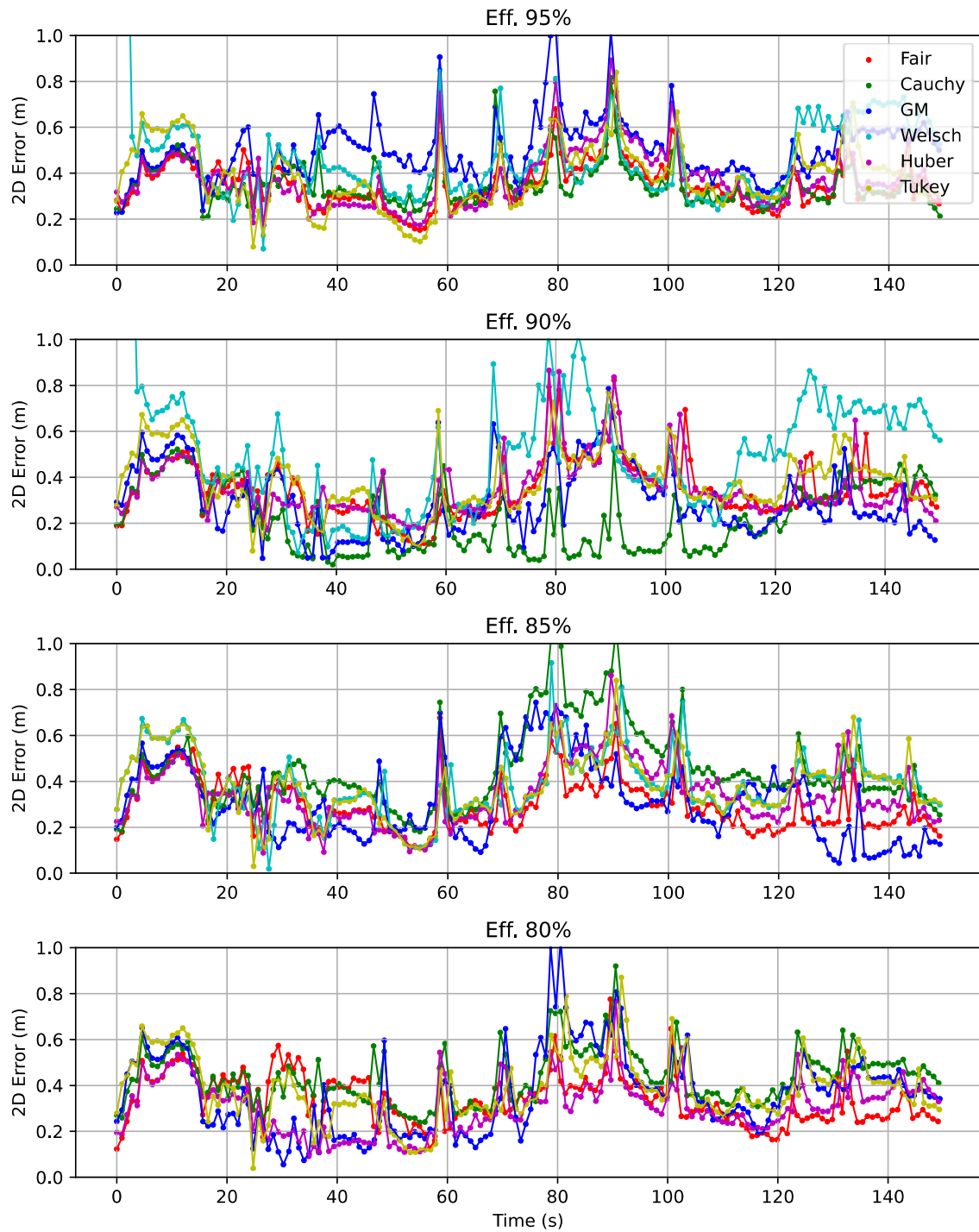


Figure 4.5: Plot of 2D error of sequence #1 in open-sky areas using different M-estimators. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

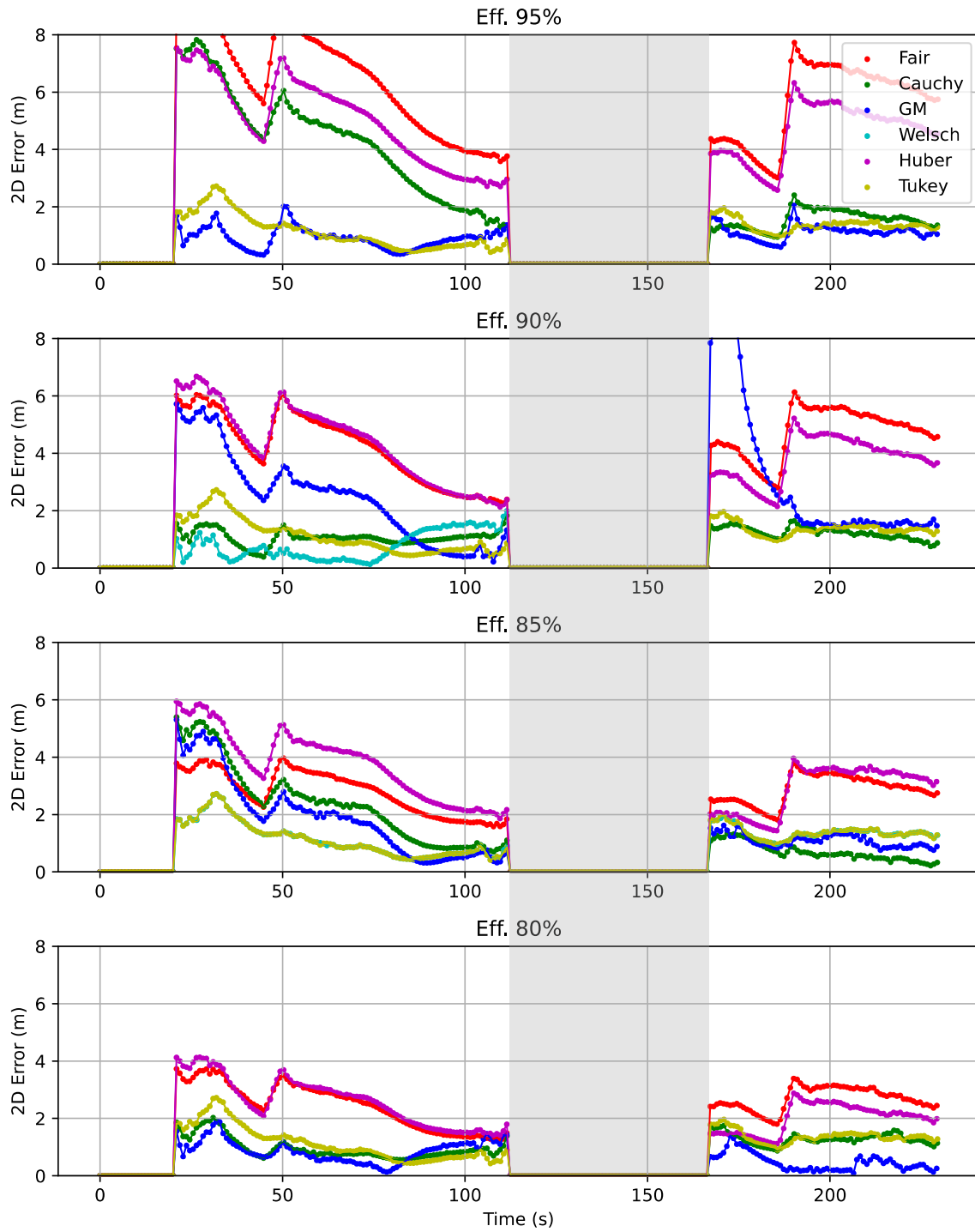


Figure 4.6: Plot of 2D error of sequence #2 in urban areas using different M-estimators. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

4 Handling Outliers using Classic M-Estimators

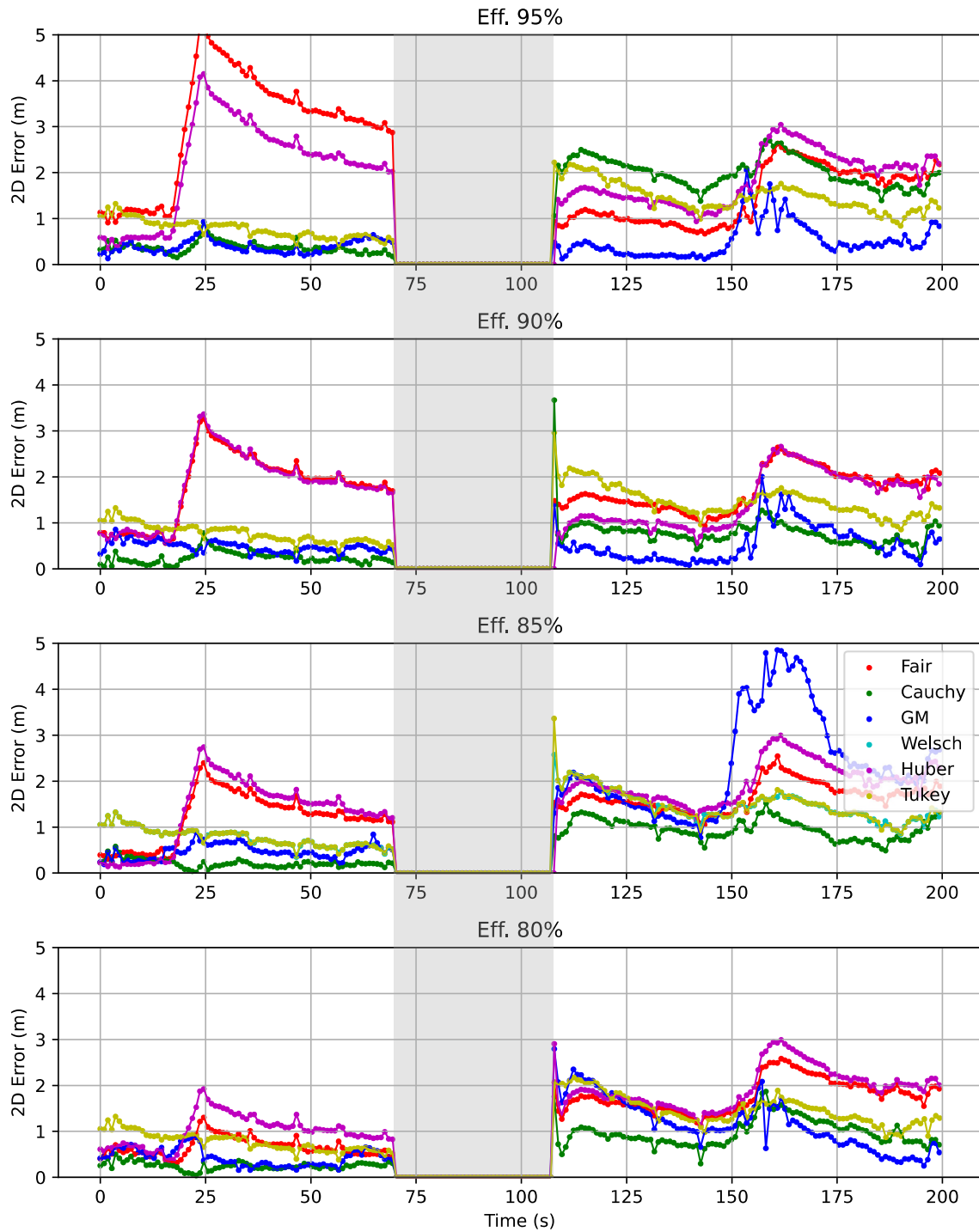


Figure 4.7: Plot of 2D error of sequence #3 in light-urban areas using different M-estimators. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

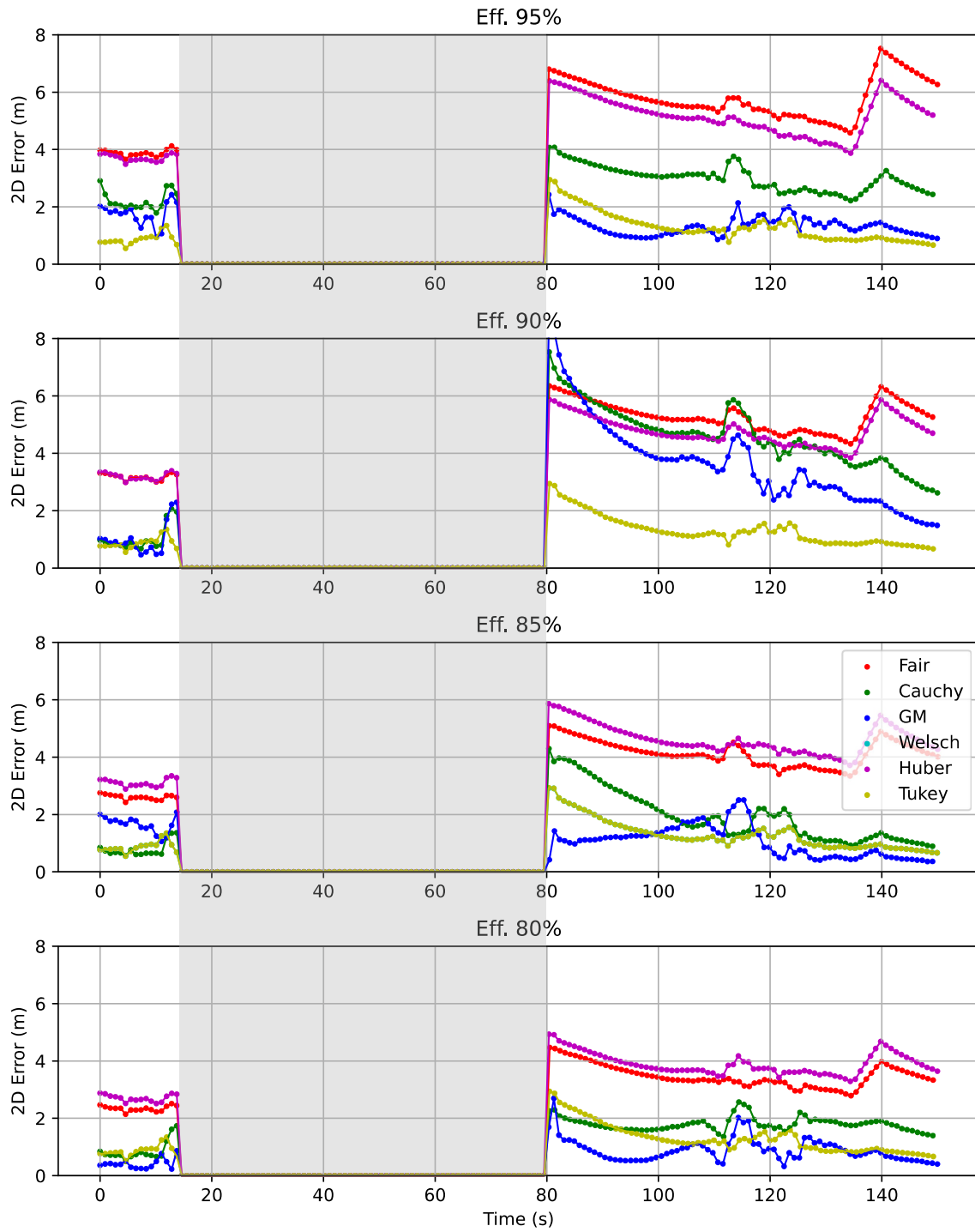


Figure 4.8: Plot of 2D error of sequence #4 in light-urban areas using different M-estimators. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

4.4.2.4 Soft- versus Hard-Redescending M-Estimators

Finally, I draw several conclusions from the experimental comparison between soft- and hard-redescending M-estimators, particularly in the context of handling GNSS outliers.

Generally, hard-redescending estimators assume a clear separation between inliers and outliers, completely suppressing the influence of measurements whose residuals exceed a predefined threshold. This behavior is well-suited to scenarios dominated by sporadic, uninformative **gross errors**. However, in GNSS-based localization applications, measurement errors in harsh environments (e.g., urban) arising from multipath and non-line-of-sight propagation are typically structured and biased rather than arbitrarily corrupted. Such measurements often retain partial geometric information and exhibit residual magnitudes that overlap substantially with those of nominal observations, violating the assumptions underlying hard rejection. As a result, hard-redescending estimators tend to enforce a conservative robustness–efficiency trade-off, prematurely discarding measurements that may still be informative. This characteristic explains the results reported in Table 4.4, where hard-redescending estimators do not outperform alternative robust costs in terms of accuracy, yet consistently contribute to stable estimation by preventing divergence.

In this regime, soft-redescending M-estimators, such as Cauchy and Geman–McClure, offer a more appropriate mechanism, which progressively downweights increasing residuals without enforcing a binary inlier–outlier decision. This allows moderately corrupted GNSS measurements to contribute to the estimation process with reduced influence, while preventing large residuals from dominating the solution. As a result, these estimators achieve a more favorable balance between robustness and information retention, particularly in scenarios with limited satellite visibility or weak geometric diversity.

When applied to state estimation problems using radio-based ranging observations, the strong empirical performance of the Cauchy and Geman–McClure M-estimators can be attributed to the heavy-tailed nature of GNSS residual distributions in challenging environments. Rather than forming a distinct class of extreme outliers, GNSS errors often follow a distribution with a Gaussian core and long tails induced by persistent propagation effects. soft-redescending estimators are well-suited to this structure, as they implicitly model heavy-tailed noise and remain effective across a wide range of residual magnitudes. Consequently, the strong empirical performance of these estimators suggests that GNSS outliers exhibit the same statistical characteristics commonly encountered in robotic perception and localization, and do not necessitate specialized or atypical robust cost formulations.

In summary, aligned with trends observed across robotic applications, the Cauchy and Geman–McClure estimators serve as strong baselines when studying advanced M-estimators for handling GNSS outliers.

CHAPTER 5

Handling Outliers using Learning-based Methods

“All models are wrong, but some are useful.”

– George E. P. Box

5.1	Motivation and Objective	93
5.2	Related Work	94
5.3	Dataset Generation and Feature Selection	95
5.3.1	Dataset Generation	95
5.3.2	Feature Selection	97
5.4	Problem Formulation	101
5.5	Classic Machine Learning Models	101
5.6	Deep Learning Models	102
5.6.1	Reference Model	102
5.6.2	Proposed Model	103
5.6.3	Loss Function of the Proposed Model	105
5.7	Experiment Design and Results	106
5.7.1	Performance Metrics of NLOS Classification	106
5.7.2	Performance of Pseudorange Error Prediction	110
5.7.3	Vehicle Localization with GNSS Fault Exclusion	114

5.1 Motivation and Objective

From my studies in the previous chapter, it is evident that approaches utilizing M-estimators to handle measurement outliers cannot explicitly exclude faulty GNSS measurements, and therefore, do not fully guarantee stability and robustness. This conclusion also applies to other classic methods relying on consistency checking with redundant observations, which often cannot be satisfied in urban areas, leading to conservative fault classifications

[175]. Inspired by Zhang et al. [90] and Lim et al. [176], I consider the GNSS observations as time-series data and propose a transformer-enhanced LSTM network for NLOS classification and pseudorange error prediction in this chapter.

Recalling the research question **Q3** in Section 1.4.2, this chapter explores how learning-based methods can be employed to handle GNSS measurement outliers, supported by comprehensive experimental studies. The contributions are summarized as follows

1. I propose a novel neural network architecture to predict measurement outliers, taking into account the noise characteristics of GNSS observations in complex environments and incorporating both temporal and spatial information into the feature extraction process.
2. Beside the available dataset from Hong Kong city, I propose a dataset generation process utilizing lidar maps to directly identify signal blockages of received satellites in 3-D space.
3. I implement and train the proposed model alongside a baseline LSTM network [90] and two classical machine learning models: support vector machine (SVM) and extreme gradient boosting (XGBoost). This setup aims to provide an in-depth analysis comparing classical models with deep networks, as well as evaluating performance on in-distribution and out-of-distribution data. I also conduct ablation studies on the proposed network components and present comprehensive discussions on model performance, dataset quality, and the importance of features.
4. I demonstrate the outlier detection performance in a real-world vehicle localization scenario by leveraging pre-trained models to evaluate unseen data from a different city.

The essential technical backgrounds of learning-based methods are presented in Chapter A, Section A.4, while detailed information about the datasets used for training and testing is provided in Chapter 2, Section 2.4.3.

5.2 Related Work

Learning-based methods have gained significant attention for NLOS classification and multipath error prediction [177]. Pre-trained models, developed through intensive data augmentation and offline training, can be effectively used online for GNSS fault detection.

Several studies have employed classical machine learning methods, such as fuzzy logic [178] and support vector machines (SVM) [179, 180], to classify faulty GNSS measurements, achieving impressive classification accuracy on test data. Given the time-correlated and strongly nonlinear nature of faulty GNSS observations, deep-learning-based (DL) methods have emerged as a promising alternative. These methods can extract semantic features and provide better generalization performance using pre-trained models [181–183].

Among these approaches, Zhang et al. proposed a specific network architecture combining fully connected layers and long short-term memory (LSTM) networks for satellite visibility classification and pseudorange error prediction [90]. In Bai et al. [184], this pre-trained network was integrated into a state estimator to exclude NLOS observations, demonstrating notable accuracy improvements. While LSTM networks are effective for modeling sequential data [176], they face limitations when dealing with multivariate features and contextual information [185].

To address these challenges, the self-attention mechanism introduced in transformers offers a promising solution, enhancing performance in sequential data modeling by effectively capturing dependencies and relationships across the input [186].

However, although the abovementioned studies have demonstrated the potential of learning-based methods to handle outliers in GNSS measurements, none of them has thoroughly examined the feasibility and limitations of these methods in real-world applications or their generalization across different datasets and environments.

5.3 Dataset Generation and Feature Selection

Training supervised learning models requires ground-truth labels corresponding to the learning target, serving as the desired output values for the input data. The precision of these labels directly impacts the model's prediction accuracy, with higher-quality labels enabling more reliable and accurate predictions.

This section proposes a framework that leverages 3-D point clouds to generate precise labels for the extracted input time-sequence data. Additionally, I introduce the selected features used in the training and test datasets and provide details on the data augmentation process. Figure 5.1 summarizes the proposed dataset generation process.

5.3.1 Dataset Generation

Figure 5.1 summarizes the proposed dataset generation process. In this process, I collected raw data in a measurement campaign in Aachen. The raw data details are introduced in Application II (see Section 2.3.2).

To obtain pseudorange residuals and signal-blocking masks, I utilize the multi-sensor fusion state estimator and GNSS pre-processor introduced in Chapter 3. The pre-processed GNSS observations, combined with measurements from an IMU and a lidar, are fed into the state estimator. This process computes the pseudorange residuals and generates 3-D lidar maps (see Figure 5.2), which are subsequently used to identify satellite blockages in sky plots, enabling label generation. Figure 5.3 shows the masked sky plots. While generating labels, I query the

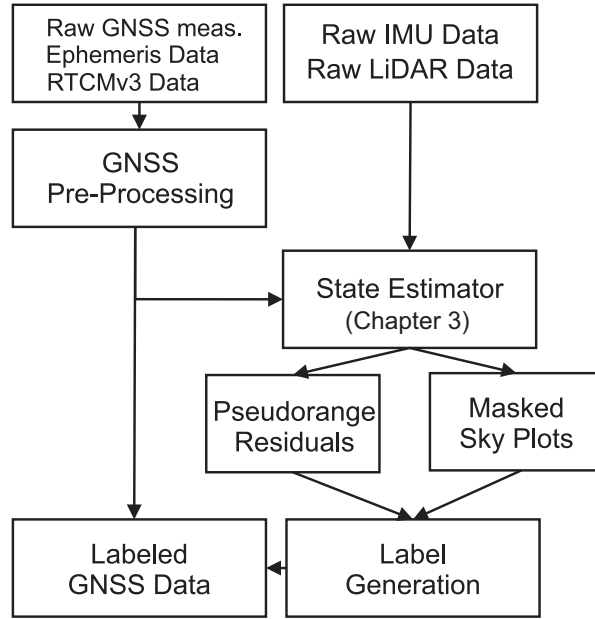


Figure 5.1: Proposed dataset generation process (reproduced from [M3] with permission of IEEE).

neighbored elevation and azimuth angles at each satellite and infer the satellite with NLOS reception if the elevation/azimuth of the satellite is masked according to and the pseudorange residual is above a predefined threshold (e.g., $|e| > 15$ m).

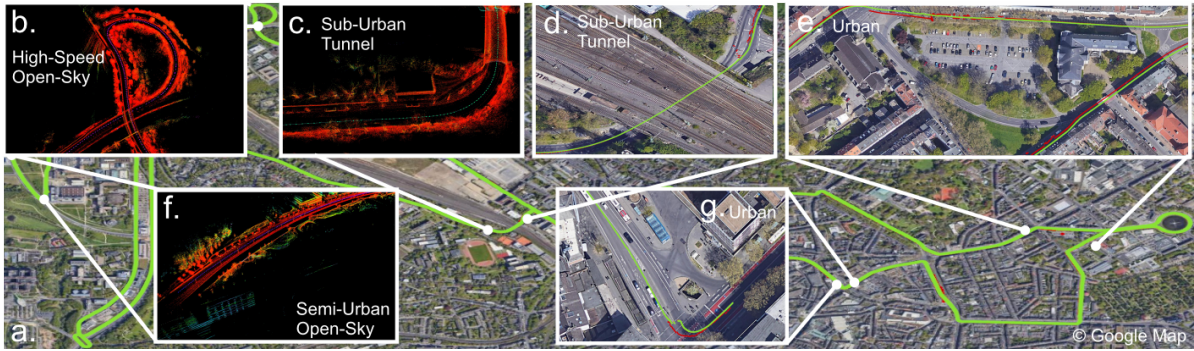


Figure 5.2: Generated lidar Map in Aachen City.

Inferring satellite visibility labels requires extracting a set of point clouds surrounding the vehicle at each timestamp. However, a single point cloud sampled at one timestamp cannot accurately represent the surrounding environment due to limitations in the number of laser emissions and their elevation range. This issue arises when surrounding objects that block or reflect satellite signals have elevation angles exceeding the upper limit of the lidar's laser emission range. Therefore, I incorporate point clouds from multiple timestamps to label the satellite visibility.

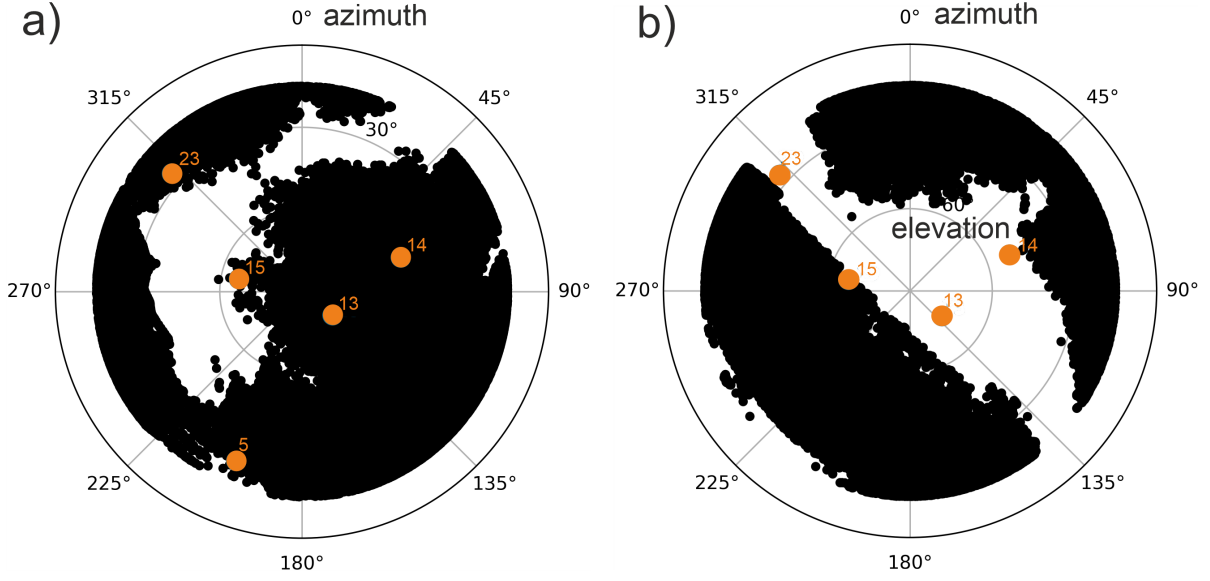


Figure 5.3: Sky plots from two scenes: a) urban tree-rich area, b) urban. The received GPS satellites are marked in orange (reproduced from [M3] with permission of IEEE).

As shown in Figure 5.4, incorporating additional time steps of point clouds leads to a more accurate reconstruction of the surrounding environment. Specifically, potential obstacles are represented with more data points, increasing the likelihood of accurately detecting signal obstructions.

5.3.2 Feature Selection

Given raw GNSS observations, navigation messages, and related vehicle positions in the global frame, multiple features can be calculated to encode geometric information, contextual details, and the quality of signal propagation from the received satellites. Following previous work [90], I select the same five features¹ from the pre-processed GNSS observations to train the proposed model for NLOS classification and pseudorange error prediction.

Additionally, I incorporate two extra features into the training process that were not utilized in the benchmark method, allowing for a more comprehensive investigation of model performance. These features are described as follows.

¹Because the HK dataset does not provide the required raw data (e.g., carrier phase and Doppler-shift) to calculate the *CMC* and *DRC*, I formulate two feature sets: the 5-feature and 6-feature sets to analyze the model's performance using different features. The 5-feature set of k -th satellite is defined as $\mathcal{F}_5 = \{El_\tau^k, Az_\tau^k, C/N_{0\tau}^k, \sigma_{LS,\tau}^k, RSS^k\}$, and the 6-feature set is given as $\mathcal{F}_6 = \{El_\tau^k, Az_\tau^k, C/N_{0\tau}^k, \sigma_{LS,\tau}^k, CMC_\tau^k, DRC_\tau^k\}$.

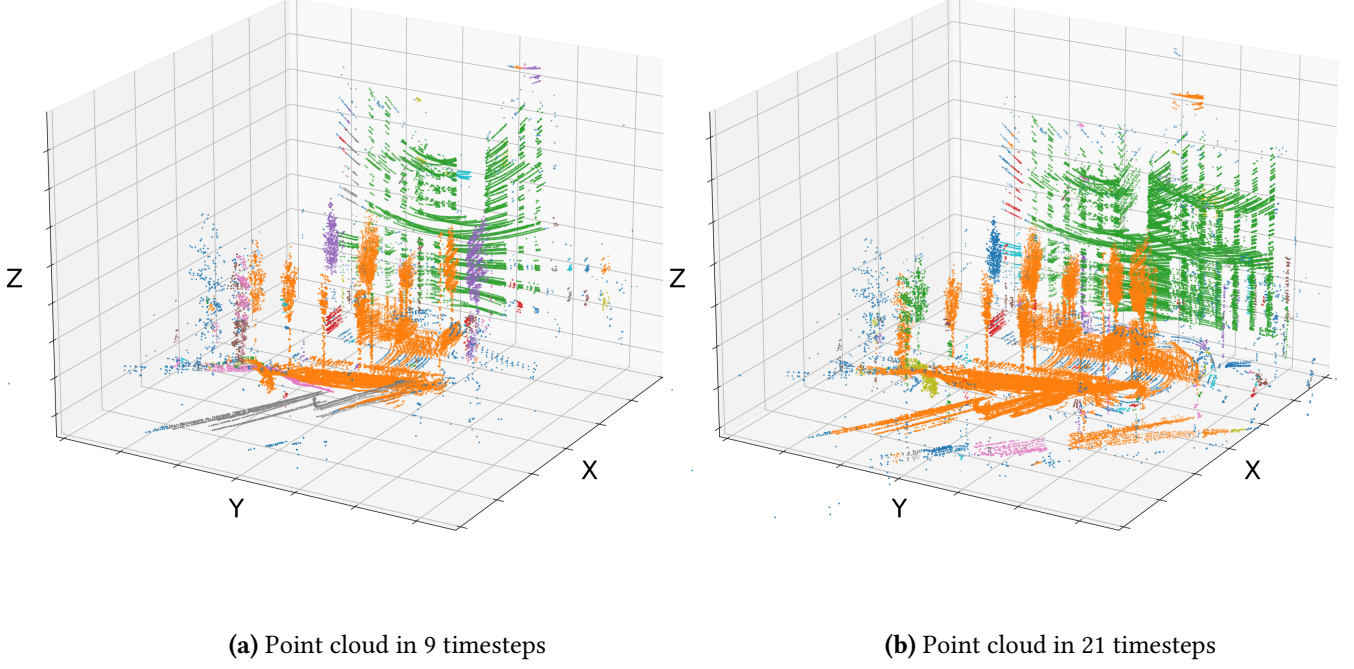


Figure 5.4: lidar point clouds collected in different time intervals.

5.3.2.1 Elevation Angle

The satellite elevation angle El provides an adequate quality measure of the satellite visibility. Satellites with higher elevation angles are generally less affected by the infrastructure objects. In classical GNSS positioning approaches, satellites with low elevation angles are excluded from state estimation algorithms.

The elevation angle El_k of k -th satellite can be calculated as

$$El^k = \arcsin(h^{\text{sat},k}/r^k), \quad (5.1)$$

where the satellite's height with respect to the vehicle in the ENU frame is represented as $h^{\text{sat},k}$. The variable r^k denotes the geometric range from the vehicle to the k -th satellite.

5.3.2.2 Azimuth Angle

The satellite azimuth angle represents the horizontal geometric relation between the satellite and the vehicle. As described in Zhang et al. [90], azimuth angles enable the deduction of satellite visibility when multiple satellites with similar azimuth angles but different elevation

angles are received. This feature is expected to provide more information on the geometric context in learning-based methods.

We can calculate the azimuth angle with

$$Az^k = \arctan(x^{\text{sat},k}/y^{\text{sat},k}), \quad (5.2)$$

where the symbols $x^{\text{sat},k}$ and $y^{\text{sat},k}$ are the positions of k -th satellite relative to vehicle's position in the ENU-frame.

5.3.2.3 Carrier-to-Noise-Ratio

The Carrier-to-Noise Ratio C/N_0 measures the satellite signal strength and characterizes the satellite signal quality. Generally, the GNSS observations used for positioning are expected to have higher values C/N_0 . In urban scenarios, C/N_0 can also effectively represent interference due to NLOS receptions and multipath effects [101].

5.3.2.4 Least-Square Pseudorange Residual

In many GNSS standalone solutions, vehicle position can be calculated with an iterative least-square (LS) method using GNSS observations. I use this positioning solution to calculate pseudorange residuals, denoted as $\sigma_{\text{LS}}^{\text{sat},k}$. Unlike the pseudorange residuals calculated in the state estimator by combining multiple sensors, the residual $\sigma_{\text{LS}}^{\text{sat},k}$ is highly related to the quality of GNSS observations and represents the scale of pseudorange errors.

We can calculate the LS pseudorange residual by

$$\sigma_{\text{LS}}^{\text{sat},k} = \hat{\rho}_k - \|\hat{\mathbf{x}}^r - \mathbf{x}^{\text{sat},k}\|_2, \quad (5.3)$$

where the range $\hat{\rho}_k$ is the measured pseudorange, which is corrected in the GNSS pre-processing process to eliminate the atmospheric delays. The satellite position and the estimated vehicle position are represented as $\mathbf{x}^{\text{sat},k}$ and $\hat{\mathbf{x}}^r$, respectively.

5.3.2.5 Root-Sum-Squares of Pseudorange Residuals

As introduced in Zhang et al. [90], the root-sum-square of the pseudorange residuals encodes the environmental condition by summing up multiple pseudorange residuals in a certain time window $t = 1 \dots T$. If the vehicle is moving in urban areas, larger RSS should be expected.

I employ the RSS, shown in

$$\text{RSS} = \sqrt{\sum_{t=1}^T (\sigma_{\text{LS},t}^{\text{sat},k})^2} \quad (5.4)$$

as the last feature in the feature vector. In this thesis, the time window $t = 1 \dots T$ is treated as a hyperparameter and is set to 1 s corresponding to 10 measurement epochs.

5.3.2.6 Code Minus Carrier signal

The Code Minus Carrier Signal (CMC) was introduced in Gogliettino et al. [187] and further utilized in Zawislak et al. [182] as a feature for learning-based methods. It is derived using the delay-free pseudorange $\hat{\rho}^k$ and the carrier phase Φ_k over a specified time window W , as described in (5.5).

$$\text{CMC}^k = \hat{\rho}_k - \lambda \Phi_k + C_t, \quad (5.5)$$

where the variable λ is the wavelength of the GNSS observation for a target signal (e.g., L1) and the scalar C_t is a balance value such that $\sum_{k=t-W}^t (\hat{\rho}_k - \lambda \Phi_k) = 0$.

Unlike the aforementioned features, the CMC^k represents a relative measure between the pseudorange and carrier phase, making it particularly useful for multipath detection. This is because the pseudorange and carrier phase are affected differently in challenging environments. Consequently, CMC^k can serve as an indicator of transitions between NLOS and LOS conditions or vice versa. In this thesis, I also set the time window W as 1 s corresponding to 10 measurement epochs.

5.3.2.7 Doppler Rate Consistency

Similar to CMC, the Doppler Rate Consistency DRC represents the difference between the Doppler shift and the pseudorange change. This feature is practical for detecting multipath and NLOS signals, as the Doppler shift and pseudorange are typically disturbed differently in challenging environments, as introduced by Suzuki and Amano [180].

The DRC is calculated using

$$\text{DRC}^k = \hat{\rho}_t^k - \hat{\rho}_{t-1,k} - c \cdot \Delta t \cdot \lambda \cdot \nu_{t,k}, \quad (5.6)$$

where the scalar c denotes the speed of light, Δt is the time difference between two measurement epochs, and $\nu_{t,k}$ represents the Doppler shift measurement.

5.4 Problem Formulation

The pseudorange observations are commonly used to represent the vehicle position in GNSS-based localization approaches, as shown in

$$\rho_t^k = r_t^k + c(\delta t_r - \delta t^{\text{sat},k}) + T_t^k + I_t^k + M_t^k + w_{\rho,t}^k, \quad (5.7)$$

$$\hat{\rho}_t^k = r_t^k + \underbrace{M_t^k + w_{\rho,t}^k}_{\text{pseudorange error}}, \quad (5.8)$$

where the range $r_t^k = \|\mathbf{x}_t^r - \mathbf{x}_t^{\text{sat},k}\|$ is the geometric range between the vehicle and k -th satellite. The receiver clock error δt_r and the satellite clock error $\delta t^{\text{sat},k}$, multiplied by the light speed c , represent the clock bias. The variables T_t^k and I_t^k are tropospheric and ionospheric delays, respectively.

In this work, I assume that the receiver clock bias is estimated in the state estimator and therefore is subtracted from (5.7). With the ephemeris and differential correction data, the satellite clock bias and atmospheric delays can be eliminated in a pre-processing process to acquire the corrected pseudorange $\hat{\rho}_t^k$ in (5.8).

I denote the multipath delay and the observation noise using M_t^k and $w_{\rho,t}^k$, respectively. These delays vary particularly in urban areas and become complex and time-related due to NLOS receptions and multipath effects [188]. Therefore, as marked in (5.8), this thesis aims to predict the pseudorange error expressed as the sum of the multipath and noise components.

Therefore, I consider the NLOS detection and error prediction of pseudorange $\hat{\rho}_t^k$ as a spatio-temporal modeling problem

$$[\mathcal{V}^k, \hat{\epsilon}^k] = \text{NN} \left(\left\{ \text{El}_\tau^k, \text{Az}_\tau^k, \text{C/N}_{0\tau}^k, \sigma_{\text{LS},\tau}^k, \text{RSS}^k \right\}_{\tau=1}^T \right), \quad (5.9)$$

which predicts the satellite visibility $\mathcal{V}^k$ and pseudorange error $\hat{\epsilon}^k$ using a neural network given all selected features in a time series of T seconds.

5.5 Classic Machine Learning Models

As shown in Figure 5.5, the classical machine learning pipeline can be divided into two components, which are trained independently to perform LOS/NLOS classification and pseudorange error prediction for satellite signals.

The first component is a classical LOS/NLOS classifier that takes as input a batch of feature vectors extracted from satellite signal measurements. A classical machine learning model, such as XGBoost or an SVM, is employed for this task. After training, the classifier determines

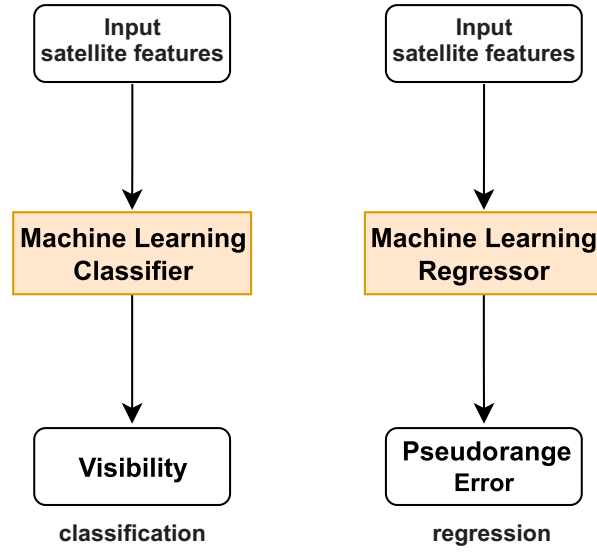


Figure 5.5: Classical Machine Learning Models.

whether a given satellite signal corresponds to a line-of-sight (LOS) or non-line-of-sight (NLOS) reception.

The second component is responsible for predicting the pseudorange error of the received satellite signals. For this task, a classical machine learning regressor, such as XGBoost or an SVM regressor, is used. The regression labels correspond to the pseudorange errors. After training, the model is able to predict the pseudorange error for each received satellite signal.

5.6 Deep Learning Models

5.6.1 Reference Model

This model is a referenced deep learning architecture proposed in [90]. Its prediction performance is used as the baseline for deep learning approaches in this thesis and is analyzed and compared with the proposed model to demonstrate the advantages of the proposed approach.

The architecture leverages both contextual satellite information and target satellite information simultaneously at the same time step for GNSS fault forecasting. In addition to the target satellite features, a set of nine context satellite feature maps (including the target satellite itself) is provided as an additional input to the model.

The target satellite features are processed by a multilayer perceptron (MLP) to extract latent representations, while the context satellite features are fed into an LSTM network to capture

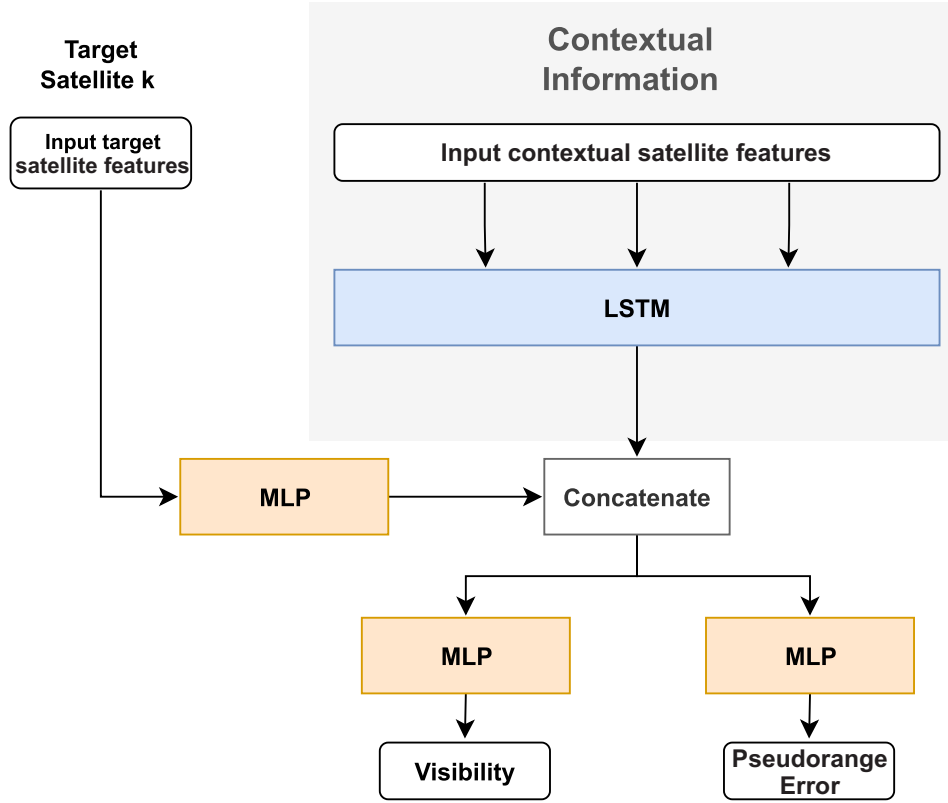


Figure 5.6: Referenced LSTM Architecture.

surrounding satellite information. The resulting feature representations from these two branches are then concatenated to form a unified representation, which is subsequently passed to separate MLP heads for LOS/NLOS classification and pseudorange error regression.

However, the predictions produced by this benchmark model are time-independent. Since both LOS/NLOS classification and pseudorange error prediction are inherently temporally continuous tasks, neglecting temporal dependencies limits the robustness of this approach.

The model outputs two predictions: satellite visibility and pseudorange error. Visibility prediction is formulated as a classification task, where the output is a continuous value that is assigned to the nearest discrete class. Pseudorange error prediction is formulated as a regression task, yielding a continuous estimate of the pseudorange error.

5.6.2 Proposed Model

Considering the noise characteristics of GNSS observations in urban areas, I propose a neural network with three key properties: (1) capturing the temporal variations of selected

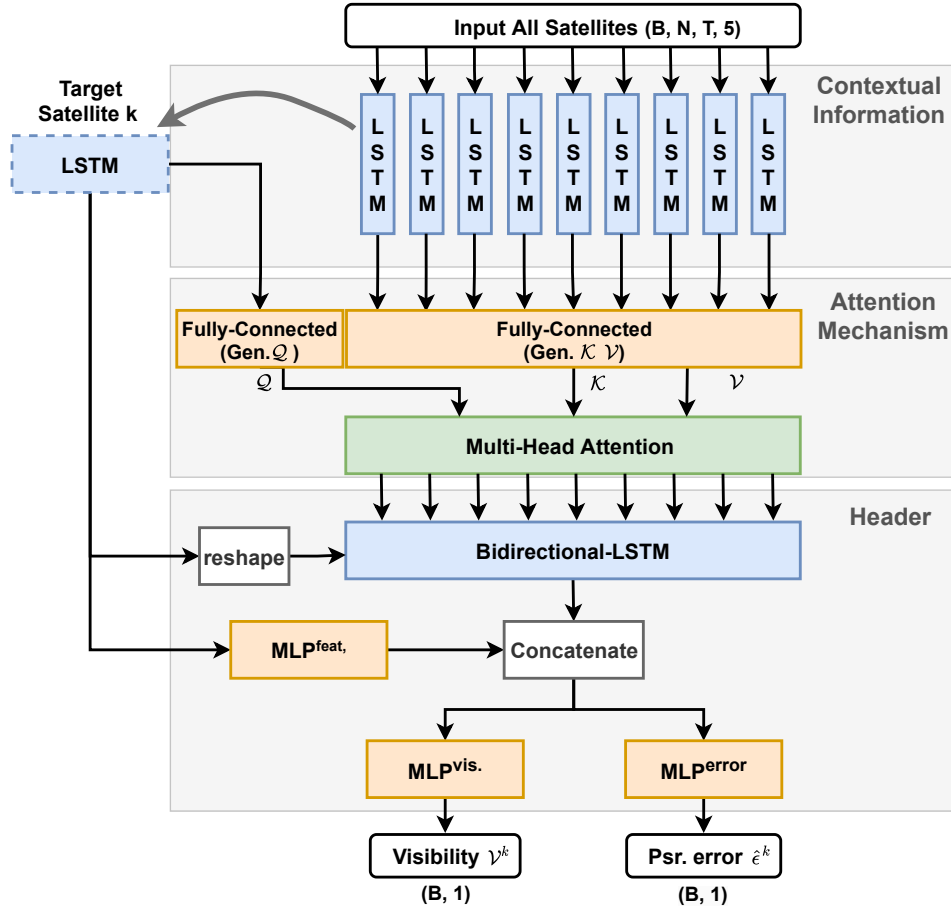


Figure 5.7: Proposed Neural Network Architecture (reproduced from [M3] with permission of IEEE).

features, (2) evaluating spatial contextual information from other satellites, and (3) providing effective attention for discriminative features.

The architecture of the proposed neural network is presented in Figure 5.7.

Assuming that N satellites can be received at maximum, I formulate the input tensor as a T second time series of feature vectors from each satellite in a mini-batch B . To extract high-level features that contain temporal information from the received observations, I process the time series of each satellite with LSTM networks individually. The LSTM output features of all satellites are forwarded to a fully connected layer to generate contextual key $\mathcal{K}$ and value $\mathcal{V}$ features equivalent to transformer networks. Meanwhile, I detach the LSTM output features as input for three network components to obtain predictions for each target satellite.

Above all, I employ the same fully connected layer to generate the query features $\mathcal{Q}$ for the attention mechanism. The features $\mathcal{K}$, $\mathcal{V}$, and $\mathcal{Q}$ are evaluated in the multi-head attention sub-network, which is designed to propose spatial discriminative features from contextual

information.

In the header stack, I employ a bidirectional LSTM (Bi-LSTM), initializing its hidden states with the output features from the preceding LSTM network of the target satellite. As a variant of LSTM models, Bi-LSTM allows information to flow both forward and backward through the network [189]. Unlike regular LSTMs, which process the input sequence only from left to right, Bi-LSTM processes the sequence in both directions. This dual processing enables the model to learn richer representations of the input, enhancing its ability to capture complex dependencies. This makes Bi-LSTM particularly suitable for extracting contextual information from surrounding satellite observations.

The output features of the attention mechanism are processed in the Bi-LSTM, which is expected to extract the temporal information of the discriminative features and mitigate the effect of satellite ordering in the input tensor. I concatenate the output of the Bi-LSTM network with LSTM features of each target satellite to obtain high-level semantic features for classification and regression problems, where a binary visibility indicator $\mathbf{V}^k$ and a predicted pseudorange error $\hat{\epsilon}^k$ of the target satellite k are inferred, as shown in Figure 5.7.

5.6.3 Loss Function of the Proposed Model

When training the proposed model, the loss function must be carefully designed to ensure optimal performance. The customized loss function is a weighted combination of the visibility loss function and the pseudorange error loss function, defined as

$$\text{Loss} = w_1 L_1(\mathbf{y}, \hat{\mathbf{y}}) + w_2 L_2(\mathbf{y}, \hat{\mathbf{y}}), \quad (5.10)$$

where the function L_1 is the loss function for visibility prediction, L_2 is the loss function of pseudorange error prediction and the parameters w_1, w_2 are the weight coefficients of loss L_1 and L_2 . The variables $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote the input feature tensor and the ground-truth label tensor, respectively.

In addition, loss functions L_1 and L_2 are not invariant. Different loss functions will be assigned to L_1 and L_2 to achieve satisfactory validation results. If the model demonstrates good performance on the validation dataset, it is considered well-trained, and the model will then be evaluated on the test dataset. In this thesis, the summed MSE assigned to L_1 and the MAE will be assigned to L_2 .

5.7 Experiment Design and Results

5.7.1 Performance Metrics of NLOS Classification

5.7.1.1 Data In-Distribution

To evaluate the effectiveness and generalization capability of the proposed attention-based deep learning model for NLOS classification, I conducted a comprehensive comparative study involving both classical machine learning algorithms and a state-of-the-art deep learning baseline proposed by Zhang et al. [90]. Specifically, I considered Support Vector Machines (SVM), XGBoost, and two deep learning models: the baseline LSTM and my proposed attention-enhanced architecture. Each model was trained and evaluated on the same dataset to ensure a fair comparison, with performance metrics including precision, recall, F1-score, and overall accuracy summarized in Tables 5.1 and 5.2.

The HK dataset, characterized by a relatively balanced ratio of LOS and NLOS observations, provided an ideal testbed to examine model performance under balanced class conditions (see Table 2.4). In this setting, classical models such as SVM and XGBoost performed comparably well, achieving high scores across all metrics. Interestingly, the classical models can slightly outperform the deep learning baseline in terms of both precision and recall. My reimplementation of the baseline model in Zhang et al. [90] produced results consistent with those reported in the original work, though I observed slightly improved precision.

However, the most notable performance came from my proposed model, which integrates an attention mechanism into the LSTM architecture. This design enabled the model to capture sequential and contextual dependencies in GNSS observations more effectively, resulting in improved classification performance, particularly in terms of NLOS recall. The attention mechanism likely enhanced the model's ability to consider multiple features and their temporal dynamics, which is critical for detecting subtle NLOS patterns that may not be evident from single snapshots of GNSS data.

The AC dataset, in contrast, is heavily skewed toward LOS observations, making it a challenging dataset for evaluating models under class imbalance. In this scenario, all models demonstrated high recall and precision for LOS observations; however, their performance in NLOS classification dropped significantly. The SVM model, in particular, was highly sensitive to the imbalance, nearly failing to detect NLOS observations, as reflected by the recall of 0.01. To further investigate this sensitivity, I constructed a balanced version of the AC dataset and retrained the SVM (denoted SVM*). The results demonstrated a substantial performance boost, particularly in NLOS recall and F1-score. This finding corroborates the conclusions in Batuwita and Palade [190], confirming that classical models are vulnerable to skewed label distributions.

Deep learning models, on the other hand, demonstrated greater resilience to data imbalance. Even when trained on the original, unbalanced AC dataset, the DL models maintained reasonable NLOS classification performance. This robustness is especially important in real-world GNSS environments, where data imbalance is common due to the prevalence of LOS signals. My proposed DL model, especially with six input features, continued to deliver strong generalization and achieved higher NLOS recall and F1-score compared to the baseline, highlighting the benefits of architectural enhancements and feature integration.

Table 5.1: Performance Metrics using HK Dataset (reproduced from [M3] with permission of IEEE).

Model		Precision	Recall	F1-Score	Accuracy
SVM	LOS	0.87	0.86	0.86	0.82
	NLOS	0.72	0.74	0.73	
XGboost	LOS	0.90	0.88	0.89	0.85
	NLOS	0.77	0.80	0.78	
DL [90]	LOS	0.80	0.96	0.87	0.81
	NLOS	0.85	0.51	0.64	
DL (ours)	LOS	0.90	0.91	0.90	0.87
	NLOS	0.80	0.79	0.80	

5.7.1.2 Data Out-Of-Distribution

To evaluate the generalization capability of the models under distributional shift, I designed an experiment in which all models were trained using the HK dataset and then tested on the AC dataset. This out-of-distribution (OOD) scenario mimics real-world conditions where models are applied in new environments that may differ significantly from the training domain.

The results, presented in Table 5.3, indicate a significant decline in performance across all models, particularly for the NLOS class. While the SVM and my proposed DL model achieved slightly better results than other models, the overall recall and precision for NLOS classification were notably lower than in the in-distribution case. This finding underscores the challenge of domain generalization in GNSS learning tasks, where environmental and signal propagation characteristics can vary significantly across urban areas.

One possible reason for the performance drop is the shift in signal characteristics between the HK and AC datasets, such as differences in C/N_0 , satellite geometry, and environmental structures. These changes affect the underlying feature distributions, making it difficult for models trained on one dataset to effectively capture the patterns in another. This performance gap underscores the importance of developing models that are not only accurate but also adaptable to new operating conditions—a goal that my proposed attention-based model partially achieves, yet still leaves room for further improvement.

Table 5.2: Performance Metrics using AC Dataset (reproduced from [M3] with permission of IEEE).

Model		Precision	Recall	F1-Score	Accuracy
SVM	LOS	0.80	1.00	0.89	0.80
	NLOS	0.39	0.01	0.01	
SVM*	LOS	0.91	0.72	0.80	0.72
	NLOS	0.38	0.70	0.49	
XGboost	LOS	0.86	0.88	0.87	0.79
	NLOS	0.47	0.42	0.45	
DL [90]	LOS	0.86	0.83	0.84	0.76
	NLOS	0.40	0.47	0.43	
DL (ours, 5-feature)	LOS	0.86	0.90	0.88	0.81
	NLOS	0.51	0.42	0.46	
DL (ours, 6-feature)	LOS	0.89	0.86	0.87	0.80
	NLOS	0.49	0.55	0.52	

Table 5.3: Cross-Dataset Performance Metrics: The models are trained using the HK dataset and tested with the AC dataset.(reproduced from [M3] with permission of IEEE).

Model		Precision	Recall	F1-Score	Accuracy
SVM	LOS	0.88	0.82	0.85	0.77
	NLOS	0.42	0.54	0.47	
XGboost	LOS	0.87	0.74	0.80	0.70
	NLOS	0.32	0.54	0.40	
DL [90]	LOS	0.86	0.81	0.84	0.74
	NLOS	0.35	0.45	0.39	
DL (ours)	LOS	0.89	0.79	0.84	0.75
	NLOS	0.40	0.60	0.48	

5.7.1.3 Feature Importance in NLOS Classification

To further understand the behavior and learning strategy of each model, I analyzed the permutation feature importance of both classical and deep learning models (Figure 5.8). This analysis provides insight into which features contribute most to model decisions and how feature reliance may affect generalization.

Classical models, such as SVM and XGBoost, exhibited strong dependence on a small subset of features—most notably the elevation angle and C/N_0 . These features are indeed informative for detecting NLOS conditions but may not capture the full context of GNSS signal propagation, especially in dense urban environments where indirect indicators—such as azimuth angle or RSS—can also play a critical role.

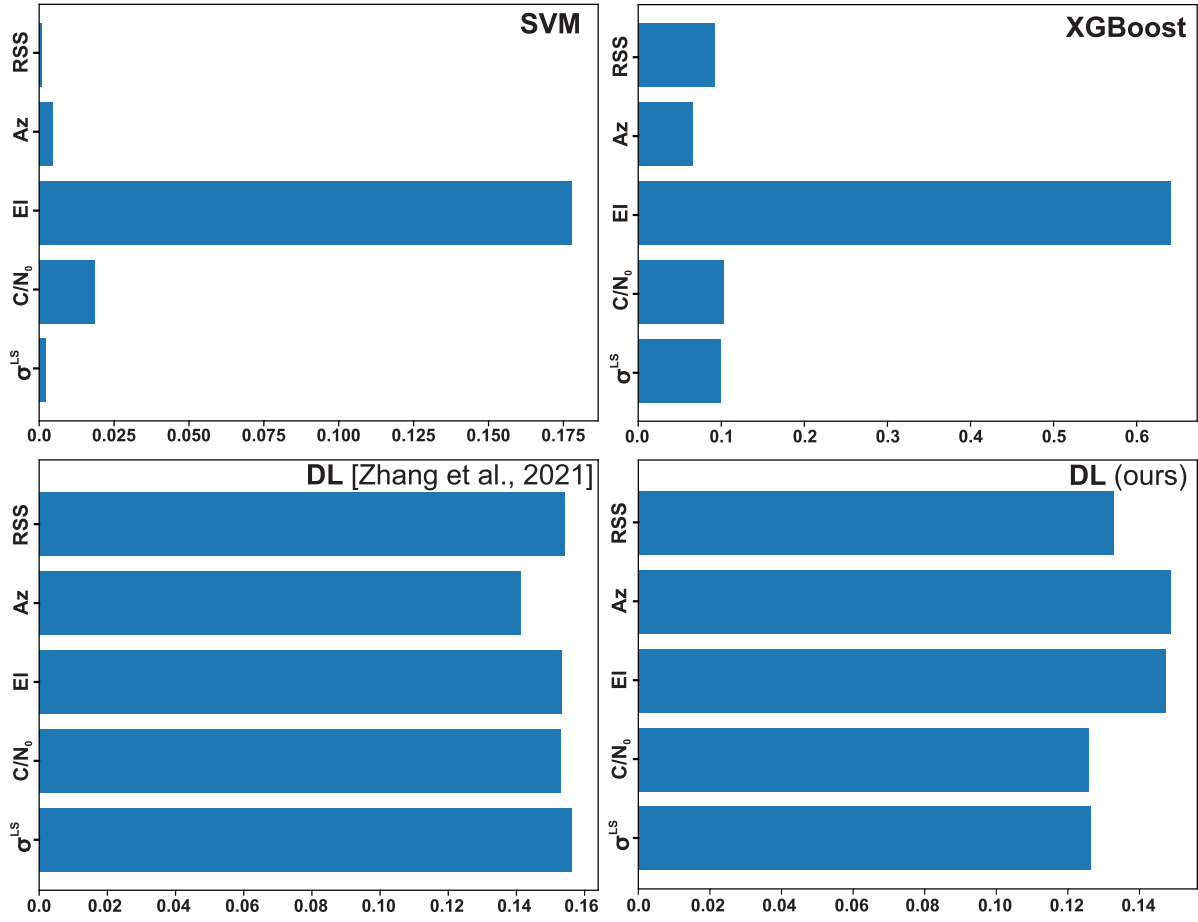


Figure 5.8: Feature permutation importance of different models for NLOS classification (reproduced from [M3] with permission of IEEE).

In contrast, deep learning models showed a more balanced importance distribution across all features. This behavior suggests that DL models are better at integrating contextual cues and learning non-linear interactions among features, contributing to their superior generalization capabilities. The attention mechanism in my proposed model likely enhances this property by explicitly guiding the network to weigh feature contributions dynamically based on the input sequence. This ability to consider broader contextual information makes DL models particularly suited for detecting subtle NLOS conditions that are not easily separable based on a few isolated features.

5.7.1.4 Discussion on NLOS Classification

Through this extensive series of experiments, I demonstrated that both classical and deep learning models can achieve strong performance in NLOS classification when trained on

balanced and representative datasets. However, the generalization ability of these models diverges significantly under more realistic conditions, such as unbalanced or out-of-distribution data.

Classical models often achieve high accuracy by relying heavily on a few dominant features but suffer from limited robustness when deployed in new environments. Their inability to model complex feature interactions makes them susceptible to overfitting, particularly in the presence of spurious correlations.

In contrast, the deep learning models, especially my proposed attention-based network, exhibit greater flexibility and resilience. They maintain reasonable performance across different datasets and demonstrate better utilization of contextual features. Nevertheless, these benefits come with trade-offs in training complexity, interpretability, and computational cost.

I also emphasize the critical role of dataset balance and domain similarity in ensuring reliable performance. Distribution shifts between datasets, particularly in C/N_0 values and environmental conditions (as seen in Figure 2.5), can severely degrade model performance. Addressing this issue may require strategies such as domain adaptation, data augmentation, or transfer learning, which are promising directions for future work.

5.7.2 Performance of Pseudorange Error Prediction

In addition to NLOS classification, I evaluated all models on the task of pseudorange error prediction, which is crucial for enhancing the reliability of GNSS-based positioning in urban environments. This task is inherently more challenging than binary classification, as predicting continuous pseudorange errors involves learning from noisy, multi-modal, and often sparse data distributions.

To visualize the regression performance, I compared the predicted pseudorange errors with the ground truth in Figures 5.9 and 5.10. These figures offer qualitative insights into the model's behavior under various signal conditions. While both classical models (SVM, XGBoost) and deep learning models demonstrated the ability to approximate the general trend of pseudorange errors, none of the models could consistently predict large negative errors, which often correspond to severe multipath or NLOS effects.

This shortcoming is particularly concerning, as large negative pseudorange errors typically have a significant impact on the overall localization performance. These errors are also less frequent in the training dataset, making them more difficult for models to learn. The imbalance in the error distribution likely biases the models toward the more frequently occurring small or positive errors, which in turn limits their capacity to generalize to rare but critical fault cases.

Despite these challenges, the proposed attention-based deep learning model achieved the lowest average prediction error of 5.68 m, outperforming the reference model from Zhang

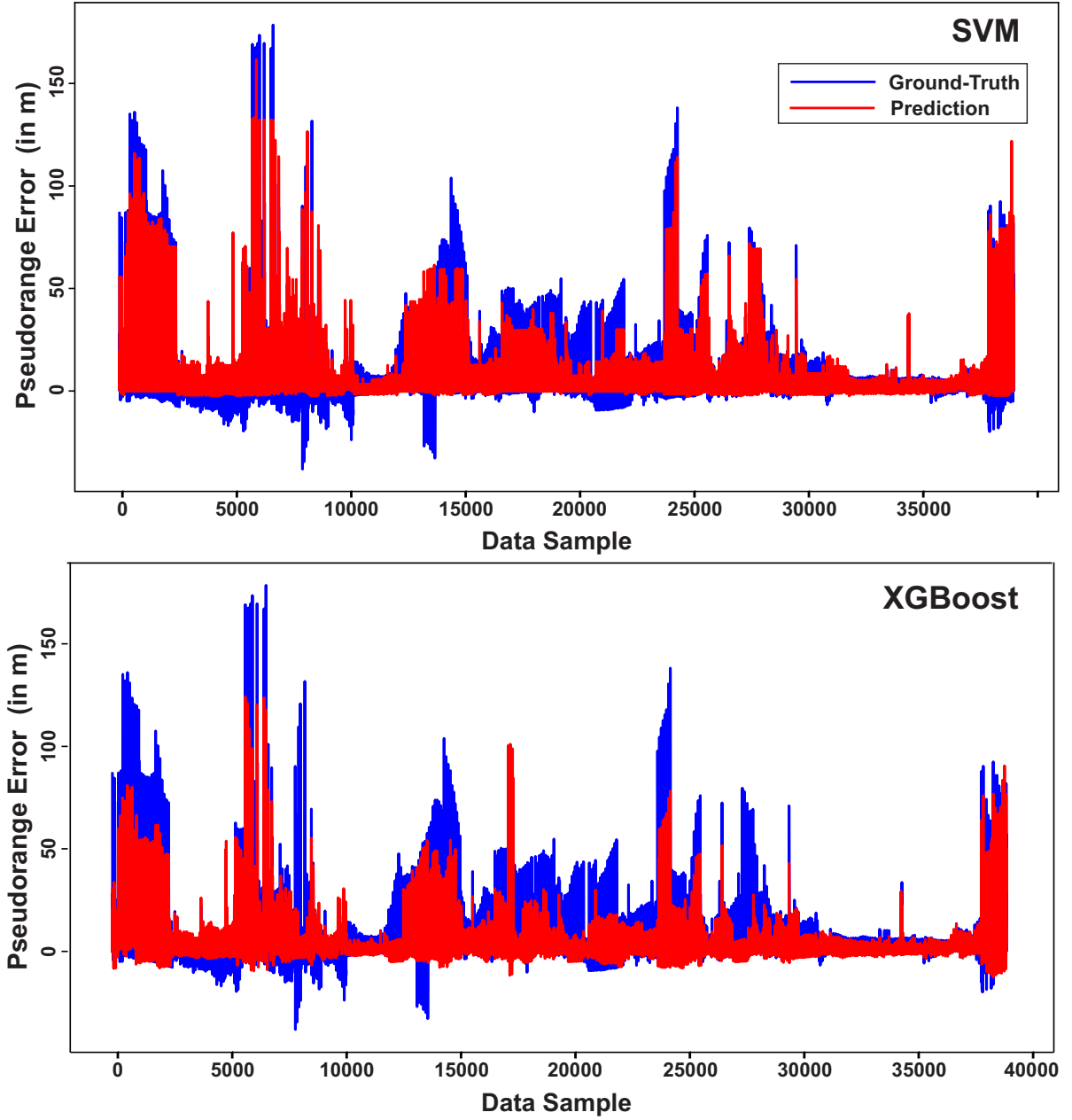


Figure 5.9: Predicted and ground-truth pseudorange errors with SVM and XGBoost.

et al. [90], which had an average error of 6.53 m. This improvement, although moderate, suggests that the attention mechanism enables the model to focus on the most informative temporal and contextual cues when estimating pseudorange deviations. Such cues are often embedded in sequences of satellite geometry, signal quality indicators, and residuals from GNSS positioning solutions.

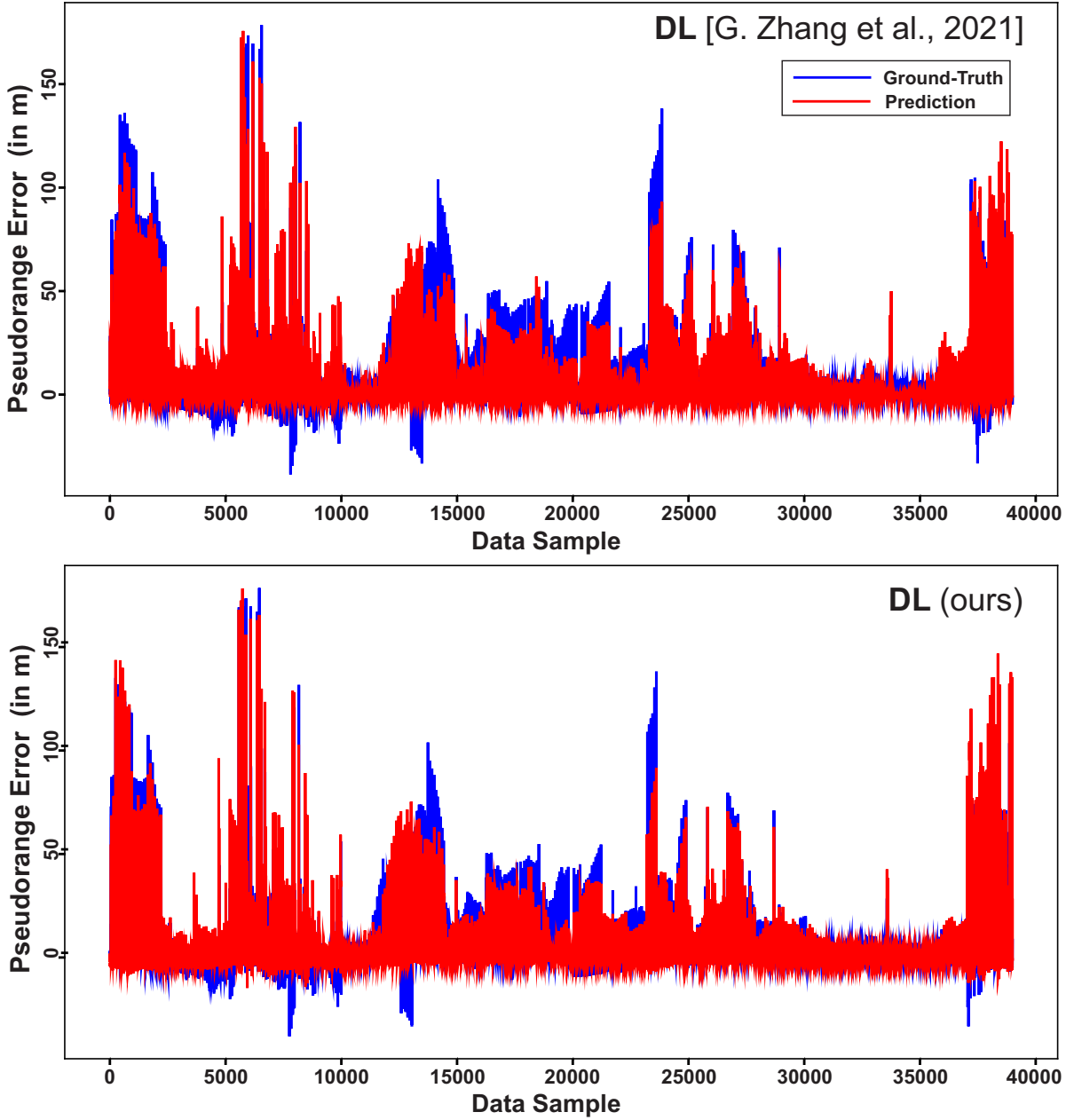


Figure 5.10: Predicted and ground-truth pseudorange errors with the baseline model [90] and the proposed network (reproduced from [M3] with permission of IEEE).

Nevertheless, the absolute prediction accuracy remains insufficient for direct correction or rejection of faulty measurements in critical navigation systems. This observation highlights the limitations of current learning-based approaches for high-precision error regression in GNSS.

5.7.2.1 Feature Importance in Error Prediction

To gain further insight into the regression models, I analyzed feature importance using permutation-based metrics, shown in Figure 5.11. Compared to NLOS classification, the feature preferences in pseudorange error prediction shifted significantly. Notably, in the XGBoost model, the azimuth angle emerged as the most influential feature, indicating that the satellite's orientation relative to the environment plays a significant role in explaining error magnitudes. This finding aligns with the physical understanding that low azimuth angles often correlate with reflected signals or partial obstructions in urban canyons.

Similarly, the SVM model showed increased reliance on the pseudorange residual and the RSS, two features that encapsulate statistical and environmental cues of signal integrity. The diminished role of elevation angle in these models indicates that vertical geometry alone is insufficient for capturing the complexities of pseudorange distortion in urban settings.

Importantly, both classical and deep learning models recognized the value of contextual features, such as RSS and residuals, confirming their relevance in predicting pseudorange anomalies. The DL models, enhanced by the attention mechanism, were better at integrating these features over time, potentially explaining their superior performance. This observation highlights the significance of modeling temporal correlations in GNSS error dynamics, which can be obscured in frame-based approaches.

5.7.2.2 Discussion on Pseudorange Error Prediction

The experiments revealed several limitations in learning-based pseudorange error prediction, despite moderate improvements achieved by my proposed deep learning model. First, the data distribution of errors is heavily skewed, with large errors being underrepresented. This makes it challenging for models to learn robust regression patterns for the most critical cases.

Second, the underlying sources of pseudorange error—such as multipath, diffraction, and signal blockage—are complex and often non-stationary, making it difficult to generalize error behavior solely from historical observations. These factors introduce high variance into the regression task, which classical models and even deep architectures struggle to accommodate.

Lastly, I observed that the current models are more effective at identifying relative trends rather than absolute values of pseudorange errors. While this can still be useful in prioritizing or weighting satellite observations, it is not yet sufficient for precise correction or hard-threshold-based exclusion of faulty signals.

To address these challenges in future work, I plan to expand the dataset with more examples of high-error conditions and explore advanced network architectures specifically tailored

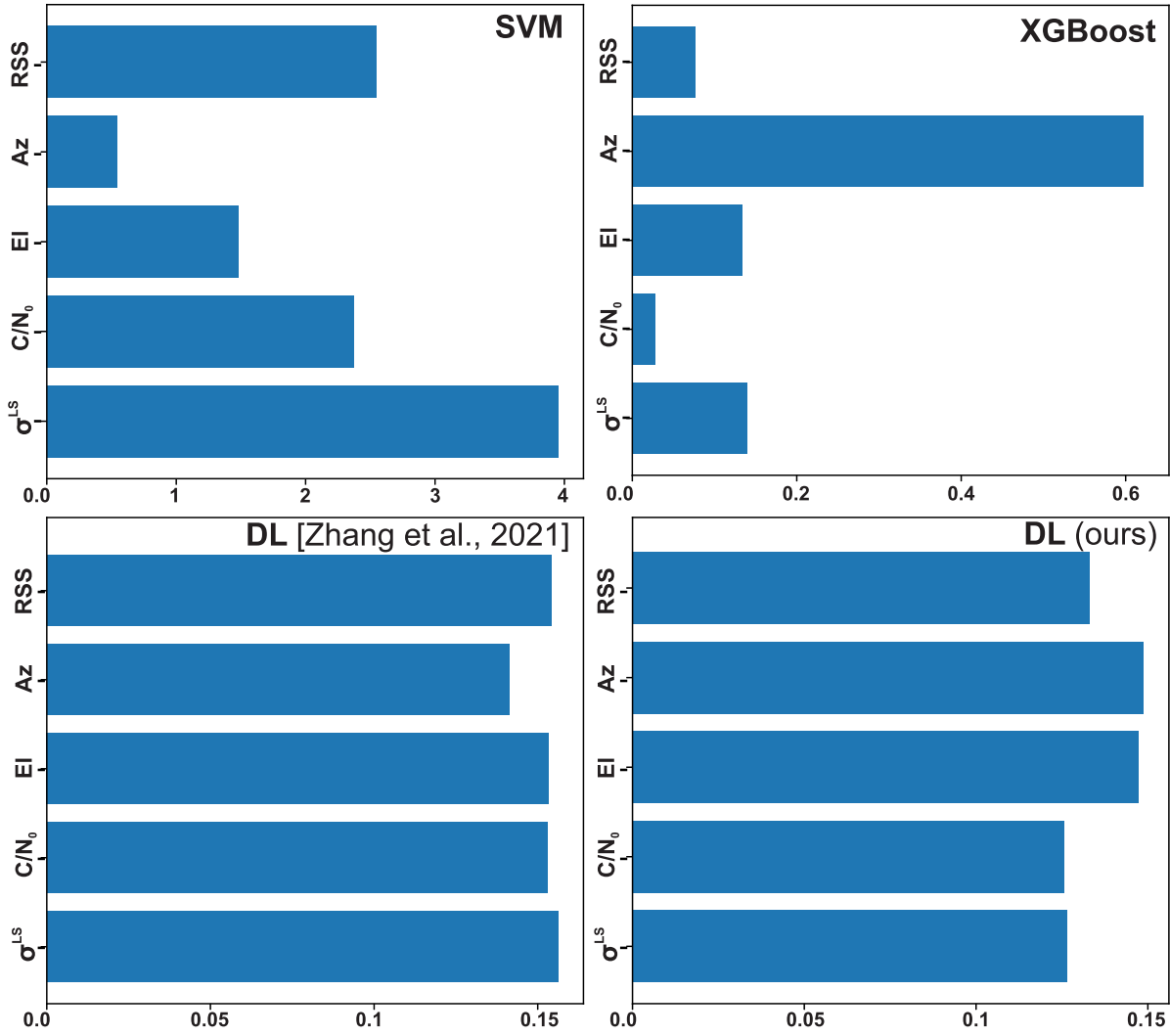


Figure 5.11: Feature Permutation Importance of different models for pseudorange error prediction.

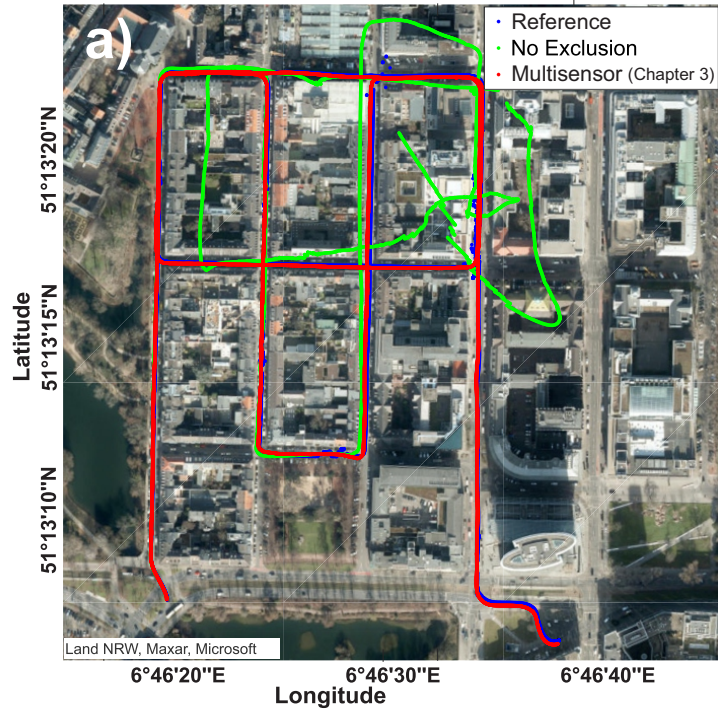
for regression under uncertainty, such as mixture density networks or uncertainty-aware transformers.

5.7.3 Vehicle Localization with GNSS Fault Exclusion

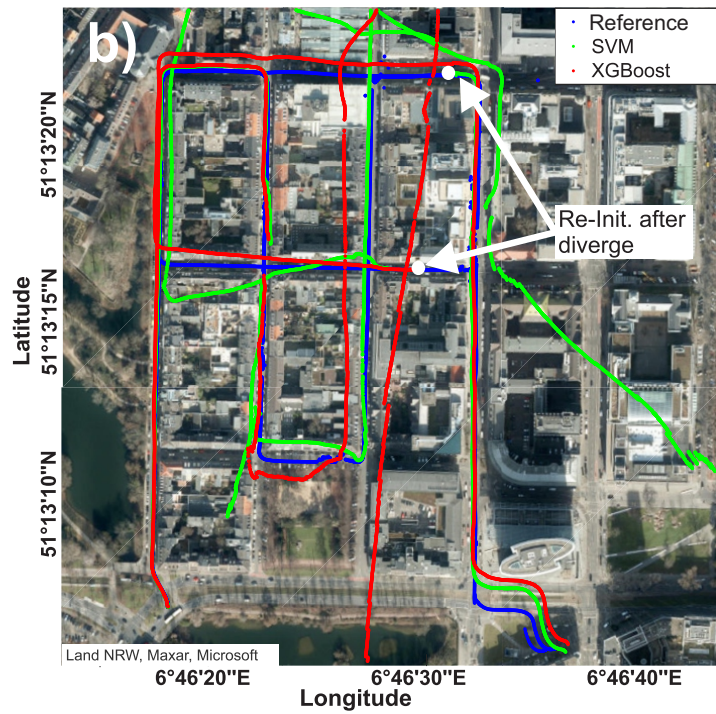
The final set of experiments evaluates the impact of GNSS fault exclusion on vehicle localization in a real-world urban environment. For this, I applied the pre-trained NLOS classification models to GNSS observations collected during a measurement campaign in Düsseldorf. These observations were obtained using a NovAtel PwrPak7 receiver and were not included in the training data, thus representing a realistic out-of-distribution scenario.

Table 5.4: Result of NLOS classification with out-of-distribution data from the measurement campaign in Düsseldorf.

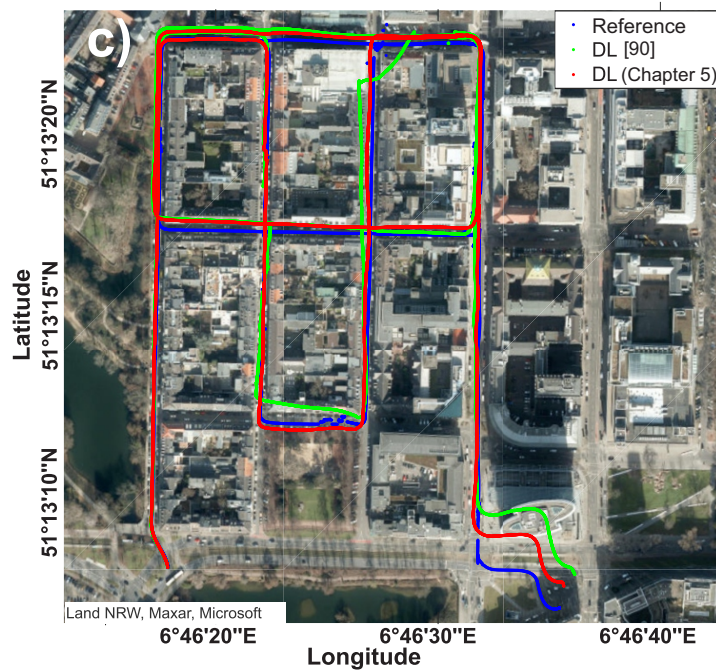
Model	$\hat{R}^{\text{LOS}}$ (%)	$\hat{R}^{\text{NLOS}}$ (%)
SVM	64.83	35.17
XGBoost	58.68	41.32
DL [90]	76.74	23.26
DL (ours)	74.76	25.24



(a) Estimated trajectory without NLOS exclusion and the solution by fusing GNSS observations and lidar odometry in a tight coupling.



(b) Estimated trajectories with NLOS exclusion using classic machine learning models.



(c) Estimated trajectories with NLOS exclusion using deep learning models.

Figure 5.12: Vehicle Localization in Urban Area in Düsseldorf with NLOS Exclusion. The GNSS Positioning-Velocity-Timing (PVT) solution is shown in blue (reproduced from [M3] with permission of IEEE).

Table 5.4 presents the predicted ratio of LOS and NLOS observations for each model. As expected, classical models such as SVM and XGBoost classified a higher proportion of observations as NLOS, while deep learning models retained a larger share as LOS. This discrepancy directly influenced the subsequent vehicle localization results.

I integrated the predicted NLOS classifications into a graph-based factor graph optimization (FGO) framework [M2], which fuses GNSS pseudorange and Doppler shift with inertial measurements in a tightly coupled fashion. RTCM corrections were applied to both GPS and Galileo constellations, and all pseudorange observations flagged as NLOS were excluded from the optimization graph.

As illustrated in Figure 5.12a, GNSS measurements in dense urban environments exhibit severe corruption, often leading to divergence in the estimated trajectory when all observations are naively integrated. In this baseline case, the fusion framework lacks the ability to distinguish between valid and faulty measurements, causing inconsistencies in the factor graph and resulting in estimation errors.

When NLOS observations are excluded using classical models (Figure 5.12b), the trajectory improves in terms of drift suppression, but over-rejection of GNSS observations also leads to trajectory fragmentation and instability. This problem arises because classical models tend to overfit a small subset of strong features, leading to false positives in NLOS classification. As I discussed in Section 5.7.2.1, this overconfidence results from their inability to extract high-level or contextual representations, particularly under novel environmental conditions.

By contrast, deep learning models retained more LOS observations while effectively excluding the faulty ones, producing more consistent trajectories (Figure 5.12c). My proposed network, in particular, excluded a slightly larger number of NLOS observations compared to the baseline DL model while preserving the structure of the trajectory. This outcome indirectly confirms that my model was better at detecting true-positive NLOS cases, as reflected in both classification metrics and localization stability.

However, despite improvements in trajectory consistency, the absolute localization accuracy did not show significant improvement. This limitation is primarily due to the reduced number of GNSS observations available after NLOS exclusion. In dense urban environments, even LOS signals can be sparse, and overly conservative rejection may leave the estimator under-constrained. As a result, consistency improves, but accuracy remains suboptimal.

This experiment demonstrates that learning-based NLOS classification is a viable strategy for improving GNSS robustness in urban settings. However, fault exclusion alone is not sufficient for high-precision localization. To overcome this, I advocate for integrating additional sensors, such as visual odometry or lidar, into the estimation framework. These modalities can complement GNSS in constrained environments, offering redundancy and resilience against signal degradation.

CHAPTER 6

Conclusion

“A person who never made a mistake never tried anything new.”

– Albert Einstein

6.1	Thesis Summary	119
6.2	Contribution and Results	120
6.2.1	C1: Generalized Factor Graph Optimization	120
6.2.2	C2: Handling Outliers using M-Estimators	121
6.2.3	C3: Handling Outliers using Learning-based Methods	123
6.3	Limitations	124
6.4	Open Questions and Future Work	125

6.1 Thesis Summary

The research presented in this dissertation is motivated by unresolved state estimation challenges tailored to vehicle localization in challenging environments. In particular, I emphasized the critical importance of addressing measurement outliers in GNSS-based localization approaches from two key perspectives.

Building on the motivation outlined in Chapter 1, this work provides a clear understanding of the research gaps inherent in classic estimation techniques when faced with measurement outliers. To explore advanced algorithms for robust outlier handling, Chapter 3 introduced a generalized state estimator grounded in continuous-time factor graph optimization. This approach was rigorously validated and evaluated using real-world experimental data across diverse applications, as detailed in Chapter 2.

From a practical perspective, this dissertation investigated two complementary strategies for addressing GNSS measurement outliers. Chapter 4 examined the classical M-estimation framework, while Chapter 5 explored learning-based approaches, which have garnered significant attention in recent years. Together, these chapters provide a comprehensive

examination of both traditional and emerging methods for improving the reliability and resilience of vehicle localization in complex environments.

In this chapter, I summarize our investigations in light of the experimental results, revisiting the contributions and research questions outlined in Chapter 1, Section 1.4.

6.2 Contribution and Results

Firstly, I present three key contributions (denoted as **C**) introduced in Section 1.5 that collectively address the research questions posed in this dissertation.

6.2.1 C1: Generalized Factor Graph Optimization

The first contribution involves the development of an online factor graph optimization framework that generalizes multisensor fusion for robust trajectory estimation, with a particular focus on GNSS. This framework was validated through two distinct driving applications.

Recalling the research question **Q1**:

Remark 6.1: Research Question Q1

How should a state estimator be designed to enable seamless and scalable integration of outlier handling algorithms?

H1.1: Compared to filtering-based approaches, graph optimization typically uses iterative nonlinear optimization techniques, allowing robust error models to be incorporated directly into the solver.

H1.2: In addition, graph optimization creates a local structure of vehicle dynamics that encompasses all historical states, measurements, and control variables, providing a natural framework for approximating error models with data-driven methods.

I can confirm that both hypotheses **H1.1** and **H1.2** are valid, given the experimental summaries presented above.

Compared to filtering methods, such as Kalman filters, methods based on factor graph optimization (FGO) offer several advantages. FGO offers enhanced trajectory smoothness and accuracy by utilizing a batch optimization framework that incorporates a broader temporal context. This enables FGO to more effectively handle challenging scenarios with corrupted or denied GNSS observations, demonstrating superior robustness and flexibility in selecting motion priors and solver configurations.

Moreover, the nonlinear optimization framework of FGO inherently provides an iterative updating strategy, allowing seamless integration with robust error modeling algorithms to mitigate the impact of outliers and measurement errors. Additionally, the factor graph serves as a natural “data container”, storing all information related to the estimation problem.

This structure not only allows for an automatic correction of inaccurate estimates with new sensor observations, improving trajectory smoothness—an essential factor for downstream control applications—but also facilitates advanced inference algorithms. Specifically, stored data, such as residual samples from particular sensors, can be queried to enable techniques like online learning of noise models, further enhancing the adaptability and precision of the estimation process.

I have further demonstrated that continuous-time trajectory representation can enhance the generalization of FGO. A time-centric FGO framework does not depend on any specific sensor, thereby avoiding information loss and mitigating the risk of systematic failures, making it a more robust and flexible approach for state estimation.

Therefore, I summarize the following key insights derived from our investigations to this question: Integrating outlier handling algorithms requires

- an online and flexible state estimator that can incorporate different sensing modalities independently to guarantee state observability in case of sensor degradations.
- an iterative state update that can correct erroneous estimates once new observations and model identification become available.
- the access to relevant data (e.g., past states and measurements) that is crucial to enable advanced inference techniques, such as learning-based methods.

However, as discussed in Section A.2.9, factor graph optimization and Kalman filters address the same problem from a theoretical perspective. Therefore, **I cannot conclude that factor graph optimization is always superior to a well-designed Kalman filter**. Kalman filters retain significant advantages in many practical applications: they are simple, computationally efficient, and capable of rapid convergence due to their Markovian property, especially when consistent sensor observations are available (see Figure 3.7a).

6.2.2 C2: Handling Outliers using M-Estimators

The second contribution comprises comprehensive studies on handling measurement outliers using M-estimators, a topic that has not been extensively explored in the research communities to date.

In summary, soft-redescending M-estimators, particularly the Cauchy and GM estimators, strike the best balance between robustness and accuracy, making them more suitable for range-based vehicle localization. However, hard-redescending estimators pose challenges in

parameter tuning and consistent performance, especially in large-scale, long-term environments. These findings underscore the importance of selecting and tuning M-estimators in accordance with the specific application and data characteristics.

Our conclusion partially addresses research question **Q2**, shedding light on how to effectively utilize M-estimators for handling measurement outliers. Additionally, I validate our first hypothesis (**H2.1**) on M-estimation, demonstrating that a greedy "robust" estimator with lower efficiency does not necessarily lead to improved localization performance.

Remark 6.2: Research Question Q2

How should kernel functions for M-estimators be chosen and parameterized for ranging measurements? Additionally, given a proper kernel for range outliers, how can we effectively balance its robustness and efficiency in degraded environments where measurements are both insufficient and highly corrupted?

H2.1: A well-parameterized M-estimator for one scenario may not deliver the same performance in other scenarios, especially when the state estimator faces challenges from both measurement outliers and insufficient state observability.

H2.2: To effectively tune M-estimators, relying solely on statistical information, such as past residuals, is insufficient, as the noise behavior of measurement outliers can dramatically change due to environmental interference.

However, I cannot offer a definitive strategy for selecting hyperparameters to generalize M-estimation for long-term vehicle localization in large-scale environments. The variability in outlier characteristics and environmental conditions poses significant challenges, particularly when balancing robustness and efficiency. This trade-off must be carefully assessed to ensure sufficient state observability while effectively rejecting outlier measurements.

This challenge is particularly pronounced for GNSS-based localization methods, as measurement corruption often coincides with a complete loss of signal reception. Consequently, even a well-designed M-estimator with appropriately tuned hyperparameters cannot guarantee the desired estimation performance across diverse scenarios, especially in the face of varying outlier characteristics and signal reception conditions.

Although previous works, such as those proposing generalized kernel functions [50, 51] or self-tuning estimators [48, 49], have demonstrated improved generalization of M-estimation across different scenarios, these approaches still face limitations in achieving consistent performance under the diverse and dynamic conditions of GNSS-based localization.

Particularly, current self-tuning approaches, which rely on past residuals, face significant challenges when the noise model of the current measurement suddenly changes, such as in the case of multipath receptions. This sudden variation can disrupt the tuning process and degrade estimation performance. Even if I lack experimental results to directly verify this

hypothesis (H2.2), I can infer its validity from Figures 4.5–4.8, where the errors across all M-estimators at different efficiencies consistently jump to higher values as the vehicle exits GNSS-degraded locations.

This leads us to a new research question associated with the third research question Q3 (see the next section):

“Can we predict measurement outliers using learning-based methods before integrating the measurements into the state estimator?”

Or, even better,

“Can we leverage learning to obtain the missing information required for tuning the M-estimator?”

6.2.3 C3: Handling Outliers using Learning-based Methods

The last contribution presents a transformer-enhanced LSTM network for detecting NLOS receptions and predicting pseudorange errors in GNSS observations. This proposed network was evaluated against a baseline LSTM model and classical machine learning approaches using datasets from Hong Kong and Aachen. Moreover, I introduced a dataset generation process leveraging a multi-sensor state estimator, with experimental studies conducted on both in-distribution and out-of-distribution data.

For the NLOS classification task, I demonstrate that learning-based models perform well on well-balanced datasets, with classical models achieving high precision and recall due to their reliance on features strongly correlated with NLOS receptions. However, their recall drops significantly on out-of-distribution data, exposing limitations in generalization. In contrast, deep learning models deliver better generalization, with our attention-enhanced network outperforming others by capturing spatial and temporal GNSS features, even for out-of-distribution data.

In contrast, all investigated models struggle to reliably predict pseudorange errors, although the proposed network achieves higher prediction accuracy. This limitation arises from the complexity of noise distributions resulting from multiple interference sources and an insufficient number of training samples that represent faulty pseudoranges with large errors. Improving the network architecture for regression tasks and increasing the availability of faulty GNSS observations in the training dataset could help address this issue and enhance prediction performance.

Based on the conclusions above, I confirm the validity of hypothesis H3.1 related to research question Q3: deep networks generally outperform classical models in terms of generalization, particularly for outlier detection tasks. However, classical models can achieve higher accuracy

and recall when trained with well-defined and balanced datasets, although they tend to neglect contextual information provided by features such as mean residual errors.

Remark 6.3: Research Question Q3

How should learning-based methods be integrated into probabilistic state estimators? Specifically, how effectively can learning-based methods generalize to predict measurement errors with limited training samples?

H3.1: Deep neural networks are expected to perform and generalize better than classic machine learning methods (e.g., kernel machines), particularly in handling range outliers that were not encountered during training.

H3.2: Outlier handling with discriminative models should be limited to the measurement pre-processing stage, as these models lack the necessary guarantee of consistent uncertainty prediction required for robust state estimation.

As for hypothesis **H3.2**, I cannot assert that all learning-based methods addressed in this thesis can be integrated into the state estimator, particularly in the back-end optimization process. Discriminative models excel in outlier detection and measurement correction, making them highly valuable for pre-processing stages.

However, their current limitations in uncertainty prediction and handling complex, out-of-distribution scenarios underscore why they should not yet replace more robust techniques in the core state estimation process. This is supported by the results of pseudorange error prediction, where all models struggled to accurately infer large pseudorange errors, primarily due to the limited availability of representative training samples.

Therefore, I can also validate this hypothesis based on the results presented in the vehicle localization tasks, as illustrated in Figure 5.12.

In summary, current studies on learning-based methods for outlier handling **primarily support their integration at the measurement pre-processing stage**. This approach enables these methods to detect and mitigate outliers effectively before passing the data to probabilistic state estimators, thereby addressing our third research question.

6.3 Limitations

However, the work presented in this thesis has three major limitations (denoted as **L**):

L1 Underexplored state representations and measurements:

In Chapter 3, I did not fully leverage GNSS observations such as carrier-phase data, which require complex techniques to resolve ambiguities and detect satellite cycle slips.

Furthermore, I did not prioritize hyperparameter tuning for the GP-WNOJ model or investigate advanced state representations, such as those on $SE_2(3)$ [191] or Galilean Group $SGal(3)$ [192].

Incorporating additional sensor modalities, such as visual odometry, into the fusion framework could further enhance performance.

L2 No studies on generalized and self-tuning M-estimators:

As summarized in Section 6.2.2, our studies on M-estimation focused exclusively on individual kernel functions, which are likely to fail in large-scale scenarios with diverse outlier characteristics.

Therefore, this work can be extended to generalized and self-tuning M-estimators, particularly to explore their limitations, as measurement residuals collected in the past may not adequately represent future outliers. Furthermore, none of the current studies have investigated the residual properties necessary to optimally tune the hyperparameters in adaptive M-estimators.

L3 Limited data and experiments for learning-based methods:

As discussed in Section 5.7.1 and Section 5.7.2, our studies on learning-based methods for handling measurement outliers relied on limited data from Aachen and Hong Kong. While this allows us to evaluate the methods' performance in specific scenarios, it constrains the generalizability of our findings to broader or more diverse contexts. Expanding the dataset to include more varied environments and conditions would provide a stronger foundation for assessing and improving these methods.

In addition, I did not evaluate the computational efficiency of learning-based methods, an aspect that plays a crucial role in real-world applications. Future studies could focus on this area to ensure these methods meet the performance requirements for practical deployment.

6.4 Open Questions and Future Work

As discussed in the previous section, the developments and studies presented in this thesis are subject to several limitations that should be investigated in future work.

Additionally, since my investigations encompass a broad spectrum across multiple research areas, there exist additional advanced ideas and techniques for each component addressed in this work. These alternative approaches may prove more effective in handling outliers, ultimately contributing to more reliable and resilient vehicle localization for long-term operations in challenging environments.

This section highlights several open questions and potential new directions, in addition to the limitations discussed earlier, providing a foundation for further exploration and innovation in the field.

How can contextual information be integrated to adaptively tune M-estimators online?

Our results in Chapter 4 highlight that the efficiency of M-estimators must also be considered, particularly in scenarios with insufficient measurements for handling GNSS outliers. Unfortunately, current adaptive M-estimator approaches primarily emphasize improving robustness, often overlooking a well-balanced adaptation between robustness and efficiency. Therefore, future research should aim to address this gap by optimizing both aspects simultaneously.

This can be addressed by developing a novel generalized kernel function that encompasses various categories of M-estimators, ranging from soft-redescending to hard-redescending types. To effectively tune the adaptation parameters, contextual information, such as outlier predictions from learning-based methods, can be integrated with past residual samples, ensuring a more dynamic and adaptive approach. Studies on residual qualities and kernel sensitivities can also be explored in future works to better understand their impact on performance and to guide the development of more robust and adaptive M-estimator frameworks.

Can we develop reliable online algorithms to identify and adapt to changing sensor noise characteristics?

As Pfeifer and Protzel [63] presented, noise distribution approximation using a Gaussian Mixture Model (GMM) trained via variational Bayesian inference offers a promising technique for handling measurement outliers. However, current studies have only demonstrated the potential of this approach, leaving several research questions unanswered.

For instance, the quality and quantity requirements for the residual samples used to train the GMM remain unclear. Additionally, the authors in Pfeifer and Protzel [63] did not address critical aspects such as uncertainties and distribution shifts in the residual samples, which could significantly impact the robustness and applicability of this technique.

Another critical aspect of this approach lies in the high nonlinearity and non-convexity of the trained GMM, which represents the noise characteristics of GNSS measurements. This poses significant challenges for classical state estimators that rely on solving nonlinear least-square problems. Although Pfeifer et al. [193] proposed a method to address this issue by damping the cost between max-mixture and sum-mixture models, it does not completely resolve the inherent non-convexity of the optimization problem. Consequently, alternative

techniques, such as Hessian approximation [194] and hybrid optimization approaches [74], warrant further investigation to effectively tackle this challenge.

What strategies can enhance the resilience of state estimators under sudden sensor failures or adversarial conditions?

Finally, the research topic of resilient state estimation can be revisited in light of the studies presented in this thesis and recent advances in outlier handling and multisensor fusion techniques.

For example, future work could focus on integrating different learning and tuning processes to enhance classical state estimators, as illustrated in Figure 6.1. This approach could help achieve a balance between robustness, efficiency, and adaptability in dynamic and challenging environments.

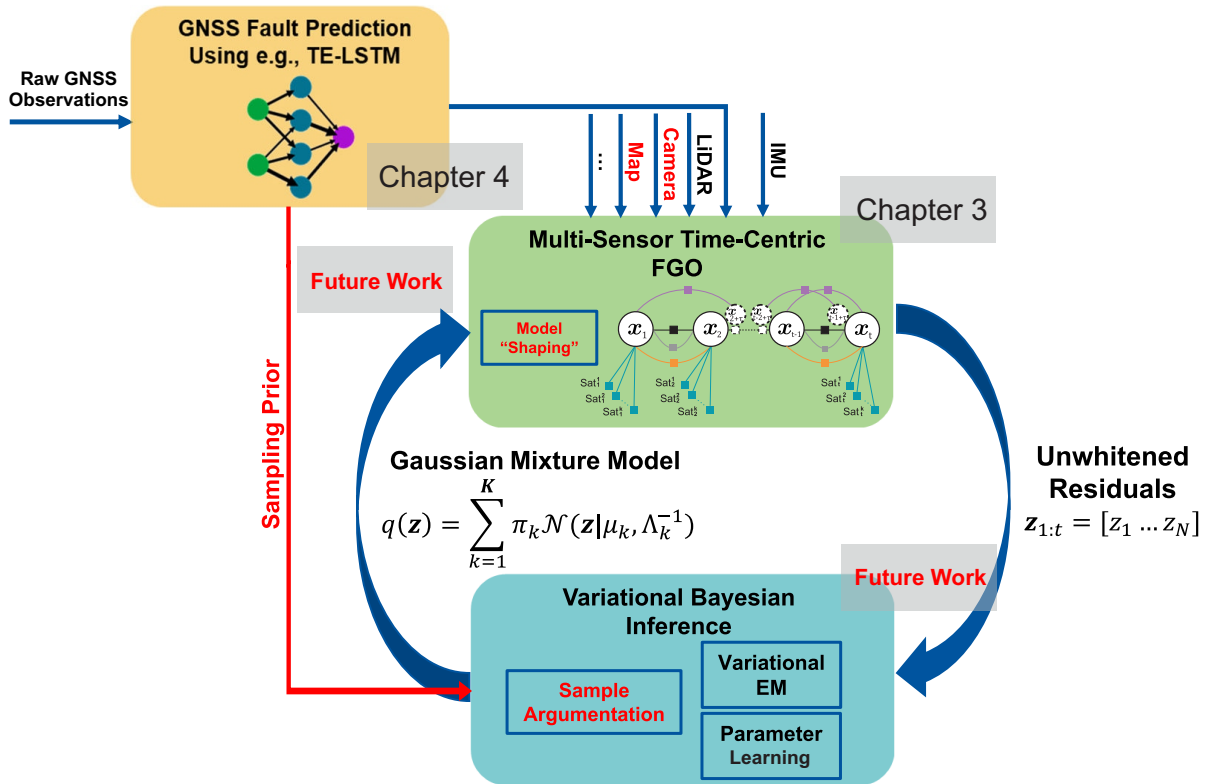


Figure 6.1: Example of resilient state estimation system by extending the methods proposed in this thesis with online noise distribution approximation using variational Bayesian inference, where the red contents indicate future work.

APPENDIX A

Theoretical Foundation

A.1	Motivation	129
A.2	Primer on State Estimation	130
A.2.1	State Representation and Observation	131
A.2.2	Probabilistic Theory and Inference	132
A.2.3	Least-Square Estimation	140
A.2.4	Kalman Filter Family	147
A.2.5	Filtering versus Smoothing	156
A.2.6	Inference using Graphical Models	158
A.2.7	Factor Graph Optimization	161
A.2.8	Incremental Smoothing and Mapping	165
A.2.9	Kalman Filter versus Factor Graph Optimization	168
A.3	Primer on Vehicle Localization	171
A.3.1	Frames and Frame Transformation	171
A.3.2	State Space for Vehicle Localization	173
A.3.3	Lie Group and Lie Algebra	174
A.3.4	Related Sensors and Observations	177
A.3.5	Sensor Fusion	185
A.3.6	Simultaneous Localization and Mapping	186
A.3.7	Performance Metrics of Vehicle Localization	188
A.4	Primer on Machine Learning	190
A.4.1	Learning from Data	190
A.4.2	Classic Machine Learning	191
A.4.3	Deep Learning	194
A.4.4	Performance Metrics of Learning-based Methods	200

A.1 Motivation

This chapter covers the theoretical foundations and practical tools utilized throughout this thesis for developing robust state estimation algorithms with advanced outlier handling techniques. I begin with the fundamentals of the state estimation problem from the

perspectives of probabilistic theory and Bayesian inference. Additionally, I introduce prevalent algorithms, such as Kalman filters and graph optimizers, providing detailed interpretations and discussions of their strengths and limitations. Following this, I present a comprehensive review of vehicle localization techniques within the sensor fusion framework, covering relevant sensor models, various fusion mechanisms, and simultaneous localization and mapping (SLAM). Finally, I briefly introduce the key concepts in machine learning and deep learning that are applied in this work.

Our goal in this chapter is to provide an in-depth understanding of the theories behind widely adopted methods in vehicle localization applications. I hope that my interpretation of these foundational concepts and well-established tools can serve as a beginner-friendly guide for tackling similar challenges and applications.

While this thesis cannot cover all theoretical details, I recommend that readers refer to Thrun et al. [3] and Barfoot [1] for fundamentals of state estimation and to Bishop [56] and Murphy [195] for learning-based methods. The GNSS foundations and details about inertial-supported navigation can be found in Groves [107].

A.2 Primer on State Estimation

State estimation formulates a fundamental problem in autonomous systems, aiming to reconstruct the underlying state of a system based on sensor observations and prior system models [1, Chapter 1.2]. Given the extensive body of research on state estimation that has developed over the past few decades, this section provides an in-depth review of the foundations and theories underpinning state estimation, addressing the following key questions to establish a clear roadmap for readers.

1. Why do we need to estimate the state rather than directly measure/compute it?

This question is addressed in Section A.2.1, where I introduce the state-space model and discuss the uncertainties in state observations.

2. What fundamental mathematical principles underpin state estimation?

Various theoretical frameworks underpin state estimation applications, including *control theory* [196, Chapter 8], *interval analysis* [197], and *Bayesian inference* [56, Chapter 2.3]. In this thesis, I base our state estimation approach on Bayesian inference, guided by probabilistic theory in Section A.2.2, as it provides an intuitive framework for understanding uncertainties and solving inference.

3. How can state estimation problems be solved?

Section A.2.3 introduces least-squares estimation, a fundamental technique for solving state estimation problems. I also cover the basic solvers used for least-squares estimation, which are implemented in the practical algorithms presented in the following section.

4. How can state estimators be implemented in practice?

I introduce popular state estimation algorithms, including *recursive Kalman filters*, in Section A.2.4 and *batch optimization* with factor graphs in Section A.2.7. Given the extensive use of Kalman filters in various applications, particularly in GNSS-based vehicle localization, I provide a detailed explanation of Kalman filters to highlight their differences from graph optimization, which serves as the foundational state estimator throughout this thesis.

A.2.1 State Representation and Observation

The most common machinery for representing system dynamics in control and estimation is the *state-space model*, driven from the linear *stochastic differential equation* (SDE)

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{F}\mathbf{w}(t), \quad (\text{A.1})$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \boldsymbol{\nu}(t), \quad (\text{A.2})$$

where the vectors $\mathbf{x}$ and $\mathbf{u}$ in (A.1) denote the system's state and input (aka. driving function) in continuous time, respectively. The vector $\mathbf{w}(t)$ represents the white process noise affecting the system dynamics. The system matrix $\mathbf{A}$ characterizes the system dynamics, while $\mathbf{B}$ and $\mathbf{F}$ are the input matrix and noise shaping matrix, respectively. These matrices map the system's input and process noise into the state space. As the equation (A.1) formulates the system's interoceptive dynamics, equation (A.2) captures the relationship between the system's state and input vectors and its observables, $\mathbf{y}$ ¹. The matrix $\mathbf{C}$, known as the observation matrix, projects the state space into the measurement space. The term $\boldsymbol{\nu}$ represents the measurement noise.

In discrete time, the state-space model can be expressed as

$$\mathbf{x}_k = \mathbf{A}_D\mathbf{x}_{k-1} + \mathbf{B}_D\mathbf{u}_{k-1} + \mathbf{F}_D\mathbf{w}_k, \quad (\text{A.3})$$

$$\mathbf{y}_k = \mathbf{C}_D\mathbf{x}_k + \boldsymbol{\nu}_k, \quad (\text{A.4})$$

where all system matrices are denoted in their discrete form $\{\cdot\}_D$. Specifically, we note that the system matrix $\mathbf{A}$ must be converted from continuous time into discrete time as $\mathbf{A}_D = e^{\mathbf{A}t}$, which is commonly referred to as the system transition matrix.

The state-space model is fundamental in many estimation algorithms, such as Kalman filters. While the equation (A.1) is generally used to propagate the system states given *interoceptive* system variables, such as inertial measurements from an IMU², the measurement $\mathbf{y}$ in (A.2)

¹This thesis assumes that the input vector $\mathbf{u}$ and the observation $\mathbf{y}$ are independent.

²Since the actual input signals and the vehicle's precise dynamics model are often unavailable, state-of-the-art localization methods typically rely on the IMU to estimate the motion input variables, even though it functions as a sensor.

can be obtained from *exteroceptive* sensors (e.g., GPS). However, our focus at this stage is not on how the state-space model is used in estimation algorithms. Rather, we are interested in examining the problems of this model to understand why it is not sufficient to directly calculate or observe the system’s state using (A.1) and (A.2).

There are two primary challenges associated with the state-space model in real-world applications: *nonlinearity* and *uncertainty*. In practice, technical systems are often described by nonlinear dynamics and sensor models. While common estimation approaches employ linearization techniques (e.g., Taylor expansion) for uncertainty propagation, they tend to be highly sensitive to the magnitude of the uncertainty [198, Chapter 2.5]. Alternative methods, such as the unscented transformation [198, Chapter 3.2] or cubature rules [199, Chapter 14.7], aim to mitigate the effects of system nonlinearity. However, these techniques still assume that the noise processes can be quantified in advance and ideally follow a Gaussian distribution. In scenarios where the noise process is unknown or influenced by complex, unpredictable disturbances (particularly in the case of measurement noise), the performance of these methods deteriorates.

Thus, the primary challenge of addressing uncertainties is a crucial motivation behind state estimation. In particular, uncertainties are typically characterized as probability distributions that do not provide exact noise quantities $w(t)$ to propagate the state in (A.1) precisely. Therefore, state estimation methods typically interpret uncertainty measures (e.g., noise parameters) as covariance matrices, which are used to weigh state representations, including sensor observations and dynamics models. Now we can further extend the definition of state estimation in Barfoot [1, Chapter 1.2] from a system perspective as:

“Given redundant state representations (e.g., measurements and prior information) with consistent noise parameters, the core objective of state estimation is to find the best estimate while minimizing its noise level (aka. variance) by optimally weighting the redundant representations.”

However, this definition comes with a notable cost: the uncertainties of redundant state representations must be predefined and consistent throughout the operation. This is a challenging requirement, as system dynamics and sensor measurements are often disrupted by external interference. In the next section, I introduce a probabilistic framework that generalizes state estimation from an inference perspective, enabling a mathematical representation of uncertainties.

A.2.2 Probabilistic Theory and Inference

As uncertainty plays a central role in state estimation problems, *probabilistic theory* provides a well-defined language to frame the estimation problem within a mathematical framework,

thereby making the uncertainties tractable. In this section, I will primarily cover the fundamentals of probabilistic theory and Bayesian inference, starting with distributions in the exponential family.

A.2.2.1 Gaussian Distribution

The Gaussian distribution (aka normal distribution) is commonly employed as the foundational noise model in state-space representations for a wide range of estimation algorithms. It reflects the fact, as noted in Bishop [56, Chapter 2.3], that random variables in state estimation problems are typically associated with a symmetric, unimodal noise distribution, consistent with the *central limit theorem*.

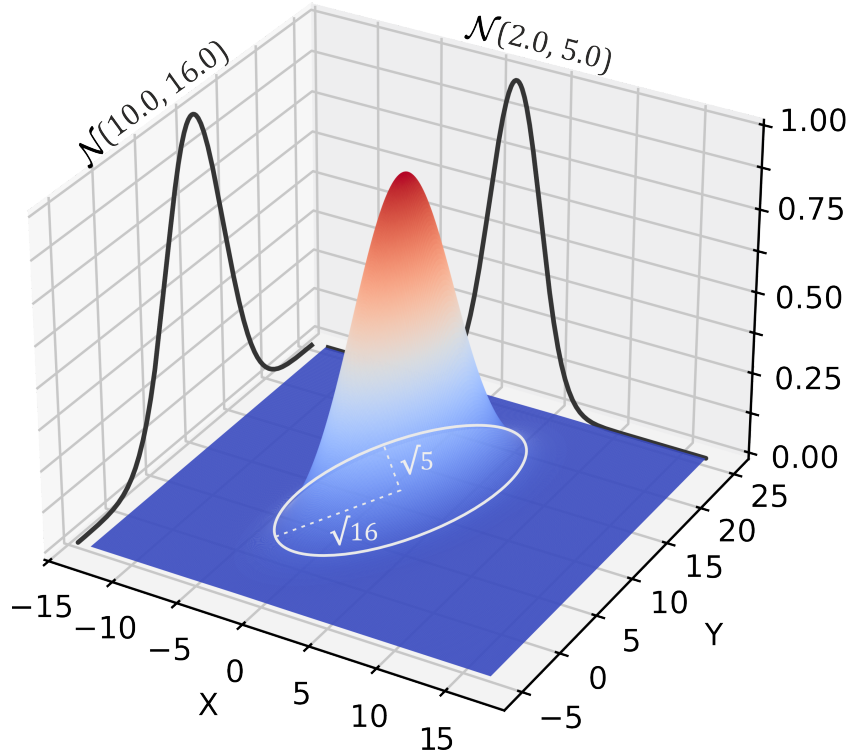


Figure A.1: Example of a 2-dimensional Gaussian distribution.

The probabilistic density function $p(\mathbf{x})$ of a Gaussian distribution associated with a D -dimensional state vector $\mathbf{x}$ is written as

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2} \|\mathbf{x} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}}^2\right). \quad (\text{A.5})$$

The Gaussian distribution is typically characterized by two *moments*: the D -dimensional mean vector $\boldsymbol{\mu}$ and the $D \times D$ -dimensional covariance matrix $\boldsymbol{\Sigma}$. It presents, as illustrated in Figure

A.1, a symmetric and unique-mode distribution shaped by the moments of each dimension. The most important part of the Gaussian distribution, its exponential function-formulates a squared *Mahalanobis distance*

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}), \quad (\text{A.6})$$

which offers a normalized measure to determine the distance of a point estimate from the related distribution, given its scale (variance).

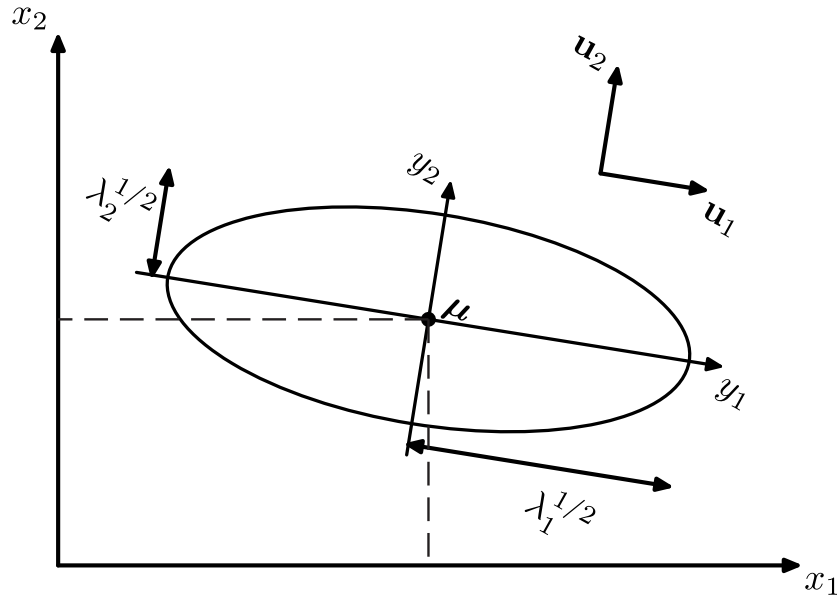


Figure A.2: Projection of 2-D Gaussian into a new coordinate y , where the covariance matrix is decomposed into two independent eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$ (figure adapted from Bishop [56, Figure 2.7]).

The Gaussian distribution serves as a foundational probabilistic model in estimation problems, not only due to the *central limit theorem* (attributed to Laplace), which states that the distribution of a large number of random variables under certain mild conditions converges to the Gaussian [200]. More importantly, it offers numerous critical analytical properties ideal for modeling state variables and their associated uncertainties. One key property is the symmetry of the Gaussian distribution, which ensures that its covariance matrix, $\boldsymbol{\Sigma}$, is also symmetric without loss of generality [56, Chapter 2]. The Gaussian distribution's symmetry eliminates all asymmetric components (aka *skewness*) from its exponent, guaranteeing unbiased errors. Additionally, the covariance matrix can be decomposed into its eigenvectors, enabling factorization as:

$$\boldsymbol{\Sigma} = \sum_{i=1}^D \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \quad (\text{A.7})$$

where the variable λ_i denotes the eigenvalue for dimension i , and the vector $\mathbf{u}_i$ is the corresponding eigenvector. This decomposition shifts and rotates the original coordinate $\mathbf{x}_i$ into a new coordinate system defined by $\mathbf{u}_i$, as illustrated in Figure A.2 [56, Chapter 2]. By decoupling the covariance matrix into eigenvectors, the multivariate Gaussian distribution is indeed *normalized*. Specifically, it satisfies

$$\int p(\mathbf{y}) d\mathbf{y} = \prod_{j=1}^D \int_{-\infty}^{\infty} \frac{1}{(2\pi\lambda_i)^{1/2}} \exp -\frac{y_i^2}{2\lambda_i} dy_i = 1, \quad (\text{A.8})$$

where each transformed coordinate y_i is defined as $y_i = \mathbf{u}_i^T(\mathbf{x} - \boldsymbol{\mu})$. Thus, we have $\mathbf{y} = (y_1 \dots y_D)^T = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$ with the shaping matrix $\mathbf{U}$ composed of rows $\mathbf{u}_i$, where $i = 1, \dots, D$. As elaborated in the subsequent section, this factorization underscores the Gaussian distribution's flexibility and efficiency in probabilistic modeling and estimation tasks, particularly in Bayesian inference (see Section A.2.2.4).

Additionally, the sum of independent Gaussian random variables³, X_1 and X_2 , remains Gaussian, with the mean and variance of the resulting distribution given by

$$X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2). \quad (\text{A.9})$$

Moreover, the product of two Gaussian distributions, $p_1(x)$ and $p_2(x)$, results in a new Gaussian distribution

$$p(x) \propto \exp -\frac{(x - \mu)^2}{2\sigma_1^2} \exp -\frac{(x - \mu)^2}{2\sigma_2^2}, \quad (\text{A.10})$$

$$\propto \exp -\frac{(x - \mu)^2}{2\sigma^2}, \quad (\text{A.11})$$

where the final mean $\mu = \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}$, and the variance is calculated as $\sigma^2 = \frac{1}{1/\sigma_1^2 + 1/\sigma_2^2}$.

In later sections, I will demonstrate these convenient analytical properties of Gaussian distributions, mainly when marginalization is introduced and Kalman filters are implemented.

A.2.2.2 Exponential Family

Distributions similar to the Gaussian that contain an exponent in the functional form

$$p_w(\mathbf{x}) = \frac{h(\mathbf{x})}{Z(\mathbf{w})} \exp(\phi(\mathbf{x})^T \mathbf{w}) = p(\mathbf{x}|\mathbf{w}) \quad (\text{A.12})$$

belong to the *exponential family*, where the function $h(\mathbf{x})$ is the base measure and the normalization constant $Z(\mathbf{w}) \in \mathbb{R}$ formulates the partition function of the distribution [56,

³I exemplify the sum and product of two Gaussians in 1-D for the sake of convenience.

Chapter 2.4]. The parameters $w \in \mathbb{R}^D$ are the natural parameters of p_w . The exponential family establishes a general construction form of distributions and guarantees the existence of *conjugate priors* to the distributions falling into this family. A conjugate prior $p'(\mathbf{x})$ to the probabilistic distribution $p(\mathbf{x})$, is a probability measure with another *probabilistic density function* (pdf) $p'(\mathbf{x}) = \pi(\mathbf{x}; \boldsymbol{\theta})$ of functional form π , such that

$$p'(\mathbf{x}) = \frac{p(\mathbf{x})\pi(\mathbf{x}; \boldsymbol{\theta})}{\int p(\mathbf{x})\pi(\mathbf{x}; \boldsymbol{\theta}) d\mathbf{x}} = \pi(\mathbf{x}; \boldsymbol{\theta}'). \quad (\text{A.13})$$

A conjugate prior $p'(\mathbf{x})$, typically defined with the same or additional parameters, shares the same functional form as the posterior distribution $p(\mathbf{x})$ [201]. In other words, a conjugate prior enables the analytical inference of the unknown posterior distribution by offering a more flexible model with additional free parameters that can be adjusted based on observed data. As an example, a *multivariate Gaussian distribution* with both unknown mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be inferred using the *Gaussian-Inverse-Wishart distribution* as its conjugate prior

$$\underbrace{p(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{\mu}_0, \beta, \mathbf{W}, \nu)}_{\text{Gaussian}} = \underbrace{\mathcal{N}(\boldsymbol{\mu} | \boldsymbol{\mu}_0, \beta \boldsymbol{\Sigma}) \cdot \mathcal{IW}(\boldsymbol{\Sigma} | \mathbf{W} \nu)}_{\text{Gaussian-Inverse-Wishart}}, \quad (\text{A.14})$$

where the matrix $\mathbf{W}$ is a $D \times D$ symmetric, positive definite matrix. The variable $\nu > D - 1$ is called the number of degrees of freedom of the distribution, and the variable β is used as a scale parameter.

Obviously, the unknown Gaussian can be reconstructed by tuning the parameter set $\{\boldsymbol{\mu}_0, \beta, \mathbf{W}, \nu\}$ in the conjugate prior distribution. Therefore, conjugate prior inference within exponential families serves as a fundamental technique in numerous applications, supporting both full Bayesian inference and variational Bayesian inference. I refer the reader to Bishop [56, Chapter 2.4] and Diaconis and Ylvisaker [201] for more details.

A.2.2.3 Cramér-Rao Lower Bound and Fisher Information

The *Cramér-Rao Lower Bound* (CRLB) is a fundamental concept in statistical estimation, providing the theoretical lower bound on the variance of any unbiased estimator. It defines the minimum achievable variance, highlighting the best possible performance of an estimator under given conditions.

To derive the CRLB, we draw a measurement sample $\mathbf{z}$ from the likelihood distribution $p(\mathbf{z} | \hat{\mathbf{x}})$, where we assume that the state $\hat{\mathbf{x}}$ is an unbiased (optimal) estimate. The CRLB assumes that the covariance of this unbiased state $\hat{\mathbf{x}}$ is bounded by the *Fisher information matrix* $\mathcal{I}_{\mathbf{x}}$ of the to-be-estimated state $\mathbf{x}$. This gives us

$$\text{cov}(\hat{\mathbf{x}} | \mathbf{z}) = \mathbb{E}[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T] \geq \mathcal{I}_{\mathbf{x}}^{-1}. \quad (\text{A.15})$$

Since the optimal state $\hat{\mathbf{x}}$ is unbiased, we have $\mathbb{E}[\hat{\mathbf{x}} - \mathbf{x}] = \mathbf{0}$, which means

$$\text{cov}(\hat{\mathbf{x}}|\mathbf{z}) - \mathcal{I}_x^{-1} \geq 0. \quad (\text{A.16})$$

The Fisher information matrix quantifies the amount of information a sample carries about an unknown parameter and is computed as

$$\mathcal{I}_x = \mathbb{E} \left[\frac{\partial^2 (-\ln p(\mathbf{z}|\mathbf{x}))}{\partial \mathbf{x}^T \partial \mathbf{x}} \right]. \quad (\text{A.17})$$

A higher Fisher Information indicates that the parameter can be estimated with greater precision. Together, these concepts serve as key tools in evaluating and comparing the efficiency of estimators. I refer the readers to Barfoot [1, Chapter 2.1.10] for an in-depth explanation.

A.2.2.4 Bayes' theorem and Bayesian Inference

In the previous section, I briefly reviewed the fundamentals of probabilistic theory, which serve as essential tools for addressing state estimation problems. Building on that foundation, I now revisit the definition of state estimation from Section 1.4 and reinterpret the problem through the lens of *Bayesian inference* (BI), which leverages the *Bayes' theorem*

$$\underbrace{p(\mathcal{X}|\mathcal{Y})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{Y}|\mathcal{X})}^{\text{likelihood}} \cdot \overbrace{p(\mathcal{X})}^{\text{prior}}}{\underbrace{p(\mathcal{Y})}_{\text{evidence/normalization } \eta}} \quad (\text{A.18})$$

to infer the *posterior* distribution, $p(\mathcal{X}|\mathcal{Y})$, with the *likelihood*, $p(\mathcal{Y}|\mathcal{X})$, and *prior*, $p(\mathcal{X})$, of the system's states $\mathcal{X}$ given the observations $\mathcal{Y}$. For the sake of generality, I wrap the states and their observations in the sets $\mathcal{X} = \{\mathbf{x}_1 \dots \mathbf{x}_M\}$ and $\mathcal{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_N\}$ in (A.18), respectively. Within the scope of Bayesian inference, the state estimation problem refers to the process of deducing the unknown state variables based on probabilistic models that encapsulate prior information (e.g., dynamics models) and sensor observations.

I now turn our attention to Bayes' theorem and the calculation of the posterior distribution. In many cases, when the likelihood and prior distributions are known—or can be substituted with their conjugate priors—the posterior distribution can be derived in closed form. This process is referred to as “*full*” *Bayesian inference* [202]⁴. The most challenging aspect of

⁴I use the term full Bayesian inference to refer to the classical Bayesian inference approach, which assumes known conjugate distributions for deriving the posterior distribution.

full Bayesian inference is computing the *evidence* (aka. normalization constant or marginal likelihood)

$$p(\mathcal{Y}) = \int_{-\infty}^{\infty} p(\mathcal{Y}|\mathcal{X}) \cdot p(\mathcal{X}) \, d\mathcal{X} \quad (\text{A.19})$$

that sums over all possible states $\mathcal{X}$ weighted by the prior $p(\mathcal{X})$ and the likelihood $p(\mathcal{Y}|\mathcal{X})$. The evidence establishes the pdf describing the measurement (i.e., the data). It ensures the posterior distribution $p(\mathcal{X}|\mathcal{Y})$ in (A.18) is properly normalized, satisfying $\int p(\mathcal{X}|\mathcal{Y}) = 1$, which, however, can be mathematically intractable if $\mathcal{X}$ is high-dimensional or no conjugate priors do not exist between the posterior and prior distributions.

These challenges can be mitigated in practical applications by assuming that full Bayesian inference is feasible. For instance, Kalman filters operate under the assumption that both the prior and likelihood distributions are Gaussian. Since the product of two Gaussian distributions results in another Gaussian, the posterior distribution in Kalman filters can be recursively computed across all sampled measurements. In such applications, a *point estimate* of the system state can be inferred as

$$\hat{\mathcal{X}}^{\text{BI}} = \mathbb{E}[\mathcal{X}|\mathcal{Y}] = \int \mathcal{X} \cdot p(\mathcal{X}|\mathcal{Y}) \, d\mathcal{X}. \quad (\text{A.20})$$

Moreover, since Bayesian inference provides a valid posterior distribution, the uncertainty $\hat{\Sigma}^{\text{BI}}$ of the point estimate $\hat{\mathcal{X}}^{\text{BI}}$ can also be quantified as

$$\hat{\Sigma}^{\text{BI}} = \mathbb{E}[(\mathcal{X} - \hat{\mathcal{X}}^{\text{BI}})(\mathcal{X} - \hat{\mathcal{X}}^{\text{BI}})^T] = \int (\mathcal{X} - \hat{\mathcal{X}}^{\text{BI}})p(\mathcal{X}|\mathcal{Y})(\mathcal{X} - \hat{\mathcal{X}}^{\text{BI}})^T \, d\mathcal{X}. \quad (\text{A.21})$$

At this point, I do not delve into scenarios where full Bayesian inference becomes infeasible due to the inability to compute the evidence in closed form. Such cases often necessitate advanced techniques like *Variational Bayes* or *Monte Carlo simulation*, which fall outside the scope of classical state estimators. Nonetheless, a thorough understanding of Bayesian inference remains essential for this thesis, as it provides the mathematical foundation for implementing state estimation algorithms, which will be introduced in the subsequent sections.

A.2.2.5 Maximum a Posteriori

Let us consider an alternative approach to leveraging Bayes' theorem for state estimation, where the evidence does not need to be computed explicitly. Instead, our objective is to find the system state that best aligns with both the likelihood and prior models from an

optimization perspective. This approach involves solving an optimization problem called *maximum a posteriori* (MAP)

$$\begin{aligned}
\hat{\mathcal{X}}^{\text{MAP}} &= \operatorname{argmax}_{\mathcal{X}} p(\mathcal{X}|\mathcal{Y}) \\
&= \operatorname{argmax}_{\mathcal{X}} \frac{p(\mathcal{Y}|\mathcal{X}) \cdot p(\mathcal{X})}{\underbrace{p(\mathcal{Y})}_{\text{independent of } \mathcal{X}}} \\
&= \operatorname{argmax}_{\mathcal{X}} p(\mathcal{Y}|\mathcal{X}) \cdot p(\mathcal{X}) \\
&= \operatorname{argmax}_{\mathcal{X}} \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n) \cdot \prod_{m=1}^M p(\mathbf{x}_m).
\end{aligned} \tag{A.22}$$

To solve this optimization problem, we now formulate the logarithms of the prior and likelihood distributions using the state-space model in (A.3) and (A.4), assuming both distributions are Gaussian, namely $p(\mathcal{X}) \sim \mathcal{N}(\boldsymbol{\mu}_x|\mathbf{Q})$ and $p(\mathcal{Y}|\mathcal{X}) \sim \mathcal{N}(\boldsymbol{\mu}_y|\mathbf{R})$. Then we calculate the logarithm of the prior distribution in a general form as

$$\begin{aligned}
\ln p(\mathcal{X}) &= \ln p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \\
&= -\frac{1}{2}(\mathbf{x}_k - \underbrace{(\mathbf{A}_D\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k)}_{\boldsymbol{\mu}_{\mathbf{x}_k}})^T \mathbf{Q}_k^{-1} (\mathbf{x}_k - \underbrace{(\mathbf{A}_D\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k)}_{\boldsymbol{\mu}_{\mathbf{x}_k}}) - \underbrace{\frac{1}{2} \ln((2\pi)^D \det(\mathbf{Q}_k))}_{\text{independent of } \mathbf{x}_k},
\end{aligned} \tag{A.23}$$

and

$$\begin{aligned}
\ln p(\mathcal{Y}) &= \ln p(\mathbf{y}_k|\mathbf{x}_k) \\
&= -\frac{1}{2}(\mathbf{y}_k - \underbrace{\mathbf{C}\mathbf{x}_k}_{\boldsymbol{\mu}_{\mathbf{y}_k}})^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \underbrace{\mathbf{C}\mathbf{x}_k}_{\boldsymbol{\mu}_{\mathbf{y}_k}}) - \underbrace{\frac{1}{2} \ln((2\pi)^D \det(\mathbf{R}_k))}_{\text{independent of } \mathbf{x}_k}.
\end{aligned} \tag{A.24}$$

Given (A.23) and (A.24), we can further express their Mahalanobis distance⁵ as the following quantities

$$J_{\mathbf{u},m} = \begin{cases} -\frac{1}{2}(\mathbf{x}_0 - \boldsymbol{\mu}_{\mathbf{x}_0})^T \mathbf{Q}_0^{-1} (\mathbf{x}_0 - \boldsymbol{\mu}_{\mathbf{x}_0}), & m = 0, \\ -\frac{1}{2}(\mathbf{x}_m - \mathbf{A}_D\mathbf{x}_{m-1} - \mathbf{B}\mathbf{u}_m)^T \mathbf{Q}_m^{-1} (\mathbf{x}_m - \mathbf{A}_D\mathbf{x}_{m-1} - \mathbf{B}\mathbf{u}_m), & m = 1 \dots M, \end{cases} \tag{A.25}$$

and

$$J_{\mathbf{y},n} = -\frac{1}{2}(\mathbf{y}_n - \mathbf{C}\mathbf{x}_n)^T \mathbf{R}_n^{-1} (\mathbf{y}_n - \mathbf{C}\mathbf{x}_n), \quad n = 0 \dots N, \tag{A.26}$$

⁵I adjust the time indices of $\mathbf{x}$ and $\mathbf{y}$ to account for potential differences in their timing in practical applications.

which is used to formulate the optimization objective function (aka. cost function)

$$J(\mathbf{x}) = \sum_{m=0}^M J_{u,m}(\mathbf{x}) + \sum_{n=0}^N J_{y,n}(\mathbf{x}). \quad (\text{A.27})$$

Clearly, we can substitute the objective function (A.27) to replace the pdfs in (A.22) and reformulate the MAP problem as

$$\hat{\mathcal{X}}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmin}} J(\mathbf{x}), \quad (\text{A.28})$$

which now reduces to an *unconstrained optimization problem* in practical implementations.

In the next section, we will demonstrate how this optimization problem can be solved to construct the classic least-squares estimator. It is important to note that the MAP problem represents a general optimization approach for obtaining the best estimate based on given likelihood and prior models. Moreover, it can be seamlessly integrated into the Bayesian inference framework to quantify uncertainties, as exemplified in Bishop [56, pp. 441].

A.2.3 Least-Square Estimation

We can see that the cost function $J(\mathbf{x})$ in (A.27) consists of quadratic terms derived from the residuals of the state-space model. Consequently, we can reformulate the MAP problem in (A.28) as

$$\hat{\mathcal{X}}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmin}} \left(\sum_{m=1}^M \frac{1}{2} \|\underbrace{\Lambda_m^{1/2} (\mathbf{x}_m - \boldsymbol{\mu}_{x_m})}_{\text{prior residuals}}\|^2 + \sum_{n=1}^N \frac{1}{2} \|\underbrace{\Lambda_n^{1/2} (\mathbf{y}_n - \boldsymbol{\mu}_{y_n})}_{\text{measurement residuals}}\|^2 \right), \quad (\text{A.29})$$

where the matrices $\Lambda_m = \mathbf{Q}_m^{-1}$ and $\Lambda_n = \mathbf{R}_n^{-1}$ are known as *precision matrix*. The mean values $\boldsymbol{\mu}_{x_n}$ are typically obtained from prior information, such as the system's dynamic model. Additionally, we calculate the *pseudo measurement* by transforming the expected state into the measurement space using $\mathbf{C}\mathbf{x}_n$, which serves as the mean for the measurement.

The MAP in (A.29) presents a so-called linear *least-square problem* [1, pp. 44]. By lifting the input variables and measurements from all time steps into a single vector $\mathbf{z}$ and reshaping the system matrices, we can generalize the MAP problem as

$$\hat{\mathcal{X}}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmin}} \frac{1}{2} (\mathbf{z} - \mathbf{H}\mathbf{x})^T \boldsymbol{\Lambda}_k (\mathbf{z} - \mathbf{H}\mathbf{x}), \quad (\text{A.30})$$

where the lifted vector and matrices become

$$\begin{aligned} \mathbf{z} &= [\mathbf{x}_0 \ \mathbf{B}\mathbf{u}_1 \ \dots \ \mathbf{B}\mathbf{u}_M \ \mathbf{y}_0 \ \dots \ \mathbf{y}_N]^T \in \mathbb{R}^{(2+M+N) \times 1}, \\ \mathbf{x} &= [\mathbf{x}_0 \ \dots \ \mathbf{x}_M], \end{aligned}$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & & & & \\ -\mathbf{A}_{D,0} & 1 & & & \\ & \ddots & \ddots & & \\ & & -\mathbf{A}_{D,M-1} & 1 & \\ \mathbf{C}_0 & & & & \\ & \mathbf{C}_1 & & & \\ & & \ddots & & \\ & & & \mathbf{C}_N & \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} \mathbf{Q}_0^{-1} & & & & \\ & \mathbf{Q}_1^{-1} & & & \\ & & \ddots & & \\ & & & \mathbf{Q}_M^{-1} & \\ & & & & \mathbf{R}_0^{-1} \\ & & & & & \mathbf{R}_1^{-1} \\ & & & & & & \ddots \\ & & & & & & & \mathbf{R}_N^{-1} \end{bmatrix}.$$

Obviously, since the linear least square is quadratic, we can find the minimum of the cost function $J(\mathbf{x})$ in (A.30) by setting its partial derivative of $\mathbf{x}$ to zero, which leads us to

$$\left. \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}^T} \right|_{\hat{\mathbf{x}}} = -\mathbf{H}^T \mathbf{\Lambda} (\mathbf{z} - \mathbf{H} \hat{\mathbf{x}}) = 0. \quad (\text{A.31})$$

Therefore, the best point estimate $\hat{\mathbf{x}}$ is calculated as

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{\Lambda} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{\Lambda} \mathbf{z}. \quad (\text{A.32})$$

The calculation above, however, applies only to linear systems. Given that nonlinear dynamics and measurement models govern most real-world systems, extending the least-squares solution to accommodate nonlinear models is necessary. This leads to the development of the *nonlinear least-square estimation* (NLS estimation).

In the presence of nonlinearities, the objective in (A.30) can no longer be explicitly represented by a linear mapping of the states $\mathbf{x}$ using $\mathbf{H}$ to calculate the residuals. Instead, I use a nonlinear function $\mathbf{h}(\mathbf{x})$ to compute the expected quantities from the model. This reformulates the MAP problem as

$$\hat{\mathcal{X}}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{k=0}^K \|\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k)\|_{\Sigma_k}^2. \quad (\text{A.33})$$

Various strategies exist for solving the nonlinear least-squares problem in state estimation applications, including *local optimization*, *simulated annealing*, and *Kalman filters*, the latter of which will be introduced in the next section. Here, we focus on local optimization methods, as they form the foundation for most state-of-the-art state estimation techniques, including the estimator employed in this thesis.

If the system models are linear with Gaussian noise, the objective function in (A.29) forms a perfect paraboloid, ensuring the existence of a *global minimum* can be found.

However, for nonlinear objective functions, a global minimum is no longer guaranteed due to the potential nonconvexity of the cost geometry [203, Chapter 1]. This problem necessitates the use of local optimization methods in practical applications, which iteratively refine a solution starting from an initial guess of the state $\mathbf{x}_0$. While local optimization can efficiently converge to a nearby minimum, it may only find a *local minimum*, especially in cases where the objective function contains multiple minima or saddle points.

Since our objective functions are primarily smooth and differentiable, we can leave the *Gradients* of the objective function $\nabla J(\mathbf{x})$ to determine the descent (search) direction. This gradient serves as a guide for the to-be-estimated states, directing them toward the region of lower cost and ultimately converging closer to the optimal solution.

Algorithm 2 illustrates the general concept of local optimization discussed above, where the steps at lines 4 and 5 vary depending on the specific algorithm employed. Here, I briefly introduce three types of optimization solvers: *Gradient Descent*, *Newton's Methods*, and *Trust-Region Methods*.

Algorithm 2: Local Optimization Algorithm.

Input : Initial guess $\mathbf{x}_0$, tolerance ϵ , maximum iterations K , objective function $J(\mathbf{x})$
Output: Optimized parameter $\mathbf{x}^*$

```

1 Initialize  $\mathbf{x} \leftarrow \mathbf{x}_0$  and  $k \leftarrow 0$ ;
2 repeat
3   Compute gradient  $\nabla J(\mathbf{x})$ ;
4   Compute update direction  $\mathbf{d} \leftarrow -\nabla J(\mathbf{x})$ ;
5   Adapt strategy to compute step size  $\alpha_k$ ;
6   Update  $\mathbf{x} \leftarrow \mathbf{x} + \alpha_k \mathbf{d}$ ;
7    $k \leftarrow k + 1$ ;
8 until  $\|\nabla J(\mathbf{x})\| < \epsilon$  or  $k \geq K$ ;
9 return  $\mathbf{x}^* \leftarrow \mathbf{x}$ 

```

A.2.3.1 Linearization

Before delving into optimization algorithms, it is important to introduce one of the most common techniques for addressing nonlinearity in $\mathbf{h}(\cdot)$ within state estimation methods: *linearization* of $\mathbf{h}(\cdot)$ using *first-order Taylor expansion* at the working point $\mathbf{x}^0$. This results in

$$\mathbf{h}_k(\mathbf{x}_k) = \mathbf{h}_k(\mathbf{x}_k^0 + \boldsymbol{\delta}_k) \approx \mathbf{h}_k(\mathbf{x}_k^0) + \mathbf{J}_k^h \boldsymbol{\delta}_k, \quad (\text{A.34})$$

where we calculate the *Jacobian matrix* $\mathbf{J}_k^h$ as the partial derivative of $\mathbf{h}_k$ as

$$\mathbf{J}_k^h \triangleq \left. \frac{\partial \mathbf{h}_k(\mathbf{x}_k)}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_k^0}. \quad (\text{A.35})$$

The vector $\boldsymbol{\delta}_k = \mathbf{x}_k - \mathbf{x}_k^0$ defines the *perturbation state* (aka. delta state), which is supposed to be small enough to ensure a valid approximation of $\mathbf{h}_k(\mathbf{x}_k)$ in (A.34).

We can now substitute (A.34) into the nonlinear least-square function in (A.33), which gives us

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \sum_{k=0}^K \left\| \mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k^0) - \mathbf{J}_k^h \boldsymbol{\delta}_k \right\|_{\boldsymbol{\Sigma}_k}^2. \quad (\text{A.36})$$

By eliminating (aka. whitening) the covariance matrix $\boldsymbol{\Sigma}_k$ in (A.36), we can reformulate the MAP in (A.36) as

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \sum_{k=0}^K \left\| \mathbf{A}_k \boldsymbol{\delta}_k - \mathbf{b}_k \right\|_2^2, \quad (\text{A.37})$$

$$= \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \left\| \mathbf{A} \boldsymbol{\delta} - \mathbf{b} \right\|_2^2, \quad (\text{A.38})$$

where the whitened Jacobi matrix $\mathbf{A}_k$ and the whitened perturbation state $\mathbf{b}_k$ are calculated as

$$\mathbf{A}_k = \boldsymbol{\Sigma}_k^{1/2} \mathbf{J}_k^h, \quad (\text{A.39})$$

$$\mathbf{b}_k = \boldsymbol{\Sigma}_k^{1/2} (\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k^0)). \quad (\text{A.40})$$

Taking advantage of the linearized, whitened formulation in (A.38), we can leverage *matrix factorization* techniques, such as *QR* or *Cholesky* decomposition, to calculate the derivative of our objective function while solving the nonlinear least squares. I refer the reader to Golub and Loan [204] for a comprehensive explanation of the matrix factorization approaches, including their theoretical foundations and practical implementations.

A.2.3.2 Gradient Descent

Gradient descent utilizes the negative gradient of the objective function to determine the descending direction

$$\mathbf{d}^{\text{SD}} = -\alpha \nabla J(\mathbf{x})|_{\mathbf{x}^0}, \quad (\text{A.41})$$

where the variable α determines the size of the current iteration step. As I introduced in the last section, we can approximate the objective function as $J(\mathbf{x}) \approx \left\| \mathbf{A} \boldsymbol{\delta} - \mathbf{b} \right\|_2^2$ (see (A.36))

and (A.38)). Thus, its gradient of our objective function (A.36) given the working point $\mathbf{x}^0$ becomes⁶

$$\nabla J(\mathbf{x})|_{\mathbf{x}^0} = \mathbf{J}^{\text{NLS}} = -2\mathbf{A}^T \mathbf{b}. \quad (\text{A.42})$$

Although gradient descent is straightforward to implement, its performance is susceptible to the choice of the step size α . A large step size can lead to divergence, while a small step size may significantly slow down the convergence, making the optimization inefficient. To address this issue, the *line search* method is commonly employed to adjust the step size dynamically, transforming the gradient descent approach into the more efficient steepest descent algorithm [205, pp. 5-16].

A.2.3.3 Newton's and Quasi-Newton Methods

Compared to gradient descent, *Newton's methods* enable more efficient computation of the descending step by using a second-order update. Here, the original nonlinear function $\mathbf{h}(\cdot)$ is approximated by the second-order Taylor expansion as

$$\mathbf{h}_k(\mathbf{x}_k) \approx \mathbf{h}_k(\mathbf{x}_k^0) + \mathbf{J}_k \boldsymbol{\delta}_k + \frac{1}{2} \boldsymbol{\delta}_k^T \nabla \mathbf{J}_k \boldsymbol{\delta}_k. \quad (\text{A.43})$$

Similar to the linearization in (A.34), a local minimum of the approximated function (A.43) can be found by setting its derivative with respect to the perturbation state $\boldsymbol{\delta}$ to zero, namely $\mathbf{J} + \nabla \mathbf{J} \boldsymbol{\delta} \triangleq \mathbf{0}$. By substituting the second-order approximation of $\mathbf{h}(\cdot)$ in (A.43) back to the nonlinear objective function (A.33), we can determine Newton's deciding direction as

$$\underbrace{\nabla^2 J(\mathbf{x})|_{\mathbf{x}^0}}_{\text{Hessian } \mathcal{H}} \cdot \mathbf{d}^{\text{N}} = - \underbrace{\nabla J(\mathbf{x})|_{\mathbf{x}^0}}_{\mathbf{J}^{\text{NLS}}}, \quad (\text{A.44})$$

where we define the *Hessian matrix* as the second derivative of our objective function $J(\mathbf{x})$ using

$$\mathcal{H}^{\text{NLS}} = (\mathbf{J}^{\text{NLS}})^T \mathbf{J}^{\text{NLS}} + \sum_{k=0}^K J(\mathbf{x}_k) \nabla^2 J(\mathbf{x}_k). \quad (\text{A.45})$$

Calculating the descending direction using Newton's method offers quadratic convergence by capturing the curvature of the objective function, enabling the algorithm to converge faster than the gradient descent method, particularly as the current estimate approaches the minimum.

However, one may realize that computing the Hessian matrix and its inverse can be particularly challenging in high-dimensional state spaces and may lead to numerical instability

⁶Please notice that $\mathbf{J}^{\mathbf{h}}$ and $\mathbf{J}^{\text{NLS}}$ are different Jacobians.

if the matrix is not well-conditioned. Therefore, practical applications often approximate the Hessian matrix, leading to the development of *Quasi-Newton methods*. A common approach in these methods is to ignore the second term $\nabla^2 J(\mathbf{x}_k)$ in (A.45), under the assumption that it is reasonably small. Together with our calculations in (A.42), we can now calculate Newton's descending direction as

$$\mathbf{d}^{\text{GN}} = ((\mathbf{J}^{\text{NLS}})^T \mathbf{J}^{\text{NLS}})^{-1} (\mathbf{J}^{\text{NLS}})^T \mathbf{b}, \quad (\text{A.46})$$

$$= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (\text{A.47})$$

This approach is known as the *Gauss-Newton* (GN) method, which I utilize later in the proposed state estimator (see Section A.2.7).

A.2.3.4 Damped and Trust-Region Methods

Since (quasi-)Newton's methods typically rely on an approximated second-order derivative to calculate step updates, they often face challenges such as large residuals, ill-conditioned Jacobians, nonconvexity, and poor step size control. To address these issues and improve optimization stability, practical applications commonly employ two robust strategies: *damped* and *trust-region* methods. These approaches prevent divergence and ensure reliable convergence, even in challenging optimization landscapes.

The damped methods regularize the update step to strike a balance between stability and efficiency, particularly in nonlinear least-squares applications. The most famous approach in this category is called *Levenberg-Marquardt* (LM) method, which introduces a damping factor $\lambda \in \mathbb{R}^+ \cup \{0\}$ to combine the Gauss-Newton and gradient descent steps. The updated direction of the LM method adapted to our nonlinear least-square objective function in (A.38) is defined as,

$$\mathbf{d}^{\text{LM}} = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{b}, \quad \text{or} \quad (\text{A.48})$$

$$= (\mathbf{A}^T \mathbf{A} + \lambda \cdot \text{diag}(\mathbf{A}^T \mathbf{A}))^{-1} \mathbf{A}^T \mathbf{b}, \quad (\text{A.49})$$

where the variant of LM in (A.49) was introduced by Marquardt [206] to speed the convergence.

The damping factor λ controls the update direction by balancing between second-order and first-order methods. The LM method behaves like the Gauss-Newton method if $\lambda = 0$. Conversely, when an update increases the sum of squared residuals, the LM method identifies this as a sign of poor local behavior in the nonlinear function and adjusts by shrinking the step size. This is achieved by increasing the damping factor λ , often by multiplying it by a fixed scaling parameter (e.g., 10) and re-solving the system with a more constrained step.

Another category to overcome the drawbacks of Newton's methods is to define a trusted region of the approximated objective function that constrains the optimization so that we have

$$\boldsymbol{\delta}^{\text{TR}} = \underset{\|\boldsymbol{\delta}\|^2 \leq \Delta}{\operatorname{argmin}} J(\boldsymbol{\delta}). \quad (\text{A.50})$$

The most common trust-region method is called Powell's *dogleg* (DL) algorithm, which explicitly combines the Gauss-Newton and LM steps within the trust region Δ controlled by the gain ratio

$$\varrho = \frac{J(\mathbf{x}_k) - J(\mathbf{x}_k + \boldsymbol{\delta}_k)}{L(\mathbf{0} - L(\boldsymbol{\delta}))}, \quad (\text{A.51})$$

where we approximate the nonlinear quadratic objective function $J(\mathbf{x})$ in (A.33) using the second-order Taylor expansion as $L(\cdot)$. In the context of nonlinear least square shown in (A.37)-(A.40), we can equivalently get $L(\boldsymbol{\delta}) = \mathbf{A}^T \mathbf{A} \boldsymbol{\delta} - \mathbf{A}^T \mathbf{b}$.

Figure A.3 illustrates how the dogleg algorithm works. It is constructed by initially moving along the steepest descent direction $\mathbf{d}^{\text{SD}}$ and then transitioning sharply towards the Gauss-Newton direction $\mathbf{d}^{\text{GN}}$. This "dogleg" path ensures the step remains within the trust region Δ , halting at its boundary if the GN step exceeds the region's limits.

Unlike the LM method, the dogleg algorithm explicitly enforces the trust region to maintain the validity of the linear approximation. The trust region Δ is adjusted based on the gain ratio ϱ in (A.51). A smaller ϱ (e.g., $\varrho < 0.25$), the trust region Δ should be decreased to constrain the update step due to inaccurate linear approximation of the objective function, preventing the optimization from divergence. Conversely, if ϱ is sufficiently large (e.g., $\varrho > 0.75$), it suggests that the cost function has decreased as expected or even better. The trust region is then expanded based on the size of the update vector, and the step is accepted.

Therefore, the update step of the dogleg method can be summarized as

$$\mathbf{d}^{\text{DL}} = \begin{cases} \mathbf{d}^{\text{GN}}, & \text{if } \|\boldsymbol{\delta}^{\text{GN}}\| \leq \Delta, \\ \Delta \cdot \frac{\mathbf{d}^{\text{SD}}}{\|\mathbf{d}^{\text{SD}}\|}, & \text{if } \|\boldsymbol{\delta}^{\text{SD}}\| \geq \Delta, \\ \mathbf{d}^{\text{SD}} + \beta(\mathbf{d}^{\text{GN}} - \mathbf{d}^{\text{SD}}), & \text{otherwise,} \end{cases} \quad (\text{A.52})$$

where the scaling factor β can be calculated as $\beta = \frac{\Delta^2 - \|\mathbf{d}^{\text{SD}}\|^2}{(\mathbf{d}^{\text{GN}} - \mathbf{d}^{\text{SD}})^T (\mathbf{d}^{\text{GN}} - \mathbf{d}^{\text{SD}})}$.

Although the dogleg method introduces additional computational overhead compared to the Levenberg-Marquardt method, it offers improved stability by explicitly maintaining a trust region. The choice between the Gauss-Newton, Levenberg-Marquardt, and dogleg methods depends on practical considerations, such as the problem's dimensionality, the conditioning of the Jacobian, the magnitude of residuals, and the available computational resources. I

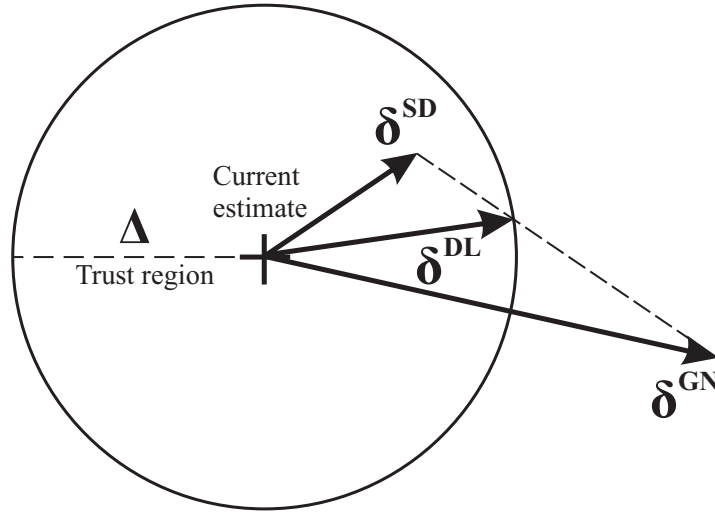


Figure A.3: Illustration of the iterative state update using gradient descent (SD), Gauss-Newton (GN), and Powell's dogleg (DL) methods.

refer the reader to Nocedal and Wright [203] for a comprehensive introduction to all the optimization methods discussed above.

In this thesis, the Gauss-Newton method is the primary approach in our incremental solver. However, the LM and dogleg methods are optional configurations that enhance flexibility and stability in specific scenarios.

A.2.4 Kalman Filter Family

Originally developed by Kalman [207], the Kalman filter provides an efficient recursive filtering-based solution to the problem of estimating the state of a linear system from dynamic models and noisy measurements. The Kalman filter can be interpreted in multiple ways. Kalman [207] introduced the concept of *orthogonal projection*, which involves projecting the predicted state onto the measurement subspace under the assumption that both are Gaussian. Alternatively, other approaches frame the Kalman filter using optimization techniques, such as *gain optimization* [1, Chapter 3.3.4] and *MAP estimation* [1, Chapter 3.3.2], providing additional perspectives on its underlying principles.

In this thesis, I adopt the perspective of Thrun et al. [3] and view the Kalman filter as a derivative of the *Bayes filter*, as it provides the most intuitive framework for gaining a profound and comprehensive understanding of its principles and operations.

A.2.4.1 Bayes Filter

A Bayes filter establishes a recursive procedure to infer the *Belief* of the system state based on Bayes's theorem (A.18) given noisy prior information and measurements. I introduce the term *Belief* $bel(\cdot)$ to represent the system's internal knowledge while interacting with its environment.

In the context of state estimation, we can leverage Bayes's theorem to define the belief of the expected (aka predicted) $bel(\mathbf{x}^-)$ and the belief of the estimated (aka observed) state $bel(\mathbf{x}^+)$ as

$$bel(\mathbf{x}_k^-) = \int p(\mathbf{x}_k^- | \mathbf{u}_k, \mathbf{x}_{k-1}^+) \cdot bel(\mathbf{x}_{k-1}^+) d\mathbf{x}_{k-1}^+, \quad (\text{A.53})$$

$$bel(\mathbf{x}_k^+) = \eta \cdot p(\mathbf{z}_k | \mathbf{x}_k^-) bel(\mathbf{x}_k^-), \quad (\text{A.54})$$

where the parameter η represents the evidence (aka normalization constant) in (A.18). Then, we can summarize the Bayes filter in Algorithm 3, which consists of an endless loop alternating between the time update (also known as the prediction step) and the measurement update (also known as the correction step). For a more intuitive understanding, we can illustrate this procedure in Figure A.4, serving as the foundation for all practical algorithms designed for recursive state estimation.

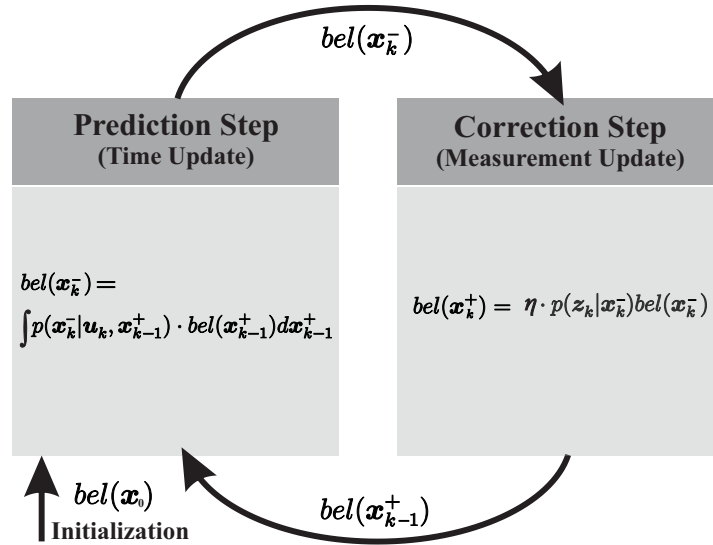


Figure A.4: Schema of the recursive Bayes filter.

A.2.4.2 Kalman Filter

Clearly, implementing the Bayes filter in closed form is challenging due to the integrations required in lines 2 and 3, as discussed earlier in Section A.2.2. To address this, the Kalman

Algorithm 3: Bayes Filter

Input : Previous belief $bel(\mathbf{x}_{k-1}^+)$, control input $\mathbf{u}_k$, measurement $\mathbf{z}_k$
Output : Updated belief $bel(\mathbf{x}_k^+)$

```

1 for all  $\mathbf{x}_k$  do
2   Time Update: Compute the expected belief based on the control input:
      
$$bel(\mathbf{x}_k^-) = \int p(\mathbf{x}_k^- | \mathbf{u}_k, \mathbf{x}_{k-1}^+) \cdot bel(\mathbf{x}_{k-1}^+) d\mathbf{x}_{k-1}^+$$

3   Measurement Update: Incorporate the measurement to refine the belief:
      
$$bel(\mathbf{x}_k^+) = \eta \cdot p(\mathbf{z}_k | \mathbf{x}_k^-) bel(\mathbf{x}_k^-)$$

      where  $\eta$  is a normalization constant.
4 end for
5 return  $bel(\mathbf{x}_k^+)$ 

```

filter⁷ emerges as a practical and prominent technique for implementing the Bayes filter by assuming Gaussian distributions for both the prior and likelihood, which stands alongside the optimization algorithms introduced in Section A.2.3.

By adapting the framework of the Bayesian filter, we again leverage the state-space model (Section A.2.1) to instantiate the beliefs in (A.53)-(A.54) as $bel(\mathbf{x}_k) \sim \mathcal{N}(\mathbf{x}_k, \Sigma_k)$ ⁸ and set up the Kalman filter with the following equations

$$\mathbf{x}_k^- = \mathbf{A}_D \mathbf{x}_{k-1}^+ + \mathbf{B} \mathbf{u}_k, \quad (\text{A.55})$$

$$\Sigma_k^- = \mathbf{A}_D \Sigma_{k-1}^+ \mathbf{A}_D^T + \mathbf{Q}_k, \quad (\text{A.56})$$

$$\mathbf{K}_k = \Sigma_k^- \mathbf{C}^T (\mathbf{C} \Sigma_k^- \mathbf{C}^T + \mathbf{R}_k)^{-1}, \quad (\text{A.57})$$

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{C}_k \mathbf{x}_k^-), \quad (\text{A.58})$$

$$\Sigma_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \Sigma_k^-. \quad (\text{A.59})$$

The equations (A.55) and (A.56) are used to propagate the system's state and uncertainty given the system transition matrix $\mathbf{A}_D$ and the pre-defined covariance matrix $\mathbf{Q}_k$. The matrix $\mathbf{K}_k$ is called the *Kalman gain*, which determines a “trust” ratio between the system propagation $\{\mathbf{x}_k^-, \Sigma_k^-\}$ and observation $\{\mathbf{z}_k, \mathbf{R}_k\}$. We employ the Kalman gain to calculate the *Kalman innovation* $\mathbf{K}_k (\mathbf{z}_k - \mathbf{C}_k \mathbf{x}_k^-)$, which serves as an update to the propagated state in (A.58). Similarly, the propagated covariance can be updated as in (A.59). Then, the Kalman filter can be implemented as in Algorithm 4.

⁷In fact, the Kalman filter is equivalent to linear least-squares estimation derived from a MAP perspective [1], and it can also be implemented in a recursive form [208]. However, I deliberately highlight the MAP framework within the context of optimization algorithms to maintain clarity.

⁸I use the term $\mathbf{x}_k$ instead of μ_k for the sake of clarity.

Algorithm 4: Kalman Filter

Input : Previous estimate $\{\mathbf{x}_{k-1}^+, \Sigma_{k-1}^+\}$, control input $\mathbf{u}_k$, measurement z_k
Output: Current estimate $\{\mathbf{x}_k^+, \Sigma_k^+\}$

- 1 **for all** $\mathbf{x}_k$ **do**
- 2 **Time Update:**

$$\mathbf{x}_k^- = \mathbf{A}_D \mathbf{x}_{k-1}^+ + \mathbf{B} \mathbf{u}_k$$

$$\Sigma_k^- = \mathbf{A}_D \Sigma_{k-1}^+ \mathbf{A}_D^T + \mathbf{Q}_k$$
- 3 **Compute the Kalman Gain:**

$$\mathbf{K}_k = \Sigma_k^- \mathbf{C}^T (\mathbf{C} \Sigma_k^- \mathbf{C}^T + \mathbf{R}_k)^{-1}$$
- 4 **Measurement Update:**

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k (z_k - \mathbf{C}_k \mathbf{x}_k^-)$$

$$\Sigma_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \Sigma_k^-$$
- 5 **end for**
- 6 **return** $\{\mu_k^+, \Sigma_k^+\}$

To gain a deeper understanding of the Kalman filter, we ask why the Kalman gain can be used to correct the predicted state. Since the Kalman equations in (A.55)-(A.59) are derived under the assumption of multivariate Gaussian distributions, we simplify the scenario by assuming that both the state and measurement are Gaussian-distributed random scalars in the same space. By reformulating the equations from (A.11) while multiplying two Gaussians, we obtain

$$\mu_x^+ = \frac{\mu_x^- / \sigma_x^2 + \mu_z / \sigma_z^2}{1 / \sigma_x^2 + 1 / \sigma_z^2}, \quad (\text{A.60})$$

$$= \frac{\sigma_x^2 \sigma_z^2}{\sigma_x^2 + \sigma_z^2} \left(\frac{\mu_x^-}{\sigma_x^2} + \frac{\mu_z}{\sigma_z^2} \right), \quad (\text{A.61})$$

$$= \mu_x^- \left(\frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} - \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} \right) + \mu_z \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2}, \quad (\text{A.62})$$

$$= \mu_x^- + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} (\mu_z - \mu_x^-), \quad (\text{A.63})$$

where the last equation (A.63) interestingly carries the same form as the Kalman equation (A.58). Therefore, we can intuitively get the idea of the Kalman filter, which employs the pre-defined covariance matrix of two Gaussians to formulate a weighting factor $\sigma_x^2 / (\sigma_x^2 + \sigma_z^2)$ (aka Kalman gain) and finds the best state μ_x^+ corrected by the weighted difference between

the propagated state μ_x^- and its observation μ_z . As we observed earlier in Section A.2.2, the multiplication of two Gaussian results in a new Gaussian with a smaller variance, proving that the Kalman filter inherently acts as a variance-minimizing estimator.

Finally, we can summarize the assumptions (often considered as limitations) of the (linear) Kalman filter from Algorithm 4 [3]:

- The system transition and measurement models must be **linear**,
- The noise process of the models must be **zero-mean Gaussian**,
- The **initial belief** must be zero-mean Gaussian,

which, however, cannot be fulfilled in real-world applications. Therefore, we need to extend the Kalman equations to handle non-linearities, for example.

A.2.4.3 Extended Kalman Filter

One straightforward way to implement the Kalman filter for nonlinear systems is to linearize the nonlinear equations using first-order Taylor expansion, as introduced in Section A.2.3. This concept is called *Extended Kalman Filter* (EKF), where we use implicit equations

$$\mathbf{x}_k^- = \mathbf{f}(\mathbf{u}_k, \mathbf{x}_{k-1}^+), \quad (\text{A.64})$$

$$\hat{\mathbf{z}}_k = \mathbf{h}(\mathbf{x}_k^-) \quad (\text{A.65})$$

to represent the nonlinear state propagation and measurement (pseudo-measurement) models, respectively. By linearizing (see Section A.2.3) the nonlinear functions (A.64) and (A.65), we can adapt the original Kalman filter into the Extended Kalman filter in Algorithm 5, where we calculate the first-order derivatives (Jacobians $\mathbf{J}$) for the nonlinear propagation function $\mathbf{f}(\cdot)$ and observation function $\mathbf{h}(\cdot)$ to approximately feature the nonlinearities into Kalman filter. The reader may have noticed that we used different working points while linearizing both nonlinear functions. While we propagate the uncertainties using the linearized system transition matrix $\mathbf{J}_{f,k}$ from the last (estimated) state $\mathbf{x}_{k-1}^+$, the Jacobian matrix $\mathbf{J}_{h,k}$ is required to project the expected state $\mathbf{x}_k^-$ into the measurement space and the Kalman gain $\mathbf{K}_k$ into the state space to correct the propagations $\{\mathbf{x}_k^-, \Sigma_k^-\}$.

Although the EKF offers a straightforward approach to handling nonlinear state estimation, its performance heavily depends on achieving a careful balance between the degree of nonlinearity and the magnitude of uncertainty, as discussed in Van Der Merwe [198, Chapter 2.5]. The first-order linearization using the Taylor expansion inherently neglects higher-order terms. However, as the degree of nonlinearity increases or the variance becomes more extensive, these neglected terms become increasingly significant, thereby reducing the accuracy of the EKF's estimates.

Algorithm 5: Extended Kalman Filter

Input : Previous estimate $\{\mathbf{x}_{k-1}^+, \Sigma_{k-1}^+\}$, control input $\mathbf{u}_k$, measurement $\mathbf{z}_k$
Output: Current estimate $\{\mathbf{x}_k^+, \Sigma_k^+\}$

- 1 **for all** $\mathbf{x}_k$ **do**
- 2 **Time Update:**

$$\mathbf{x}_k^- = \mathbf{f}(\mathbf{x}_{k-1}^+, \mathbf{u}_k)$$

$$\mathbf{J}_{f,k} = \left. \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}^+, \mathbf{u}_k)}{\partial \mathbf{x}_{k-1}^+} \right|_{\mathbf{x}^0 = \mathbf{x}_{k-1}^+}$$

$$\Sigma_k^- = \mathbf{J}_{f,k} \Sigma_{k-1}^+ \mathbf{J}_{f,k}^T + \mathbf{Q}_k$$
- 3 **Compute the Kalman Gain:**

$$\mathbf{J}_{h,k} = \left. \frac{\partial \mathbf{h}(\mathbf{x}_k^-)}{\partial \mathbf{x}_k^-} \right|_{\mathbf{x}^0 = \mathbf{x}_k^-}$$

$$\mathbf{K}_k = \Sigma_k^- \mathbf{J}_{h,k}^T (\mathbf{J}_{h,k} \Sigma_k^- \mathbf{J}_{h,k}^T + \mathbf{R}_k)^{-1}$$
- 4 **Measurement Update:**

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k^-))$$

$$\Sigma_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{J}_{h,k}) \Sigma_k^-$$
- 5 **end for**
- 6 **return** $\{\mathbf{x}_k^+, \Sigma_k^+\}$

A.2.4.4 Unscented Kalman Filter

Since the first-order linearization introduces limitations in EKF, we may consider employing other techniques to transform the nonlinearity while implementing the Kalman filter. One prominent method is the *Unscented Transform* (UT) introduced by Julier and Uhlmann [209], which employs a finite set of deterministic samples (aka sigma points) propagated by the nonlinear function to capture the uncertainty. Using this method, we can develop another Kalman filter derivative: *Unscented Kalman Filter* (UKF), also named the sigma-point Kalman filter.

Algorithm 6 shows the unscented transform, where $2D+1$ sigma points for a D -dimensional mean vector are generated following the method described by Van Der Merwe [198], as this method is widely used in practical applications. New sigma points (excluding $\mathbf{x}_0$) are generated by adding and subtracting offsets calculated as the scaled square root of the

Algorithm 6: Unscented Transform (UT)

Input : Input mean μ , covariance matrix Σ , nonlinear function $f(\cdot)$, scaling parameter κ , noise covariance Q

Output: Transformed state $\hat{\mu}$ and covariance $\hat{\Sigma}$

1 **Step 1:** Generate sigma points following Julier and Uhlmann [209]

2 $D \leftarrow$ dimension of μ

3 $x_0 \leftarrow \mu$

4 **for** $d = 1, \dots, D$ **do**

5

$$\begin{aligned} x_d &\leftarrow x_0 + \sqrt{(D + \kappa)\Sigma_d} \\ x_{d+D} &\leftarrow x_0 - \sqrt{(D + \kappa)\Sigma_d} \end{aligned}$$

6 **end for**

7 **Step 2:** Propagate sigma points through the nonlinear function

8 **for** $d = 0, \dots, 2D$ **do**

9

$$y_d \leftarrow f(x_d)$$

10 **end for**

11 **Step 3:** Compute transformed mean

12

$$\hat{\mu} \leftarrow \sum_{d=0}^{2D} w_d^m y_d$$

13 **Step 4:** Compute Transformed Covariance

14

$$\hat{\Sigma} \leftarrow \sum_{d=0}^{2D} w_d^c (y_d - \hat{\mu})(y_d - \hat{\mu})^T + Q$$

return $\{\hat{\mu}, \hat{\Sigma}\}$

predefined covariance matrix Σ (line 2-6). This ensures that the sigma points are symmetrically distributed around the mean, effectively capturing the spread and shape of the original (symmetric) distribution.

The sigma points are propagated using the nonlinear function to calculate the transformed sigma points $y = [y_0 \dots y_{2D+1}]^T$. Since the transformed sigma points represent the state propagation disturbed under varying disturbance (variance) levels, the final mean vector is computed as a weighted sum of these transformed sigma points. Here, Van Der Merwe [198]

introduced a scaled unscented transformation, where the parameters w^m and w^c are used to calculate the mean and covariance, respectively. The weighting parameters are defined as

$$\begin{aligned} w_0^m &= \frac{\lambda}{D + \lambda}, & d = 0, \\ w_0^c &= \frac{\lambda}{D + \lambda} + (1 - \alpha^2 + \beta), & d = 0, \\ w_d^c &= w_d^m = \frac{1}{2(D + \lambda)} & d = 1, \dots, D, \end{aligned}$$

where we calculate the parameter $\lambda = \alpha^2(D + \kappa) - D$. The hyperparameters⁹ κ , α , and β must be selected for particular applications to control the position and spreading of the sigma points.

We can now present the UKF Algorithm 7 by leveraging the UT in Algorithm 6. The major difference between the UKF and the EKF lies in the uncertainty propagation, where the covariance matrices of the predicted state and pseudo-measurement are calculated based on the deviations of the sigma points from their mean. This eliminates the need to linearize the nonlinear functions (hence the term 'unscented'), allowing for a more accurate and efficient update to the predictions without requiring Jacobian computations. Regardless of the distribution shape of the models, various strategies can be employed to generate sigma points. Popular methods include the simple sigma points proposed by Julier and Uhlmann [209] and the simplex sigma points introduced by Julier [211], which are particularly effective in propagating uncertainty from asymmetric distributions.

The UKF is widely regarded as a powerful method for nonlinear state estimation, particularly in scenarios where large uncertainties are present in the system models, as demonstrated in numerous studies [198, 209, 212]. However, its performance remains highly dependent on selecting hyperparameters for generating and weighting the sigma points, making sophisticated tuning essential for different applications.

A.2.4.5 Discussion on Kalman Filters

In this section, I have briefly introduced the Kalman filter and its popular derivatives. However, numerous additional Kalman variants and filtering algorithms fall under the broader framework of the Bayes filter. For instance, the ensemble Kalman filter offers a more flexible approach to sampling sigma points [213], while the iterated Kalman filter performs multiple iterations during the update step to refine the linearization and select a more accurate working point [1, Chapter 4.2.6]. Recent works proposed invariant Kalman filters [155, 214], where the state and uncertainty propagation are processed on a manifold instead of in vector space.

⁹Since the UKF is not the primary focus of this thesis, I do not provide a comprehensive discussion on parameter tuning. For a more detailed and practical guide, I refer the reader to Van Der Merwe [198, Chapter 3.2.2] and Labbe [210, Chapter 10].

Algorithm 7: Unscented (Sigma-Points) Kalman Filter

Input : Previous estimate $\{\mathbf{x}_{k-1}^+, \Sigma_{k-1}^+\}$, control input $\mathbf{u}_k$, measurement $\mathbf{z}_k$, parameters α, β , and κ

Output: Current estimate $\{\mathbf{x}_k^+, \Sigma_k^+\}$

```

1 for all  $\mathbf{x}_k$  do
2   Init:  $\chi = \{\emptyset\}, \chi' = \{\emptyset\}$ , and  $\mathbf{Y} = \{\emptyset\}$ 
3   Time Update:
4     /* Generate sigma points */
5      $\chi \leftarrow \mathbf{x}_{k-1,0} = \mathbf{x}_{k-1}^+$ ,  $d = 0$ 
6      $\chi \leftarrow \mathbf{x}_{k-1,d} = \mathbf{x}_{k-1,0} + \sqrt{(D + \kappa)\Sigma_{k-1}}$ ,  $d = 1, \dots, D$ 
7      $\chi \leftarrow \mathbf{x}_{k-1,d+D} = \mathbf{x}_{k-1,0} - \sqrt{(D + \kappa)\Sigma_{k-1}}$ ,  $d = D + 1, \dots, 2D$ 
8     /* Propagate sigma points through the process model */
9      $\chi' \leftarrow \mathbf{x}'_{k,d} = \mathbf{f}(\mathbf{x}_{k-1,d}, \mathbf{u}_k)$ ,  $d = 0, \dots, 2D$ 
10    /* Compute predicted mean */
11     $\mathbf{x}_k^- = \sum_{d=0}^{2D} w_d^m \mathbf{x}'_{k,d}$ 
12    /* Compute predicted covariance */
13     $\Sigma_k^- \leftarrow \sum_{d=0}^{2D} w_d^c (\mathbf{x}'_{k,d} - \mathbf{x}_k^-)(\mathbf{x}'_{k,d} - \mathbf{x}_k^-)^T + \mathbf{Q}$ 
14    Compute the Kalman Gain;
15    /* Transform sigma points through the measurement model */
16     $\mathbf{Y} \leftarrow \mathbf{y}_{k,d} = \mathbf{h}(\mathbf{x}_{k-1,d})$ ,  $d = 0, \dots, 2D$ 
17    /* Compute pseudo-measurement */
18     $\hat{\mathbf{y}}_k = \sum_{d=0}^{2D} w_d^m \mathbf{y}_{k,d}$ 
19    /* Compute pseudo-measurement covariance */
20     $\mathbf{S}_k = \sum_{i=0}^{2D} w_i^c (\mathbf{y}_{k,i} - \hat{\mathbf{y}}_k)(\mathbf{y}_{k,i} - \hat{\mathbf{y}}_k)^T + \mathbf{R}$ ;
21    /* Compute cross-covariance */
22     $\mathbf{C}_k = \sum_{i=0}^{2D} w_i^c (\mathbf{x}'_{k,i} - \mathbf{x}_k^-)(\mathbf{y}_{k,i} - \hat{\mathbf{y}}_k)^T$ 
23    /* Compute Kalman gain */
24     $\mathbf{K}_k = \mathbf{C}_k \mathbf{S}_k^{-1}$ ;
25    Measurement Update:
26    /* Update state estimate */
27     $\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_k (\mathbf{z}_k - \hat{\mathbf{y}}_k)$ ;
28    /* Update covariance */
29     $\Sigma_k^+ = \Sigma_k^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T$ ;
30 end for
31 return  $\{\mathbf{x}_k^+, \Sigma_k^+\}$ 

```

Nevertheless, as previously discussed regarding the assumptions of the Kalman filter, all its variants still rely on the underlying noise being Gaussian (ideally, zero-mean Gaussian). Since this assumption is frequently violated in real-world applications, as discussed in Chapter 1,

Kalman filters are often replaced by alternative filtering methods, such as the particle filter [3, Chapter 4], which are better suited for handling non-Gaussian noise.

Although Kalman filters have been extensively studied for decades, with numerous works demonstrating their success in achieving accurate and real-time state estimation in practical applications and commercial products, they remain inherently limited by their strong assumptions. These limitations become particularly apparent when addressing the need for scalability to integrate advanced robustification techniques into the filter without excessive over-engineering. In this thesis, I will use the Unscented Kalman Filter as a baseline method alongside our newly proposed state estimator based on graph optimization, which will be introduced in a later chapter (Chapter 3). Through extensive discussions and real-world experiments, I aim to answer the question:

When is a Kalman filter a suitable choice for state estimation, and in what scenarios might it be unsuitable?

In the upcoming chapter, where another prominent state estimation technique, called *smoothing*, is introduced, I hope the reader will find the answer through my discussions on the differences between filtering and smoothing.

A.2.5 Filtering versus Smoothing

So far, I have introduced the Bayes filter and its practical implementations, which establish a recursive process where each new epoch uses the previous estimate as the working point, for e.g., state propagation. These approaches are classified as *filtering-based* methods, which aim to determine the best estimate by relying on single-time-step state propagation and noise measurement [215].

I refer to this characteristic in the filtering process as the *Markov property* or a *first-order Markov process*, which states that the estimation of the current belief $bel(\mathbf{x}_k)$ of the system depends only on the previous belief $bel(\mathbf{x}_{k-1})$ [1, Chapter 4.2.1]. Recalling the Kalman filter implementations in Algorithms 4, 5, and 7, where the previous state is used as the working point to propagate or linearize the system models, it becomes evident that other past estimates, control inputs, and measurements are effectively “filtered out” during the process.

While the Markov property assumes that the previous belief optimally represents the system’s knowledge, we now question whether this assumption can be consistently upheld in practical applications. In fact, as discussed in Section A.2.4, ensuring the optimality of state estimation is particularly challenging, as it can be compromised by factors such as imperfect models, inconsistent noise processes, or sub-optimal parameter tuning.

This raises the question of whether leveraging all past information (e.g., past measurements) could improve the accuracy and consistency of state estimation. In this context, methods that

utilize a batch of state information for estimation are referred to as *smoothing-based* methods, which are capable of extending the state estimator beyond the Markov property.

Compared to Bayes filters, smoothing-based methods reconstruct all states whenever new information (e.g., sensor observation) is received. This can be accomplished by recursively updating all previous estimates $\{\mathbf{x}_k, \Sigma_k\}, k = K - 1, \dots, 0$ in a backward pass, starting from the current state estimate $\{\mathbf{x}_K, \Sigma_K\}$. The so-called *Rauch-Tung-Striebel smoother* (aka Kalman smoother) is developed based on this idea [215, Chapter 8.2]. Technically, the backward pass of the Kalman smoother allows for a recursive correction of previous estimates following each forward pass (Kalman filtering). This process yields a smoother trajectory compared to single-time-step estimation, particularly when the estimates are influenced by imperfections in the system models.

Since the Kalman smoother still relies on the Markov property, we can extend the concept of least-squares estimation introduced in Section A.2.3 to develop a *Batch Optimization* approach. As shown in equations (A.29) and (A.33), the MAP problem can be formulated as a least-squares optimization, where all relevant variables (e.g., states and measurements) and matrices (e.g., Jacobian and covariance matrices) are lifted into a unified linear system. Optimization techniques introduced in Section A.2.3 are employed to iteratively solve the nonlinear least-square problem, making batch optimization widely adopted in many robotic applications.

We can now delve deeper into batch optimization and explore its advantages, particularly in extending beyond the limitations of the Markov assumption. If we recall the way of solving the nonlinear least-square in (A.33) using any of the three optimization methods (e.g., Gauss-Newton) introduced in Section A.2.3, we roughly come to a linear update as

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{A}^T \mathbf{b}, \quad (\text{A.66})$$

where the vector $\mathbf{b}$ in (A.39) and the matrix $\mathbf{A}$ in (A.40) are the whitened perturbation state and Jacobi matrix, respectively. The step size parameter α can be replaced by the inverse of the Hessian matrix, $\alpha = (\mathbf{A}^T \mathbf{A})^{-1}$, when using the Gauss-Newton method.

Now, let us consider the changes to $\mathbf{b}$ and $\mathbf{A}$ during each iteration of solving the nonlinear least-squares problem (see Algorithm 2). In each iteration, we update the current estimate $\mathbf{x}$ using (A.66) and employ it to compute a new vector $\mathbf{b}$ and matrix $\mathbf{A}$ for the subsequent optimization epoch. This process implies that we are *re-linearizing* the nonlinear system models and re-calculating the perturbation state for all timestamps within a single batch, which implicitly combines a forward filtering and backward smoothing process within each optimization epoch. Compared to filtering approaches that follow the first-order Markov process, this equivalent “backward pass” does not necessarily follow the state ordering on a Markov chain. Instead, since the matrix $\mathbf{A}$ encapsulates all relevant state information in a single structure, the iterative solver progresses by performing *matrix elimination* to compute the state update.

I will introduce this technique in the next section. Simply put, the construction of the linear system and the method for matrix elimination can be flexibly tailored in batch optimization. Therefore, the most noteworthy advantage of batch optimization, aside from its performance improvements, is *its extensive capacity to model and solve least-squares problems* [102], which is particularly valuable for this thesis. Further discussions on this topic has been presented in Chapter 3.

A.2.6 Inference using Graphical Models

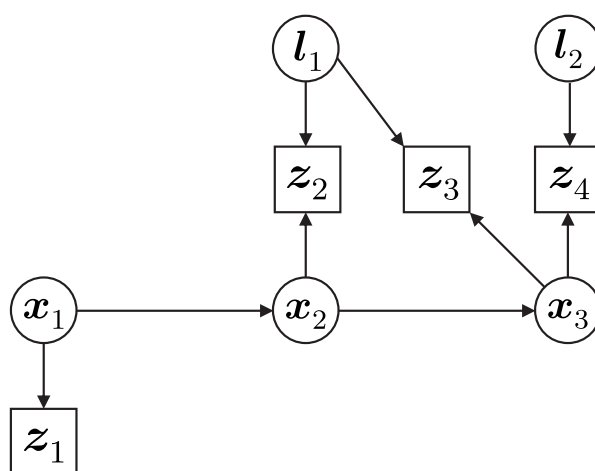
To develop a batch-optimization-based method, it is essential to establish a structured model of the system's belief, where the states to be estimated and their related variables are interconnected. In this context, we turn to probabilistic graphical models $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, which enable the systematic representation of dependencies among variables $\mathbf{V}$ connected by edges $\mathbf{E}$, providing a structured framework for modeling the system's belief and facilitating efficient computation in batch optimization. Among the various probabilistic graphical models, *Bayesian network* (aka Bayes net), *Markov random field* (aka Markov network), and *factor graph*, as illustrated in Figure A.5, are the most widely used graphical models in inference problems.

A.2.6.1 Bayesian Network

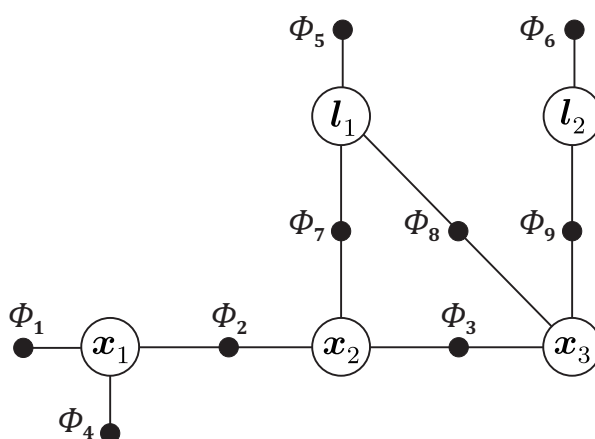
The Bayesian network is a *direct acyclic graph* that presents the joint probability distribution of a set of random variables $\mathcal{X} = \{x^1 \dots x^k\}$ and belongs to the most popular graphical language model in autonomous systems [216, Chapter 3.2]. The *nodes* in a Bayesian network represent random variables, which are connected by directed *edges* to indicate conditional independence between neighboring nodes. A Bayesian network formulates a joint probability density $p(\mathcal{X})$ over all nodes as

$$p(\mathcal{X}) \triangleq \prod_k p(x_k | x'_k), \quad (\text{A.67})$$

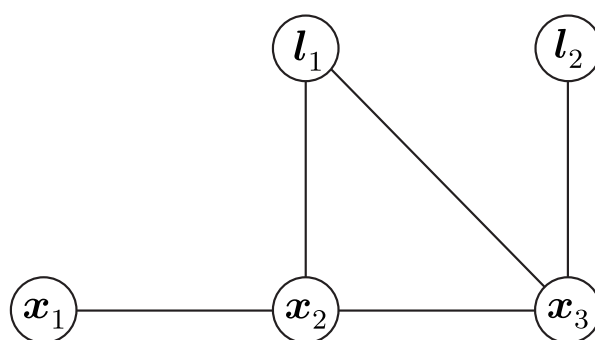
where the variable x'_k denotes the parent node of x_k . This formulation allows the Bayesian network to factorize the joint density within its graph structure, explicitly capturing the probabilistic relationships between nodes. Consider the example in Figure A.5a, which



(a) Toy example of a Bayesian network.



(b) Toy example of a factor graph.



(c) Toy example of a Markov random field.

Figure A.5: Toy examples of three graphical models.

illustrates a toy example of landmark-based localization with the following joint distribution

$$p(\mathcal{X}|\mathcal{Z}) \propto \underbrace{p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_2)}_{\text{priors of state propagation}} \quad (\text{A.68})$$

$$\times \underbrace{p(\mathbf{l}_1)p(\mathbf{l}_2)}_{\text{priors of landmark}} \quad (\text{A.69})$$

$$\times \underbrace{p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{z}_2|\mathbf{x}_2, \mathbf{l}_1)p(\mathbf{z}_3|\mathbf{x}_3, \mathbf{l}_1)p(\mathbf{z}_4|\mathbf{x}_3, \mathbf{l}_2)}_{\text{likelihoods}}. \quad (\text{A.70})$$

We can determine how to calculate the belief of the state variable $\mathbf{x}_3$ by marginalizing its associated measurements $\mathbf{z}_4$ and $\mathbf{z}_3$ giving the neighboring state $\mathbf{x}_2$ and the landmark $\mathbf{l}_2$, which gives us

$$p(\mathbf{x}_3) = \eta \cdot (p(\mathbf{z}_3|\mathbf{l}_1, \mathbf{x}_3) + p(\mathbf{z}_4|\mathbf{l}_2, \mathbf{x}_3))p(\mathbf{x}_3|\mathbf{x}_2) \quad (\text{A.71})$$

A.2.6.2 Markov Random Field

On the contrary, the Markov random field (MRF) uses an *undirected graph* to represent dependencies, focusing on mutual relationships between variables based on the Markov property [216, Chapter 4]. It factorizes the joint distribution over *cliques* $\mathcal{C}$ (aka fully connected subgraphs), which results in

$$p(\mathcal{X}) \propto \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c), \quad (\text{A.72})$$

where the potential function $\psi(\cdot)$ quantifies a particular affinity (e.g., Gaussian) over the clique c .

Unlike Bayesian networks, the MRF does not impose directionality, which means it does not explicitly capture causal structures. As a result, it is less commonly used in state estimation problems to model a robot's belief given, for example, observations. However, I introduce the MRF here because it will reappear in a temporal step, derived from both the Bayesian network and the factor graph, while inferring the posterior distribution.

A.2.6.3 Factor Graph

We will consider both of the aforementioned graphs to model the state estimation problem. However, the inference causality of the to-be-estimated states, given observed landmarks, is often undetermined due to uncertainties. For example, we cannot marginalize the posterior distribution of $\mathbf{x}_3$ if the likelihood $p(\mathbf{z}_4|\mathbf{x}_3, \mathbf{l}_2)$ cannot be calculated when the landmark state is unknown or the noise models are inconsistent. Consequently, we require a more flexible

graphical model that relaxes strict causal dependencies while substantiating the relationships between variables. This leads to the adoption of the factor graph.

A factor graph is a bipartite graph $\mathcal{F} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ with factor nodes $\phi_i \in \mathcal{U}$ and state variable nodes $\mathbf{x}_j \in \mathcal{V}$. Edges $e_{ij} \in \mathcal{E}$ connect a factor node to a variable node. The adjacency set $\mathcal{A}(\phi_i)$ contains all variables adjacent to the factor ϕ_i and we denote an assignment to it by $\mathbf{x}_i$. Given an arbitrary global function $\phi(\mathbf{x})$, $\mathcal{F}$ defines its factorization as

$$\phi(\mathcal{V}) = \prod_k \phi_k(\mathbf{x}_k), \quad (\text{A.73})$$

where each factor $\phi_k(\mathbf{x}_k)$ denotes the potential function of the variables in $\mathcal{A}(\phi_k)$.

If we transform the toy example of landmark-based localization¹⁰ into a factor graph, as shown in Figure A.5b, we can represent the posterior $p(\mathcal{X}|\mathcal{Z})$ as

$$\begin{aligned} \phi(\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{l}_1, \mathbf{l}_2) &\propto \phi_1(\mathbf{x}_1) \phi_2(\mathbf{x}_2, \mathbf{x}_1) \phi_3(\mathbf{x}_3, \mathbf{x}_2) \\ &\quad \times \phi_4(\mathbf{x}_1) \\ &\quad \times \phi_5(\mathbf{l}_1) \phi_6(\mathbf{l}_2) \\ &\quad \times \phi_7(\mathbf{x}_2, \mathbf{l}_1) \phi_8(\mathbf{x}_3, \mathbf{l}_1) \phi_9(\mathbf{x}_3, \mathbf{l}_2), \end{aligned} \quad (\text{A.74})$$

where we see that the function $\phi(\cdot)$ denotes the factor connecting two state variables, implicitly encoding their causal relationship.

Therefore, the direction of marginalization can be redefined as new factors (e.g., new measurements) are introduced, as I will demonstrate in the next section. Once the factorization function of to-be-estimated states $\phi(\mathbf{x})$ is determined, e.g., as proper probabilistic density functions of measurements given state variables $\phi_8(\mathbf{x}_3, \mathbf{l}_1) \propto p(\mathbf{z}_3|\mathbf{x}_3, \mathbf{l}_1)$, the factor graph can be transformed into a Bayesian network (aka elimination). This transformation enables us to infer the belief of all variables in a batch. Simply put, a factor graph can be viewed as a meta-graph of the Bayesian network.

A.2.7 Factor Graph Optimization

Following the previous section, we now use the factor graph framework to solve state estimation problems. Given (A.74) and recalling our introduction of the MAP inference in Section A.2.2.5, we can leverage the MAP estimation to perform inference on the factor graph as

$$\mathcal{X}^{\text{MAP}} = \underset{\mathcal{X}}{\operatorname{argmax}} \phi(\mathbf{x}) \quad (\text{A.75})$$

$$= \underset{\mathcal{X}}{\operatorname{argmax}} \prod_k \phi_k(\mathbf{x}_k), \quad (\text{A.76})$$

¹⁰I assume that, in this example, the positions of the landmarks are unknown and must also be estimated alongside the robot's state variables.

where the factors $\phi_j(\cdot)$ encode the prior information or measurement of the associated to-be-estimated states. I refer to the optimization problem in (A.75) as *factor graph optimization*. By assuming that all factors follow the Gaussian distribution

$$\phi_k(\mathbf{x}_k, \mathbf{l}_m) \propto \exp \left\{ -\frac{1}{2} \|\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k, \mathbf{l}_m)\|_{\Sigma_k}^2 \right\}, \quad (\text{A.77})$$

the factor graph optimization is equivalent to the nonlinear least-square estimation introduced in Section A.2.3, where we can employ the iterative nonlinear solvers (e.g., Gauss-Newton) to solve the optimization problem.

Since we are performing a batch optimization involving large-dimensional state variables using factor graphs, it is essential to examine the internal process of factor graph optimization, particularly how the Jacobian matrix is constructed and, more importantly, how its inverse is computed as additional factors are introduced into the graph.

Recalling the direct methods for solving least-squares optimization, we generally encounter the following *normal equation*

$$\underbrace{(\mathbf{A}^T \mathbf{A})}_{\text{Hessian}} \boldsymbol{\delta}^* = \mathbf{A}^T \mathbf{b} \quad (\text{A.78})$$

as described in (A.38) and (A.46), where the Jacobian matrix $\mathbf{A}$ is computed from m factors and k state variables ($\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{b} \in \mathbb{R}^{m \times 1}$). Solving these equations requires re-factorization of the Hessian matrix, commonly referred to as the *information matrix*

$$\boldsymbol{\Lambda} \triangleq \mathbf{A}^T \mathbf{A} \quad (\text{A.79})$$

in the context of state estimation. Common algorithms for re-factorization include the *Cholesky factorization* and the *QR factorization*, to which the reader can find a detailed introduction in Golub and Loan [204, Chapter 4 and 5].

Cholesky factorization

Briefly, the Cholesky factorization decomposes the (A.79) into the product of an upper-triangular (also under-triangular) matrix $\mathbf{R}$ (aka Cholesky triangle) and its transpose as $\boldsymbol{\Lambda} = \mathbf{R}^T \mathbf{R}$. This transforms the normal equations into a two-step process

$$\mathbf{R}^T \mathbf{y} = \mathbf{A}^T \mathbf{b}, \quad (\text{A.80})$$

$$\mathbf{R} \boldsymbol{\delta}^* = \mathbf{y}. \quad (\text{A.81})$$

Using the equations (A.80) and (A.81), the perturbation state $\boldsymbol{\delta}^*$ can be efficiently computed through forward and back-substitution. Similarly, the QR factorization algorithm involves transforming the system into an upper-triangular form, followed by back-substitution to calculate the solution.

QR factorization

The QR factorization, on the contrary, aims to re-structure the Jacobian matrix $\mathbf{A}$ and the right-hand-side of (A.78) as

$$\mathbf{A} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{Q}^T \mathbf{b} = \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix},$$

where the matrix $\mathbf{Q}$ is an $m \times m$ orthogonal matrix. The matrix $\mathbf{R}$ is the same triangular matrix as in (A.80). The vectors $\mathbf{d}$ and $\mathbf{e}$ are formulated with dimension k and $m - k$, respectively. By introducing the *Householder reflection matrix* $\mathbf{H}_n$, such that

$$\mathbf{H}_n \dots \mathbf{H}_2 \mathbf{H}_1 \mathbf{A} = \mathbf{Q}^T \mathbf{A} = \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix},$$

we can reformulate the update step in (A.78) as

$$\|\mathbf{A}\delta - \mathbf{b}\|_2^2 = \|\mathbf{Q}^T \mathbf{A}\delta - \mathbf{Q}^T \mathbf{b}\|_2^2 = \|\mathbf{R}\delta - \mathbf{d}\|_2^2 + \|\mathbf{e}\|_2^2. \quad (\text{A.82})$$

By assuming the perturbation state is optimal, the least-squares sum $\|\mathbf{e}\|_2^2$ vanishes. Thus, we can compute the optimal perturbation state

$$\mathbf{R}\delta^* = \mathbf{d} \quad (\text{A.83})$$

via back-substitution, as comprehensively explained in Dellaert and Kaess [153, Chapter 2.4].

The most vital property for ensuring an efficient factorization of the Jacobi matrix $\mathbf{A}$ (i.e., the information matrix $\mathbf{\Lambda}$) is *sparse*. Sparsity is a characteristic feature of state estimation approaches, as states are typically influenced by only a limited number of measurements or constraints, resulting in a sparse structure in the associated matrices.

For example, if we depict the toy example in Figure A.5b, the linear system of (A.37) can be visualized as in Figure A.6a. Due to the sparsity of the Jacobian matrix $\mathbf{A}$, its Hessian (information) matrix $\mathbf{\Lambda}$ also retains a sparse structure, as illustrated in Figure A.6b. Interestingly, the Hessian matrix reflects the adjacency structure of the Markov random field (MRF) when the MRF in Figure A.5c is represented in matrix form. This demonstrates the intrinsic relationship between factor graphs and MRFs.

From an inference perspective, re-factorizing the Hessian matrix, for example, using the Cholesky algorithm, is analogous to the elimination process that transforms the factor graph into a Bayesian network. This elimination process reorganizes the sparse information matrix into a triangular form, facilitating both forward and back substitution to solve the linear system. As Dellaert and Kaess [153, Chapter 3 and 4] explains, different elimination heuristics can lead to varying factorization results by altering the order in which matrices are triangularized.

$$[A|b] = \begin{array}{c} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \end{array} \begin{array}{c} \delta x_1 \quad \delta x_2 \quad \delta x_3 \quad \delta l_1 \quad \delta l_2 \quad b \\ \left[\begin{array}{cccccc} \mathbf{A}_{11} & & & & & b_1 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & & & & b_2 \\ & \mathbf{A}_{32} & \mathbf{A}_{33} & & & b_3 \\ \mathbf{A}_{41} & \mathbf{A}_{65} & & & & b_4 \\ & & & \mathbf{A}_{54} & & b_5 \\ & & & & & b_6 \\ & \mathbf{A}_{72} & & \mathbf{A}_{74} & & b_7 \\ & & \mathbf{A}_{83} & \mathbf{A}_{84} & & b_8 \\ & & \mathbf{A}_{93} & & \mathbf{A}_{95} & b_9 \end{array} \right] \end{array}$$

(a) Sparse Jacobian matrix A .

$$\Gamma \triangleq A^T A = \begin{bmatrix} \Gamma_{11} & & & & \Gamma_{14} \\ & \Gamma_{22} & & \Gamma_{24} & \Gamma_{25} \\ & & \Gamma_{33} & \Gamma_{34} & \Gamma_{35} \\ \Gamma_{41} & \Gamma_{42} & \Gamma_{43} & \Gamma_{44} & \\ & \Gamma_{52} & \Gamma_{53} & & \Gamma_{55} \end{bmatrix}$$

(b) Hessian matrix Γ of the example in A.6a.

Figure A.6: Jacobian and Hessian matrices of the toy example in Figure A.5b.

These ordering heuristics reflect the inherent flexibility of inference within factor graph optimization, allowing for the development of novel heuristics that define marginalization dependencies. Consequently, understanding the elimination process featured in a *variable elimination algorithm* is crucial for advancing factor graph optimization techniques.

The variable elimination algorithm [153, Algorithm 3.1 and 3.2] converts a factor graph to a Bayesian network by defining a joint probability density over the hidden states as

$$p(\mathcal{X}) = \prod_k p(\mathbf{x}_k | \mathbf{S}_k), \quad (\text{A.84})$$

where $\mathbf{S}_k \subseteq \mathcal{X}_i \setminus \{\mathbf{x}_k\}$ represents the separator of $\mathbf{x}_k$, denoting the variables on which $\mathbf{x}_k$ is conditioned after elimination. This conditioning is explicitly encoded in the directed structure of the resulting Bayesian network.

The elimination algorithm removes one variable from the factor graph at a time. To eliminate the variable $\mathbf{x}_k$, all factors $\phi_i(\mathcal{X}_i)$ connected to $\mathbf{x}_k$ are first removed. The separator is then constructed as $\mathbf{S}_k = \cup_i \mathcal{X}_i \setminus \{\mathbf{x}_k\}$. A new product factor is created as

$$\psi(\mathbf{x}_k, \mathbf{S}_k) = \prod_i \phi_i(\mathcal{X}_i), \quad (\text{A.85})$$

which is then factorized into $p(\mathbf{x}_k | \mathbf{S}_k) \tau(\mathbf{S}_k)$. Here, the conditional probability density $p(\mathbf{x}_k | \mathbf{S}_k)$ is added as a factor to the Bayesian network, while the new factor $\tau(\mathbf{S}_k)$ is reintroduced into the partially converted factor graph.

The algorithm continues until the last variable, $\mathbf{x}_K$ is eliminated. At this final step, all remaining factors take the form $\phi_i(\mathbf{x}_K)$. Consequently, the separator $\mathbf{S}_K$ becomes an empty set, and the final factor added to the Bayesian network is the prior $p(\mathbf{x}_K)$ over $\mathbf{x}_K$. For a better illustration of the elimination algorithm, I refer the reader to Dellaert and Kaess [153, Chapter 3.3.4] for more details.

The Bayesian network in (A.84) facilitates straightforward MAP inference and marginalization, while the separators in a sparse factor graph are typically small [153]. Under the assumption of Gaussian noise, the elimination process corresponds to a sparse QR factorization of the whitened Jacobian matrix $\mathbf{A}$.

A.2.8 Incremental Smoothing and Mapping

Thanks to the flexibility of the elimination process using factor graphs, numerous efficient algorithms have been developed over the past decade to address factor graph optimization problems. Among these, methods leveraging *incremental smoothing and mapping* (iSAM), originally introduced by Kaess et al. [217], are widely employed in practical applications.

In a dynamic estimation setting, the robot incorporates a new measurement $\mathbf{z}_{K+1}$ into the set of observations $\mathbf{Z}$. Naive nonlinear optimization approaches solve a batch problem over all states $\mathbf{X}$ to determine a new linearization point. However, due to the locality property, only a small subset $\mathbf{X}_{K+1} \subseteq \mathbf{X}$ is influenced by the new observation. The iSAM algorithms exploit this property by relinearizing only the subset of variables that are affected.

The iSAM1 algorithm [217] updates the triangular matrix $\mathbf{R}$ in the factorization techniques using *Givens rotations*, which affect only a small portion of the matrix. As the observation set $\mathbf{Z}$ grows, the quality of the variable ordering may degrade, requiring periodic batch reordering and linearization steps to maintain performance.

The successor to iSAM1 is iSAM2, described as “[...] a novel incremental exact inference method that allows for incremental reordering and just-in-time relinearization.” [161, p. 2]. This algorithm extends iSAM by leveraging a new data structure, the Bayes tree [218], which eliminates the need for batch steps altogether. Unlike its predecessor, iSAM2 operates directly on graphical models without converting the factor graph into a sparse matrix, thus streamlining the inference process.

Bayes Tree

The Bayes tree is a directed graph where each node represents a clique $\mathcal{C}_k$ from the chordal Bayesian network. A clique is a subset of densely connected variables, where every pair is connected by a single edge.

The Bayes tree factorizes the joint probability density as

$$p(\Theta) = \prod_k p(\mathbf{F}_k | \mathbf{S}_k), \quad (\text{A.86})$$

where the separator $\mathbf{S}_k$ is the intersection $\mathcal{C}_k \cap \diamond_k$ of the clique $\mathcal{C}_k$ with its parent clique $\diamond$. The frontal variables $\mathbf{F}_k$ are the variables in the clique that are not part of the parent clique, i.e., $\mathcal{C}_k \setminus \mathbf{S}_k$.

For the root nodes, the separator $\mathbf{S}_k$ is empty, and the corresponding factor simplifies to a prior over the root variables. The directed edges in the Bayes tree encode the same conditioning relationships as those in the Bayes net. Since the Bayes net resulting from variable elimination is chordal, it can be transformed into a Bayes tree by identifying all its cliques in reverse elimination order [218].

Incremental inference in factor graphs corresponds to updating the Bayes tree, which stores the Jacobian matrix $\mathbf{A}$. When a new set of observations $\mathbf{Z}_{K+1}$ is received, a new factor $\phi_{K+1}(\mathbf{X}_{K+1})$ is constructed. For all $\mathbf{x} \in \mathbf{X}_{K+1}$, only the paths between the cliques containing $\mathbf{x}$ and the root are affected. These affected nodes are detached from the unaffected subtrees and converted into a factor graph. After adding the new factor $\phi_{K+1}(\mathbf{X}_{K+1})$ to the factor graph, variable elimination and a subsequent clique search yield a new Bayes tree. The unaffected subtrees are then reattached to the updated Bayes tree.

The efficiency of incremental inference heavily depends on the variable ordering during elimination. An optimal ordering minimizes “fill-in,” i.e., the number of additional non-zero entries in $\mathbf{A}$ during elimination, thereby keeping clique sizes small. Finding an optimal ordering is NP-hard, so heuristics like the column approximate minimum degree (COLAMD) algorithm [219] are commonly used. In an incremental setting, COLAMD provides a reasonable variable ordering for the current step. New observations often connect to recently accessed variables, such as a between-state factor that depends on the robot’s current and previous poses. These recently affected variables, denoted $\mathbf{X}_{K+1}$, are more likely to be involved in subsequent

updates. If the variables in $\mathbf{X}_{K+1}$ are near the root of the Bayes tree, the affected subtree remains small, allowing for efficient updates.

Motivated by this, Kaess et al. [161] propose a variable ordering strategy that integrates COLAMD while constraining $\mathbf{X}_{K+1}$ to appear at the end of the ordering. This approach helps maintain small affected subtrees and keeps the fill-in at manageable levels, resulting in efficient updates during incremental inference.

Fluid Relinearization

Another critical component of iSAM2 is the *fluid re-linearization*. Performing incremental inference often results in changes to the state estimate. For each variable $x_k \in \mathbf{X}$, fluid relinearization tracks the current perturbation state δ_k and identifies variables where the norm of the delta exceeds a predefined threshold β . Once these variables are marked, their relinearization points are updated, and all affected cliques, along with their parent cliques, are also marked for relinearization.

During the Bayes tree update, the top portion of the tree, which includes all marked cliques and their parent cliques up to the root, is severed and fluidly relinearized. After relinearizing this portion, a new Bayes tree is constructed. The orphaned subtrees, which are unaffected by the changes, are then reattached to the updated Bayes tree in their original positions.

This fluid relinearization process ensures that the inference remains efficient and accurate while minimizing unnecessary re-computation.

Partial State Updates

After performing matrix factorization with fluid relinearization, the state update is executed through full back-substitution. While updates primarily affect the top of the Bayes tree, changes in variable estimates can propagate to the subtrees. However, these changes do not always significantly impact the subtrees.

The process begins by solving for all variables at the top of the Bayes tree to compute the corresponding delta values. Each subtree is then incrementally processed until a clique is encountered where all referenced variables exhibit delta changes smaller than a specified threshold α . At this point, further processing of that subtree is halted, as no meaningful updates are required.

A.2.9 Kalman Filter versus Factor Graph Optimization

We have previously discussed the theoretical differences between filtering-based and smoothing-based methods in Section A.2.5. In this section, we focus on gaining a practical understanding of the state estimation methods introduced earlier, particularly the Kalman filter and factor graph optimization.

Briefly, the main difference between Kalman filters and factor graph optimization lies in the reliance on the *first-order Markov process*, which entails a complete marginalization of the state's belief at each step. Consequently, Kalman filters do not maintain a structured model that retains all related information, such as past measurements, over time. The factor graph optimization, in contrast, instantiates the nonlinear least-square estimation by converting the factor graph into a Bayesian network via variable elimination, where all to-be-estimated state variables in the batch are corrected on new measurements in each optimization epoch.

From another perspective, all factors on the graph “force” the system’s trajectory to adhere to the relationships and dependencies encoded by the measurements and prior information. As such, they also act as *constraints* on the to-be-estimated states. Over time, redundant constraints from various sources (e.g., sensors) can be incorporated into the graph, enhancing the robustness of the estimation process. It becomes evident that this approach enables superior performance and smoother estimated trajectories compared to filtering-based methods.

In Figure A.7, I illustrate the difference between the Kalman filter and factor graph optimization in state estimation by interpreting a one-dimensional mass-spring system, initially introduced in Barfoot [1, pp. 52].

At each step, the mass-spring system is extended by adding a new mass connected by either a spring or a fixed joint. In Kalman filters, the state of the mass (e.g., its position) becomes fixed after the measurement update due to the complete marginalization process.

In contrast, factor graph optimization maintains multiple factors on the graph over time, which can be interpreted as multiple strings connecting the masses. These strings act as constraints, pulling the mass at each step into the best-fit position based on different “forces”. Because these forces (i.e., constraints) are subject to uncertainty, they can be updated during each batch optimization when a new contradicting factor is introduced, such as a loop-closure constraint in SLAM (see Section A.3.6). Consequently, the states of all masses across the timeline are refined, resulting in a globally consistent estimate.

Although both filtering and optimization-based methods share an equivalent mathematical foundation, factor graph optimization offers several distinct advantages that are essential for our research on handling outliers in this thesis, which can be summarized as follows

- **flexible modeling:** Allows the seamless combination of multiple and diverse constraints without requiring complex engineering or significant adaptation (see Chapter 3). For example, adding more strings in Figure A.7.

- **flexible inference:** Enables adaptable marginalization across different time horizons (e.g., in a sliding window) and customizable elimination orderings (see Chapter 3). For example, replacing strings with fixed joints in Figure A.7.
- **flexible extension:** Facilitates the incorporation of advanced modeling techniques, such as robust error modeling (see Chapter 4) and covariance estimation (see Chapter 5). For example, changing the string’s tension in Figure A.7.

However, I am **not claiming** that “**factor graph optimization must always outperform Kalman filters**”, as discussed in Section 3.5.5. Meanwhile, several advanced filtering techniques, such as invariant Kalman filters [155, 214], iterated Kalman filters [154], and equivariant filters [156], have been proposed in recent years. These methods substantially relax the strong assumptions underlying the classical Kalman filter family and, as a result, improve performance and robustness. Although I have not evaluated other advanced filtering methods in this thesis, I believe they would likely demonstrate improved performance compared to the UKF. However, these approaches do not provide the same modeling flexibility as factor graph optimization and therefore fall outside the scope of the claims made in this thesis.

In short, the primary reason for employing factor graph optimization lies in its superior capacity for modeling and extensibility, which provides a sustainable and adaptable research platform.

(a) Kalman filtering operates analogously to a mass-spring system, where state constraints act as fixed joints connecting past states, inputs, and measurements. Due to the Markov property, the next state estimate at t_3 depends solely on the state at t_2 .

(b) Factor graph optimization operates analogously to a mass-spring system, where all state constraints (factors) are represented as springs. During batch optimization at each timestamp, the ‘energy’ (or information) stored in the springs is adjusted to achieve a mechanical equilibrium, corresponding to the minimum energy state. The spring constants can also be adapted dynamically using techniques such as advanced system identification.

(c) Factor graph optimization operates analogously to a mass-spring system with a sliding window (fixed lag). All states, inputs, and measurements falling into the sliding window are marginalized out.

Figure A.7: Comparison of state estimation techniques to a 1-D mass-spring system, where an estimate (e.g., measurement update) for the current state at t_2 is available. The grey areas denote complete marginalization, where all past information falling within these areas is marginalized. Figures are adapted from Barfoot [1, Figure 3.1]

A.3 Primer on Vehicle Localization

Building on the previous section, I now present a practical perspective on state estimation, tailored explicitly for vehicle localization. This section begins with an overview of coordinate systems and the mathematical foundations of Lie groups, aiming to establish the necessary technical background for the sensors and fusion mechanisms discussed in the subsequent chapters.

A.3.1 Frames and Frame Transformation

Vehicle localization in outdoor, large-scale environments often involves expressing the vehicle's state across multiple coordinate systems, each tied to distinct frames of reference, such as global, local, and body-centric frames. In this section, I introduce the relevant frames shown in Figure A.8 and the transformations between them as they pertain to this thesis.

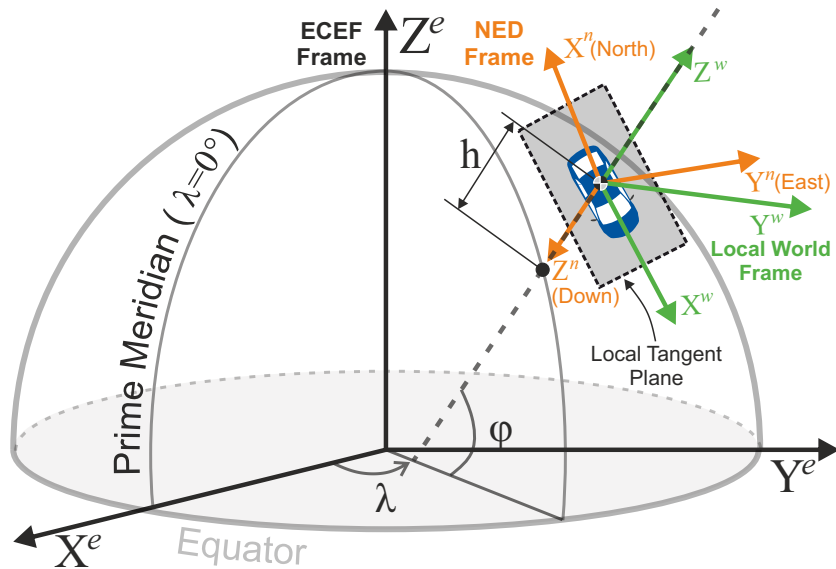


Figure A.8: Coordinate frames used in this thesis.

A.3.1.1 Global Frame

Starting with GPS in 1987, GNSS typically provides a vehicle's position in the World Geodetic System (WGS84) frame using *geodetic coordinates* (Latitude φ , Longitude λ , and Height h , LLH) [220], as shown in Figure A.8. Geodetic coordinates can be transformed into the *Earth-centered, Earth-fixed (ECEF)* frame that is defined at the center of Earth's mass, denoted as $(\cdot)^e$. Unlike geodetic coordinates, ECEF coordinates are expressed in meters in

a fixed frame that rotates with the Earth, providing a consistent, global reference frame for positioning that is independent of local geodetic variations.

A geodetic coordinate $\mathbf{p}^{\text{LLH}} = [\varphi \ \lambda \ h]^T$ can be transformed to the ECEF frame using [107]

$$x_b^e = (R_e + h) \cos \varphi \cos \lambda, \quad (\text{A.87})$$

$$y_b^e = (R_e + h) \cos \varphi \sin \lambda, \quad (\text{A.88})$$

$$z_b^e = [R_e(1 - \text{ecc}^2) + h] \sin \varphi, \quad (\text{A.89})$$

where the constant $\text{ecc} = 0.08181919$ is the eccentricity⁴ of the ellipsoid. The transverse radius of curvature given the latitude φ is calculated as $R_e(\varphi) = \frac{a}{\sqrt{1 - \text{ecc}^2 \sin^2(\varphi)}}$, where the scalar $a = 6\,378\,137$ m denotes the equatorial radius⁴ of the Earth.

However, transforming a Cartesian coordinate in frame e back to LLH using the inverse of (A.87)-(A.89) can only be solved iteratively due to the nonlinearity [107]. Although many closed-form alternatives for this transformation are available, I use Heikkinen's solution [221] in this work, as it offers high precision [222]. Due to limited space, I refer the reader to [221] or our code² for more details on the implementation.

A.3.1.2 Navigation Frame

As many estimation and control approaches require a Cartesian frame in a local tangent plane, the *North-East-Down (NED) frame* and the *East-North-Up (ENU) frame* are commonly introduced as navigation frames $(\cdot)^n$ to present the vehicle's velocity and orientation. Since the navigation frame establishes a locally tangential reference based on the current global position, with its axes always oriented toward fixed directions (e.g., the North-axis consistently pointing toward the local North, as illustrated in Figure A.8), each geodetic coordinate can define a unique navigation frame. Given a point in the e frame $\mathbf{p}_0^e$ as origin, a fixed navigation frame (aka local tangent plane) can be determined by two plane rotations associated with longitude λ_0 and latitude φ_0 of $\mathbf{p}_0^{\text{LLH}}$. Figure A.8 illustrates this transformation from the frame e to the NED frame using the direction cosine matrix (DCM)

$$\mathbf{R}_e^{\text{ned}}(\mathbf{p}_0^{\text{LLH}}) = \begin{bmatrix} -\sin \varphi_0 \cos \lambda_0 & -\sin \varphi_0 \sin \lambda_0 & \cos \varphi_0 \\ -\sin \lambda_0 & \cos \lambda_0 & 0 \\ -\cos \varphi_0 \cos \varphi_0 & -\cos \varphi_0 \sin \lambda_0 & -\sin \varphi_0 \end{bmatrix}. \quad (\text{A.90})$$

The DCM from frame e to the ENU frame is given as

$$\mathbf{R}_e^{\text{enu}}(\mathbf{p}_0^{\text{LLH}}) = \begin{bmatrix} -\sin \lambda_0 & \cos \lambda_0 & 0 \\ -\cos \lambda_0 \sin \varphi_0 & -\sin \lambda_0 \sin \varphi_0 & \cos \varphi_0 \\ \cos \lambda_0 \cos \varphi_0 & \sin \lambda_0 \cos \varphi_0 & \sin \varphi_0 \end{bmatrix}. \quad (\text{A.91})$$

⁴The eccentricity and the Earth's equatorial radius in different satellite systems vary slightly depending on the ellipsoid and the satellite geometry. In this work, we follow the parameter definitions from WGS84 [223].

Standard control and global navigation approaches often use the NED frame as their navigation frame due to the natural alignment of the z -axis with gravity. In contrast, the ENU frame is typically employed in localization approaches using local sensors (e.g., LiDAR), as it provides a convenient reference system for measured targets, such as point clouds.

A.3.1.3 Local-World Frame

As a vehicle may begin moving in a direction that does not align with the navigation frame due to its initial orientation, localization approaches utilizing local sensors (e.g., LiDARs [120] and cameras [123]) often introduce a *local-world frame* $(\cdot)^w$. This frame is concentric with the ENU frame at the vehicle's initial state and serves as a local reference system for interpreting sensor observations. Thus, a global pose $\mathbf{T}_b^e$ can be transformed into the local-world frame given the initial pose (aka anchor pose) $\mathbf{T}_{l,anc}^e = \begin{bmatrix} \mathbf{R}_{anc}^e & \mathbf{p}_{l,anc}^e \\ \mathbf{0} & 1 \end{bmatrix}$ using

$$\mathbf{T}_k^w = (\mathbf{T}_{anc}^e)^{-1} \mathbf{T}_k^e. \quad (\text{A.92})$$

A.3.1.4 Notation

To present the state variables in different frames, I use $\mathbf{R}_b^e \in \mathbb{R}^{3 \times 3}$ and $\mathbf{p}_b^e \in \mathbb{R}^3$ to denote the rotation matrix and position vector of frame b relative to frame e . This notation is extended to $\mathbf{R}_{b,t}^e$ to represent the states with respect to time t . For motion increments in the same frame, I simplify the notation as $\Delta \mathbf{p}_{ij}$ to represent the translational offset of two timestamps i and j . I follow the pose representation $\mathbf{T}_b^e = \begin{bmatrix} \mathbf{R}_b^e & \mathbf{p}_b^e \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$ to calculate the motion increment [1]. For high-dimensional transition matrices in GP motion models (e.g., $\Lambda(\tau) \in \mathbb{R}^{18 \times 18}$), I denote the subblocks $\Lambda_{mn} \in \mathbb{R}^{6 \times 6}$ associated with different state components for linear state querying in (3.19).

In this work, I present the pose and velocity of the vehicle in the ECEF frame. A transformation from frame e to frame n is used to formulate factors and calculate error metrics. The body frame is denoted as $(\cdot)^b$.

A.3.2 State Space for Vehicle Localization

We define the state variables used in the vehicle localization applications in this thesis at timestamp t as

$$\mathbf{x}_t \triangleq \{\mathbf{T}_{b,t}^e \ \mathbf{v}_{b,t}^e \ \mathbf{b}_{b,t}^a \ \mathbf{b}_{b,t}^g \ \mathbf{c}_t^r\}. \quad (\text{A.93})$$

We estimate the vehicle pose $T_{b,t}^e \in SE(3)$ and 6-D velocity v_b^e in frame e . The vectors $b_{b,t}^a$ and $b_{b,t}^g$ denote the 3-D biases of the accelerometer and gyroscope, respectively. The 2-D vector $c_t^r = [c_{b,t} \ c_{d,t}]^T$ represents the GNSS receiver clock bias $c_{b,t}$ and drift $c_{d,t}$, which is only estimated by the tightly coupled fusion of raw GNSS observations.

A.3.3 Lie Group and Lie Algebra

Throughout numerous localization applications in 3-D space, it is standard practice to represent the vehicle's pose and velocities in Euclidean (vector) space. However, unlike translations, rotations are derived from a distinct mathematical structure known as the *noncommutative group*, which does not possess all the properties of Euclidean space, such as commutativity under composition [1, Chapter 8]. As a result, handling state estimation in vector space often necessitates additional techniques to address issues like singularities, constraints, and inaccurate nonlinear uncertainty propagation, as discussed in Barfoot and Furgale [149] and Long et al. [224]. To overcome these challenges, we require a more suitable, perhaps more general, state parameterization that facilitates a natural approach to nonlinear state and uncertainty propagation. This leads us to the *Lie theory* including the *Lie Group* G and its associated *Lie algebra* $\mathfrak{g}$.

The Lie group G with its algebra $\mathfrak{g}$, a concept introduced by the mathematician Sophus Lie in the 19th century, provides a mathematical framework to study continuous symmetries and transformations [225]. It is a differential manifold and also a group providing a set of elements with operations (e.g., multiplication) that satisfy the group axioms: closure, associativity, identity, and inverse [1, Chapter 8].

The Lie algebra $\mathfrak{g}$ provides a local, linearized vector space of the Lie group G in its tangent space at the identity element ε , so that we have $\mathfrak{g} \triangleq T_\varepsilon G$. We use the *exponential map*

$$\text{Exp} : \mathfrak{g} \rightarrow G$$

to transform the elements in the tangent space back to the Lie group, its inverse is called the *logarithmic map*

$$\text{Log} : G \rightarrow \mathfrak{g},$$

which projects elements of the Lie group onto its tangent (vector) space. Figure A.9 illustrates these mappings, which form the basis for transitioning between the non-linear manifold structure of the Lie group and its locally linear tangent space.

Common Lie groups concerned with robotics in 3-D space include the *special orthogonal group* $SO(3)$ and its corresponding Lie algebra $\mathfrak{so}(3)$, which represent rotations, and the *special Euclidean group* $SE(3)$ and its Lie algebra $\mathfrak{se}(3)$, which represent rigid body transformations.

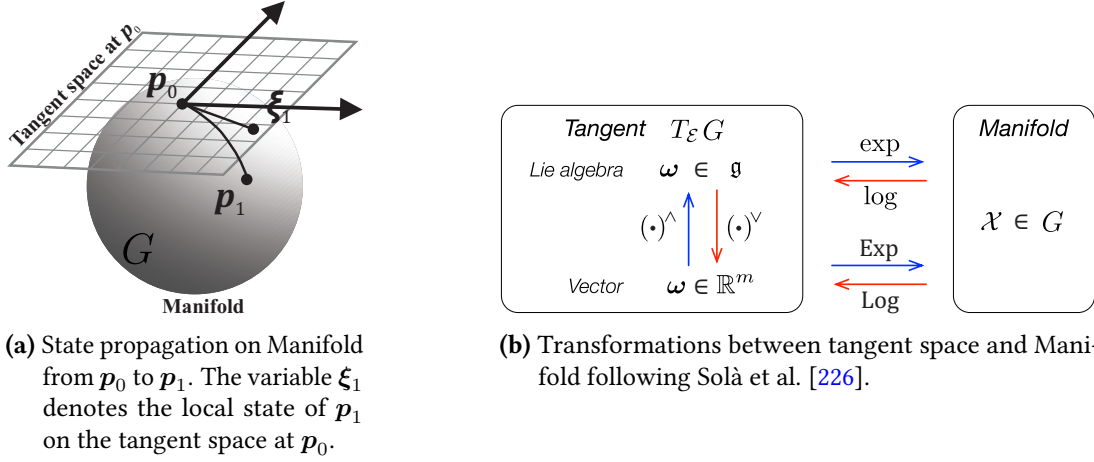


Figure A.9: Manifold, tangent space, and their transformations.

These Lie groups also belong to an important family in Lie theory called *matrix Lie group*, where the elements are invertible matrices, such as the transformation matrix $\mathbf{T} \in \mathbb{R}^{4 \times 4}$.

The rotation group $SO(3)$, represented by the rotation matrix $\mathbf{R} \in SO(3)$, satisfies the orthogonality condition $\mathbf{R}^T \mathbf{R} = \mathbf{I}$, where the matrix $\mathbf{I}$ is the identity matrix. Taking the derivative of this condition with respect to time, we obtain $\mathbf{R}^T \dot{\mathbf{R}} + \dot{\mathbf{R}}^T \mathbf{R} = 0$. This shows that the term $\mathbf{R}^T \dot{\mathbf{R}}$ is skew-symmetric, which can be represented in $SO(3)$ using its Lie algebra, defined as

$$[\boldsymbol{\omega}]_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \quad (\text{A.94})$$

Later, we use two operations $\text{hat}(\cdot)^\wedge$ and $\text{vee}(\cdot)^\vee$ to transform group elements from $\mathbb{R}^3$ to $\mathfrak{se}(3)$ and vice-versa, which are expressed as

$$\text{hat} : \mathbb{R}^3 \rightarrow \mathfrak{so}(3) \quad \boldsymbol{\omega} \mapsto \boldsymbol{\omega}^\wedge = [\boldsymbol{\omega}]_{\times} \quad (\text{A.95})$$

$$\text{vee} : \mathfrak{so}(3) \rightarrow \mathbb{R}^3 \quad [\boldsymbol{\omega}]_{\times} \mapsto [\boldsymbol{\omega}]_{\times}^\vee = \boldsymbol{\omega}. \quad (\text{A.96})$$

Given this property, the derivative of the rotation matrix can be expressed as $\dot{\mathbf{R}} = \mathbf{R} \boldsymbol{\omega}^\wedge \in T_{\mathbf{R}} SO(3)$. If the rotation matrix is aligned with the origin of the tangent space $T_{\epsilon} G$ (aka $\mathbf{R}_0 = \mathbf{I}$), we can represent the reference-less rotation rate as $\dot{\mathbf{R}} = \boldsymbol{\omega}^\wedge$. Interestingly, this skew-symmetric matrix $\boldsymbol{\omega}^\wedge$ represents the instantaneous angular velocity in the tangent space, providing a local, linearized representation of the rotation. Thus, we can derive the exponential map of $SO(3)$ as

$$\mathbf{R}(t) = \exp(\boldsymbol{\omega}^\wedge t) \quad (\text{A.97})$$

$$= \exp(\boldsymbol{\theta}^\wedge), \quad (\text{A.98})$$

where the vector $\boldsymbol{\theta}$ denotes the integrated rotation in angle-axis form, $\boldsymbol{\theta} \triangleq \boldsymbol{\omega} t \in \mathbb{R}^3$. In practice, the exponential and logarithmic map of the rotation group can be calculated in closed form using *Rodrigues' formula* [227, pp. 28] as

$$\mathbf{R} = \mathbf{I} + \frac{\sin \theta}{\theta} \boldsymbol{\omega}^\wedge + \frac{1 - \cos \theta}{\theta^2} (\boldsymbol{\omega}^\wedge)^2, \quad (\text{A.99})$$

$$\text{Log}(\mathbf{R}) = \frac{\theta}{2 \sin \theta} (\mathbf{R} - \mathbf{R}^T), \quad (\text{A.100})$$

where the term $(\boldsymbol{\omega}^\wedge)^2$ is computed as $(\boldsymbol{\omega}^\wedge)^2 = \boldsymbol{\omega} \boldsymbol{\omega}^T - \mathbf{I}$.

Analogically, the exponential and logarithmic map of transformation matrix $\mathbf{T} \in SE(3)$ is computed with the local coordinate $\boldsymbol{\xi} = [\mathbf{v} \ \boldsymbol{\omega}]^T$ as

$$\mathbf{T} = \exp \boldsymbol{\xi}^\wedge = \exp \left(\begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}^\wedge t \right) \quad (\text{A.101})$$

$$= \begin{bmatrix} \exp \boldsymbol{\omega}^\wedge & \mathbf{V} t \\ \mathbf{0} & 1 \end{bmatrix} \quad (\text{A.102})$$

and

$$\text{Log}(\mathbf{T}) = \boldsymbol{\xi}^\wedge \quad (\text{A.103})$$

$$= \begin{bmatrix} \boldsymbol{\theta} \boldsymbol{\omega}^\wedge & \mathbf{V}^{-1} t \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (\text{A.104})$$

where the matrix $\mathbf{V}$ and its inverse are computed using Rodrigues' formula as

$$\mathbf{V} = \mathbf{I} + \frac{1 - \cos \theta}{\theta^2} \boldsymbol{\omega}^\wedge + \frac{\theta - \sin \theta}{\theta^3} (\boldsymbol{\omega}^\wedge)^2, \quad (\text{A.105})$$

$$\mathbf{V}^{-1} = \mathbf{I} - \frac{1}{2} \log \mathbf{R} + \frac{1}{\theta^2} \left(1 - \frac{\theta \sin \theta}{2(1 - \cos \theta)} \right) (\log \mathbf{R})^2. \quad (\text{A.106})$$

As shown in Figure A.9b, we can calculate the exponential and logarithmic maps using elements from different spaces, which gives us

$$\mathbf{T} = \text{Exp}(\boldsymbol{\xi}) \triangleq \exp(\boldsymbol{\xi}^\wedge), \quad (\text{A.107})$$

$$\boldsymbol{\xi} = \text{Log}(\mathbf{T}) \triangleq \log(\mathbf{T})^\vee. \quad (\text{A.108})$$

Now, we can summarize the state representation using Lie theory from the equations (A.99)-(A.104) and analyze why this parameterization is superior to vector-space representations, such as Euler angles. If we consider a state update (perturbation) $\boldsymbol{\xi}$ to the current transformation $\mathbf{T}_1$, the new transformation can be calculated as

$$\mathbf{T}_2 = \exp(\boldsymbol{\xi}^\wedge) \mathbf{T}_1. \quad (\text{A.109})$$

Clearly, the exponential map provides a natural way to handle non-linear transformations, mapping local perturbations in the tangent space to the global manifold. The logarithm map efficiently computes group differences and linearizes within the tangent space.

Therefore, manifold-based representations avoid common pitfalls such as singularities (e.g., gimbal lock) and over-parameterization (e.g., quaternions). They ensure that constraints, such as orthogonality and proper rigid-body transformations, are inherently maintained. This intrinsic consistency simplifies computations and improves the accuracy and robustness of state estimation, particularly in propagating the uncertainties.

Simply put, the original nonlinear propagations in vector space are transformed into local linear computations on the tangent space when perturbation states are sufficiently small, enabling efficient and accurate state estimation on manifolds.

Of course, there are other important mathematical foundations and tools, such as the *adjoint matrix* and *derivatives* on Lie groups, underpinning Lie theory. However, for brevity and focus, I do not introduce them in the main text of this thesis. For a deeper understanding of Lie theory in the context of state estimation problems, I refer readers to Solà et al. [226] and Barfoot [1].

A.3.4 Related Sensors and Observations

In this section, I provide an overview of the sensor measurements and models relevant to this thesis. Specifically, I describe the measurement characteristics and associated noise properties, which serve as the foundation for constructing probabilistic constraints (also known as factors) and handling measurement outliers in our factor graph optimization framework.

A.3.4.1 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) provides measurements of linear accelerations and angular velocities in the sensor's body frame [107, Chapter 4]. These measurements are typically used to estimate a robot's motion dynamics, including velocity, orientation, and position. A 6-axis IMU outputs two primary signals: accelerometer data $\mathbf{a} = [a_x \ a_y \ a_z]^T \in \mathbb{R}^3$ and gyroscope data $\boldsymbol{\omega} = [\omega_x \ \omega_y \ \omega_z]^T$. The accelerometer data measures specific force along the three axes of the sensor, while the gyroscope measures the angular velocity around the three axes.

Raw IMU Measurement

There are various types and grades of IMUs, distinguished by their performance, size, and price. The most commonly used IMUs in commercial applications are consumer-grade and industrial-grade IMUs, which strike a balance between cost and functionality. In contrast, tactical-grade IMUs, known for their significantly lower noise levels and higher precision, are primarily employed in survey tasks and military applications where accuracy is critical. For example, autonomous vehicles are often equipped with the Micro-Electro-Mechanical System (MEMS) IMU as it is compact and low-cost.

However, the MEMS-IMU is subject to various noise characteristics and imperfections, such as random walk, bias instability, and temperature-dependent noise, which can affect the accuracy and reliability. Therefore, in practical applications, the accelerometer and gyroscope data from the IMU are associated with inherent bias states, which can drift over time. These biases must also be estimated and accounted for to ensure accurate state estimation, resulting in the following equations

$$\hat{\mathbf{a}}_k = \mathbf{a}_k + \mathbf{b}_k^a - \mathbf{R}_{e,k}^b \mathbf{g}_e + \boldsymbol{\eta}_a, \quad (\text{A.110})$$

$$\hat{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_k + \mathbf{b}_k^\omega + \boldsymbol{\eta}_\omega \quad (\text{A.111})$$

to model the corrected IMU observations $\hat{\mathbf{a}}$ and $\hat{\mathbf{g}}$, where the measurements $\mathbf{a}$ and $\mathbf{g}$ are obtained from the accelerometer and gyroscope, respectively.

We assume that the corrected IMU measurements are corrupted by zero-mean Gaussian noise, follows $\boldsymbol{\eta}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_a^2)$ and $\boldsymbol{\eta}_\omega \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_\omega^2)$. The Earth's gravity vector $\mathbf{g}_e$ must be appropriately corrected during integration by transforming it using $\mathbf{R}_{e,t}^b$, the rotation matrix from the ECEF frame to the body frame. The time-varying biases are commonly modeled as a random walk process, represented by the following equations

$$\dot{\mathbf{b}}_t^a = \boldsymbol{\eta}_{b^a}, \quad \dot{\mathbf{b}}_t^\omega = \boldsymbol{\eta}_{b^\omega}, \quad (\text{A.112})$$

where we assume that the bias states are corrupted by zero-mean Gaussian noise, denoted as $\boldsymbol{\eta}_{b^a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b^a}^2)$ and $\boldsymbol{\eta}_{b^\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b^\omega}^2)$.

The noise characteristics of the MEMS-IMU can be quantified using Allan variance parameters derived from long-term measurement data. Open-sourced tools, such as Buchanan [228] and Rehder et al. [229], are available.

Popular IMU products often incorporate well-engineered filtering algorithms to reduce measurement noise and enhance the accuracy of orientation estimation by integrating raw accelerometer and gyroscope measurements. Additionally, IMUs may include embedded magnetometers for heading estimation, as well as pressure and temperature sensors to provide altitude information and compensate for environmental variations. By integrating additional measurements, such as GPS data, the IMU can be extended to form an *Inertial Navigation System* (INS), capable of determining a moving vehicle's position, velocity, and orientation.

Although the IMU is considered a sensor in autonomous systems, its observations are primarily treated as pseudo-inputs to propagate the system dynamics. For instance, IMU measurements are used to predict the system's state based on the motion model in the time-prediction step of a Kalman filter. In the following, I will introduce two of the most commonly used IMU mechanisms employed in filtering and batch-optimization methods.

Algorithm 8: Strap-Down Algorithm for IMU Integration in Navigation Frame

Input :

Initial geodetic coordinate: $\mathbf{p}_{k-1}^{\text{LLH}} = [\phi_{k-1} \ \lambda_{k-1} \ h_{k-1}]^T$
Initial velocity in the navigation frame: $\mathbf{v}_{k-1}^{\text{NED}} = [v_{k-1}^n \ v_{k-1}^e \ v_{k-1}^d]^T$
Initial orientation in the navigation frame: $\mathbf{R}_{k-1}^{\text{NED}}$
Accelerometer bias: $\mathbf{b}_{k-1}^a$
Gyroscope bias: $\mathbf{b}_{k-1}^g$
Raw IMU measurements: $\mathbf{a}_k, \boldsymbol{\omega}_k$
Earth rotation rate ω^e
Current Earth gravity vector $\mathbf{g}_{k-1}^{\text{NED}}$

Output: Integrated geodetic coordinate $\mathbf{p}_k^{\text{LLH}}$, velocity $\mathbf{v}_k^{\text{NED}}$, and orientation $\mathbf{R}_k^{\text{NED}}$

1 **Correct IMU measurements:**

$$2 \quad \hat{\mathbf{a}}_k = \mathbf{a}_k - \mathbf{b}_{k-1}^a$$

$$3 \quad \hat{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_k - \mathbf{b}_{k-1}^g$$

4 **Correct current state derivatives**

5 /* Earth rotation rate in n -frame [230, pp. 390]: */

$$6 \quad \boldsymbol{\omega}_{k,\text{er}}^{\text{NED}} = [\omega^e \cos \phi, 0, -\omega^e \sin \phi]^T$$

7 /* Earth transport rate in n -frame [230, pp. 49]: */

$$8 \quad \boldsymbol{\omega}_{k,\text{tr}}^{\text{NED}} = \left[\frac{v_{k-1}^e}{R_a + h_{k-1}}, -\frac{v_{k-1}^n}{R_a + h_{k-1}}, \frac{v_{k-1}^e \tan \phi_{k-1}}{R_a + h_{k-1}} \right]^T$$

9 /* Correct due to Earth rotation [230, pp. 48]: */

$$10 \quad \bar{\boldsymbol{\omega}}_k = \hat{\boldsymbol{\omega}}_k - (\mathbf{R}_{k-1}^{\text{NED}})^T (\boldsymbol{\omega}_{k,\text{er}}^{\text{NED}} + \boldsymbol{\omega}_{k,\text{tr}}^{\text{NED}})$$

11 /* Calculate current acceleration [231, pp. 390]: */

$$12 \quad \dot{\mathbf{v}}_k = \mathbf{R}_{k-1}^{\text{NED}} \hat{\mathbf{a}}_k - (2 \cdot (\boldsymbol{\omega}_{k,\text{er}}^{\text{NED}})^\wedge + (\boldsymbol{\omega}_{k,\text{tr}}^{\text{NED}})^\wedge) \mathbf{v}_{k-1}^{\text{NED}} + \mathbf{g}_{k-1}^{\text{NED}} \quad /* \text{ Calculate current velocity [231, pp. 389]} */$$

$$13 \quad \dot{\mathbf{p}}_k = \left[\frac{v_{k-1}^n}{(R_a + h_{k-1})}, \frac{v_{k-1}^e}{(R_a + h_{k-1}) \cos \phi_{k-1}}, -v_{k-1}^d \right]^T$$

14 **Integrating state derivatives using a numerical solver (e.g., runge-kutta):**

$$15 \quad \mathbf{p}_k^{\text{LLH}} = \int \dot{\mathbf{p}}_k$$

$$16 \quad \mathbf{v}_k^{\text{NED}} = \int \dot{\mathbf{v}}_k$$

17 /* Integrating the orientation in quaternion using equations (3.117-3.118) [230, pp. 47]: */

$$18 \quad \mathbf{q}_{k-1}^{\text{NED}} \leftarrow \mathbf{R}_{k-1}^{\text{NED}}; \quad // \text{ Transform the rotation matrix to quaternion}$$

$$19 \quad \mathbf{q}_k^{\text{NED}} = \mathbf{q}_{k-1}^{\text{NED}} \odot \int \bar{\boldsymbol{\omega}}_k; \quad // \odot \text{ denotes the equation (3.117)}$$

$$20 \quad \mathbf{R}_k^{\text{NED}} \leftarrow \mathbf{q}_k^{\text{NED}}; \quad // \text{ Transform the quaternion to rotation matrix}$$

$$21 \quad \text{return } \{\mathbf{p}_k^{\text{LLH}} \ \mathbf{v}_k^{\text{NED}} \ \mathbf{R}_k^{\text{NED}}\}$$

Strap-Down Algorithm

A typical and widely used IMU mechanism for calculating a vehicle's state using raw accelerometer and gyroscope data is known as the *Strap-Down Algorithm*. It processes inertial measurements directly in the body frame and computes the vehicle's orientation, velocity, and position by integrating them over time. The strap-down mechanism can be implemented as in Algorithm 8 based on Wendel [230, Chapter 3.3] and Farrell [231, Chapter 11].

This method is computationally efficient and forms the backbone of many navigation algorithms (e.g., Kalman filters), although it requires careful handling of sensor noise and bias to maintain accuracy. The output of Algorithm 8 represents a global state within its reference frame, making it suitable for integrating other sensors or algorithms for enhanced localization and navigation.

In Chapter 3, I use the strap-down algorithm to implement an unscented Kalman filter, a baseline benchmark method for factor graph optimization (see Section 3.5).

IMU Pre-Integration

Another widely used IMU mechanism is referred to as *IMU pre-integration*, initially introduced by Lupton and Sukkarieh [232] and later extended to $SE(3)$ by Forster et al. [233] to avoid singularities. Unlike the strap-down algorithm, which integrates state derivatives in the global frame, the IMU pre-integration method computes relative state increments $\Delta \tilde{\mathbf{x}}_{ij} = \{\Delta \tilde{\mathbf{R}}_{ij}, \Delta \tilde{\mathbf{v}}_{ij}, \Delta \tilde{\mathbf{p}}_{ij}\}$ over a time interval Δt_{ij} . These increments are derived from bias-corrected IMU measurements and represent the relative changes in rotation, velocity, and position between two timestamps. Notably, the time difference Δt_{ij} is typically much larger than the IMU's sampling rate (e.g., $\Delta t_{ij} = 0.1 \text{ s} \gg \Delta t_{\text{IMU}} = 0.01 \text{ s}$), enabling efficient integration and storage of measurements for optimization algorithms. The following equations are employed to calculate the pre-integrated increments

$$\Delta \tilde{\mathbf{R}}_{ij} = \prod_{k=i}^{j-1} \text{Exp}((\boldsymbol{\omega}_k - \mathbf{b}_i^g - \boldsymbol{\eta}_k^g)\Delta t), \quad (\text{A.113})$$

$$\Delta \tilde{\mathbf{v}}_{ij} = \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{ik}(\mathbf{a}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^a)\Delta t, \quad (\text{A.114})$$

$$\Delta \tilde{\mathbf{p}}_{ij} = \sum_{k=i}^{j-1} \left[\Delta \mathbf{v}_{ik}\Delta t + \frac{1}{2}\Delta \tilde{\mathbf{R}}_{ik}(\mathbf{a}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^a)\Delta t^2 \right], \quad (\text{A.115})$$

where the raw vehicle acceleration $\mathbf{a}$ and rotation rate $\boldsymbol{\omega}$ from the IMU are integrated. The pre-defined noise parameters $\{\boldsymbol{\eta}_a, \boldsymbol{\eta}_g\}$ are propagated to acquire the covariance matrix Σ^{imu} [233].

Although IMU pre-integration does not represent the true physical motion increments—since it integrates only the relative motion changes between the vehicle states $\mathbf{x}_i$ and $\mathbf{x}_j$, excluding gravitational and Coriolis effects—it offers significant computational advantages. The pre-integrated IMU measurements are independent of the vehicle state $\mathbf{x}_j$, allowing them to be directly applied with the current bias estimation to propagate the state $\mathbf{x}_j$ for error evaluation during each optimization iteration. Given that the biases $[\mathbf{b}_t^a, \mathbf{b}_t^\omega]$ typically vary slowly, the IMU measurements can be pre-integrated without introducing additional state variables into the graph. This approach significantly reduces computational complexity, making it particularly effective for batch optimization algorithms.

I employ the IMU pre-integration method to formulate probabilistic IMU factors within the factor graph optimization framework, as elaborated in Section 3.5 and Section 3.6.

A.3.4.2 Global Navigation Satellite System

Autonomous vehicles may utilize *global navigation satellite systems* (GNSS) to achieve global positioning in large-scale environments [107]. Generally, a single-antenna GNSS receiver can provide both absolute *position, velocity, and time (PVT)* solutions and raw observations from satellites (e.g., GPS, Galileo, Beidou) in orbit. A GNSS receiver equipped with multiple antennas or an inertial sensor can also produce a *position, velocity, and altitude (PVA) solution*. As the pose and velocity of the GNSS solution can be directly associated with the state variables, I present only the measurement models for the raw GNSS observations: pseudorange ρ and Doppler-shift Δf_k in this section.

In localization approaches that tightly fuse GNSS observations, *pseudorange* and *Doppler-shift* are commonly used and well-studied [107]. The pseudorange ρ represents a geometric distance between the phase center of the GNSS antenna and the associated satellite, which contains several range delays due to satellite orbit bias and atmospheric delays. The pseudorange can be modeled with respect to the antenna position as

$$\rho_k = \underbrace{\|\mathbf{p}_{\text{ant}}^e - \mathbf{p}_{\text{sat},k}^e\|}_{\text{1-D geometric range}} + c_b - c_{b,\text{sat}} + T_t^k + I_t^k + M_t^k + w_{\rho,t}^k, \quad (\text{A.116})$$

where the vectors $\mathbf{p}_{\text{ant}}^e$ and $\mathbf{p}_{\text{sat},k}^e$ represent the positions of the GNSS antenna and k -th satellite in ECEF frame¹¹, respectively. The variables c_b and $c_{b,\text{sat}}$ represent the bias due to receiver clock delay and satellite clock delay. The tropospheric, ionospheric, and multipath delays are denoted as T_k , I_k and M_k , respectively. The pseudorange noise is $w_{\rho,t}^k$.

¹¹I represent the vehicle and satellite position in the ECEF frame instead of the Earth-centered inertial frame for clarity by assuming that the GNSS pre-processing has calibrated the earth rotation during GNSS signal propagation.

The Doppler-shift¹² Δf_k measures the difference in frequency between the original and received carrier signal of a satellite, which is usually obtained in the carrier-phase tracking loop [107]. With this observation, the vehicle velocity related to the satellite velocity can be represented as

$$-\lambda_c \Delta f_k = \underbrace{(\mathbf{u}_{\text{ant}}^{\text{sat}})^T (\mathbf{v}_{\text{ant}}^e - \mathbf{v}_{\text{sat},k}^e)}_{\text{1-D range rate}} + c_d - c_{d,\text{sat}} + w_{\Delta f_t^k}. \quad (\text{A.117})$$

In (A.117), the constant λ_c is the wavelength of the GNSS signal. The unit vector $\mathbf{u}_{\text{ant}}^{\text{sat}}$ represents the direction from the antenna to the k -th satellite. I denote the satellite and antenna velocities in the ECEF frame by $\mathbf{v}_{\text{sat},k}^e$ and $\mathbf{v}_{\text{ant}}^e$, respectively. The receiver clock drift and the satellite clock drift are given as c_d and $c_{d,\text{sat}}$. I use the scalar $w_{\Delta f_t^k}$ to denote the noise of the measured Doppler-shift.

Figure A.10 illustrates the various delays encountered during GNSS signal propagation, which contribute to significant biases in the raw GNSS pseudorange measurements compared to the true geometric range between the vehicle and each satellite. These delays stem from factors such as satellite and receiver clock offsets, atmospheric effects, and instrumental errors, all of which degrade positioning accuracy if not properly accounted for.

To use the pseudorange and Doppler-shift observations in our state estimators, we assume that the satellite clock delay $c_{b,\text{sat}}$ and drift $c_{d,\text{sat}}$ are eliminated using the received navigation messages in a GNSS pre-processing process. We used well-calibrated correction data from a reference station to cancel the tropospheric and ionospheric interference. The variable M represents the multipath delay of the pseudorange observation, which will be thoroughly examined in Chapter 4 and Chapter 5. We filter out all GNSS observations from satellites with an elevation angle less than 15° .¹³

The *multipath* and *non-line-of-sight* (NLOS) signals indicate the major challenge in GNSS-based localization approaches, as they usually present complex error dynamics that cannot be sufficiently modeled using classic geometric or statistical methods. Figure A.10b illustrates these delays. NLOS occurs when the direct line-of-sight signal between the satellite and the receiver is obstructed by objects such as buildings or terrain, leading to signal reflection or diffraction. Multipath reception happens when the GNSS signals bounce off surrounding surfaces (e.g., buildings, vehicles, or the ground) before reaching the receiver, causing multiple delayed copies of the signal to interfere with the direct signal. Both phenomena introduce errors in the pseudorange and carrier phase measurements, degrading positioning accuracy and reliability. Addressing these challenges requires advanced signal processing and robust estimation techniques [235].

¹²In some literature, the Doppler-shift is also formulated as deltarange or pseudorange rate measurement [107, 234]. A positive Doppler-shift denotes that the receiver is approaching the tracked satellite.

¹³This is an ad-hoc choice, generally used in ground vehicle navigation approaches and GNSS receivers.

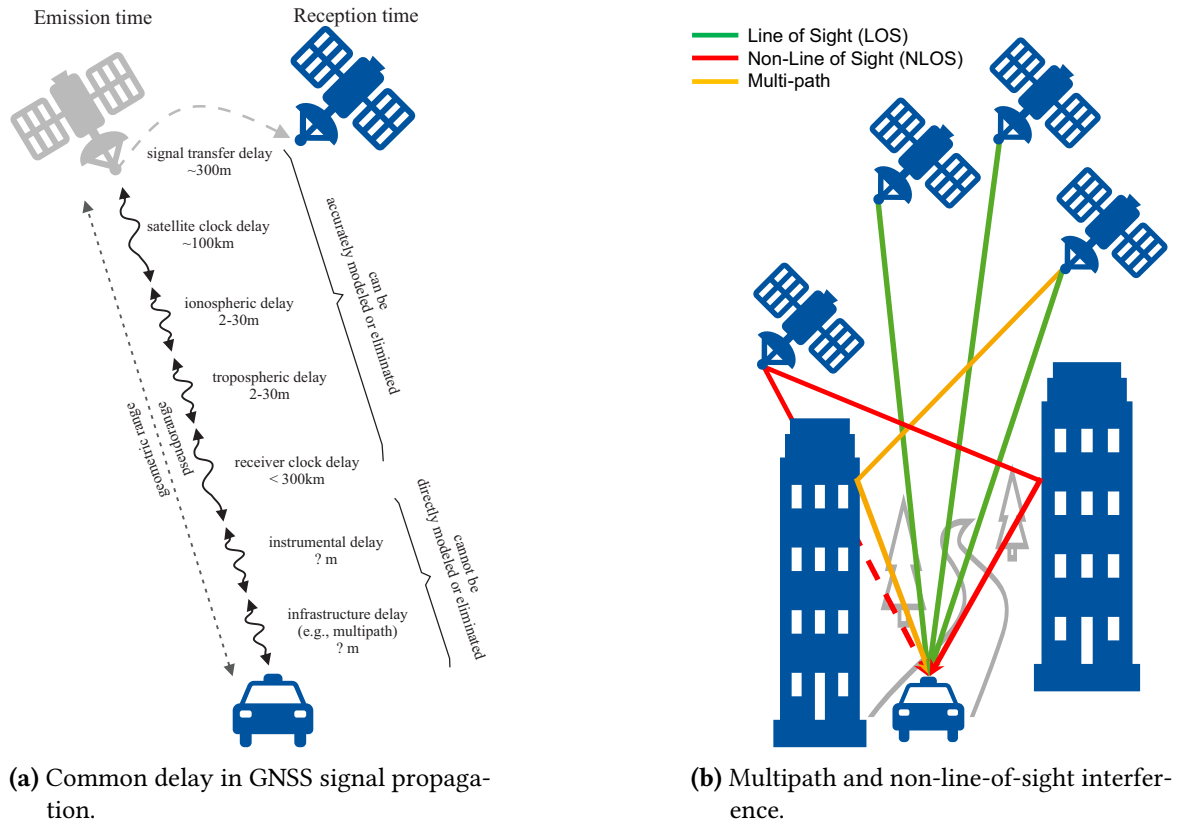


Figure A.10: GNSS delays and interference.

Consequently, raw GNSS pseudorange measurements are often highly susceptible to noise with challenging distributions, including long-tailed or multimodal characteristics. This makes the development of robust outlier handling techniques tailored to this measurement model a crucial and compelling research problem, which this thesis aims to address.

In Section B.3.6, I will demonstrate how raw GNSS observations can be utilized to compute state updates and formulate constraints within the estimation framework, and come back to their interferences in challenging areas. Additionally, other prominent GNSS observation types, such as carrier phase and multi-differenced observations, are available to establish more advanced sensor models for state estimation applications, providing higher precision and robustness under challenging conditions. However, in this thesis, I focus solely on pseudorange and Doppler-shift measurements, as they are sufficient to address our research objectives related to handling measurement outliers. For a more comprehensive explanation of GNSS and its related algorithms, I refer readers to Groves [107] and Farrell [231].

A.3.4.3 LiDAR

Light Detection and Ranging (LiDAR) is a sensor technology that uses laser beams to measure distances of the emitted light pulses. It generates highly accurate 3-D point clouds representing the surrounding environment, making it indispensable in various applications, such as autonomous driving, robotics, and mapping.

In vehicle localization applications, the lidar sensor plays a crucial role by providing highly accurate 3-D point cloud data of the surrounding environment that can be used to estimate the vehicle's position in a local frame. The basic concept of utilizing LiDAR data is known as *scan matching*, which computes motion increments by solving an optimization problem to align corresponding points. This can involve aligning two successive scans (frame-to-frame) or aligning the current scan to a (local) pre-built model (scan-to-map).

Over the past decades, scan matching has been extensively studied, resulting in two widely used approaches: global scan-matching and feature-based matching, both of which are employed in numerous applications [236, 237]. Methods that compute incremental or local (non-globally consistent) states are referred to as *LiDAR odometry*. In contrast, globally consistent vehicle localization can be achieved through *LiDAR SLAM* methods (see Section A.3.6), which incorporate global constraints, such as loop closures, to correct drift and ensure global consistency.

Commonly used algorithms to perform a global scan-matching include the *Iterative Closest Point* (ICP) and *Normal Distributions Transform* (NDT). The ICP-based algorithms align two point clouds by iteratively minimizing the distance between corresponding points [238], whereas an NDT algorithm utilizes probability distributions to represent the point cloud, thereby reducing sensitivity to noise and improving computational efficiency [239]. However, performing global scan-matching using all received points is computationally intensive and may not meet the real-time requirements of specific applications. Feature-based approaches, on the other hand, aim to extract distinctive features (e.g., edges or planes) from the point cloud, significantly reducing the computational burden while preserving key structural information for efficient and reliable matching.

Among all different methods, LiDAR Odometry and Mapping [240] and its derivatives [120, 241] are widely recognized as popular LiDAR localization techniques that rely on feature-based scan matching. These approaches extract edge and planar features from point clouds, enabling efficient and accurate motion estimation, and thus serve as a benchmark in LiDAR odometry and SLAM research.

In this method, the coordinates of raw lidar points acquired at different timestamps are recalibrated using the IMU measurement to the original timestamp of the lidar scan. The calibrated points are classified into edge and planar features, $\mathbf{F}_k = \{\mathbf{F}_k^e, \mathbf{F}_k^p\}$, based on the smoothness metric. In scan registration, all m features in $\mathbf{F}_{k+1}$ of the current scan are associated with pose priors $\mathbf{T}_{k+1,1:m}^w$ and used to find the best transformation $\Delta \mathbf{T}_{k,k+1}^w$ from

the last laser scan by solving an optimization problem that takes the distance between the corresponding features in $\mathbf{F}_k$ using a Gauss-Newton algorithm. In [120], a lidar-centric SLAM approach is presented that optionally fuses the GNSS positioning solution. This approach can only present accurate state estimates if the scan registration converges and sufficient global references (e.g., GNSS position or loop closure) are available.

In this thesis, I adapt the feature-based lidar odometry introduced in Zhang and Singh [240] to acquire motion increments between two successive states, as described in Chapter 3.

A.3.4.4 Optical Speed Sensor

I employ a high-grade vehicle optical speed sensor (see Section 2.2) designed for non-contact measurement of vehicle dynamics. Utilizing proven dual-beam optical measurement technology, it provides accurate data on speed, distance, and acceleration on various surfaces, including challenging conditions like wet or uneven roads.

The speed sensor provides unbiased 2-D velocity observations $\bar{\mathbf{v}}_t^b$ in the body frame at 100 Hz. The 2-D velocity observations can be associated with the vehicle velocity in the state vector using

$$\tilde{\mathbf{v}}_t^b = \begin{bmatrix} \tilde{v}_{t,x}^b \\ \tilde{v}_{t,y}^b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{R}_e^b \mathbf{v}_b^e, \quad (\text{A.118})$$

where the vector $\tilde{\mathbf{v}}_t^b$ represents the observed 2-D velocity components in frame b , which can be evaluated with the vehicle velocity variable $\mathbf{v}_b^e$ transformed with the inverse rotation matrix $\mathbf{R}_e^b$ back to frame b .

A.3.5 Sensor Fusion

As discussed in Chapter 1, incorporating multiple sensor observations to perform sensor fusion is a key approach to improving accuracy and consistency in vehicle localization. Sensor fusion methods are generally categorized into two main types: *loosely coupled* and *tightly coupled* approaches. These categories are briefly introduced in this section to provide a foundational understanding of their roles and applications in localization.

A.3.5.1 Loosely Coupled Sensor Fusion

In loosely coupled fusion methods for vehicle localization, each sensor independently and directly observes the vehicle's state, such as GNSS PVT solution [107, Chapter 12.1]. Fusing these observations does not necessitate highly nonlinear measurement models, as measurement corrections can be directly calculated from the sensor outputs.

Thanks to its simplicity and robust state observation capabilities, loosely coupled fusion is widely employed in standard localization methods that utilize different sensors, such as GNSS/INS fusion [242], a lidar-based approach [120], and a visual-based approach [243].

However, the robustness and accuracy of loosely coupled methods are highly dependent on the quality of the sensor measurements. As each individual sensor does not provide redundant observations to constrain the to-be-estimated states, the performance of loosely coupled approaches can degrade significantly or even fail immediately if a sensor provides unreliable or erroneous data. I will demonstrate this problem with sufficient experiments in Chapter 3.

A.3.5.2 Tightly Coupled Sensor Fusion

In contrast to loose coupling, tightly coupled fusion directly incorporates raw measurement data from sensors to constrain the vehicle's states within nonlinear sensor models. Instead of providing direct state observations, sensors in tightly coupled fusion deliver multiple simultaneous observations (e.g., LiDAR points or satellite ranges). This approach significantly enhances state observability, particularly in scenarios where only a subset of sensor measurements may be corrupted, thereby ensuring more robust and accurate localization.

Another advantage of fusing measurements in a tight coupling is the explicit modeling and monitoring of each sensor observation. This approach enables the application of advanced techniques to accurately model uncertainties and effectively handle outliers, resulting in improved robustness and reliability in state estimation. Therefore, tightly coupled fusion demonstrates its superiority across many applications involving different sensors, such as GNSS [M1, 105, 106], lidar [244, 245], and cameras [12, 123, 124].

Similar to the previous work of Wen et al. [104], which compares both loosely and tightly coupled fusion approaches, this thesis focuses on tightly coupled fusion of GNSS observations in the following chapters, along with an in-depth discussion on fusion mechanisms.

A.3.6 Simultaneous Localization and Mapping

Another essential technique for vehicle localization is *Simultaneous Localization and Mapping* (SLAM), which was initially introduced by Smith and Cheeseman [246] in the 1980s. This technique specifically addresses the localization problem in scenarios where a global reference or reliable observation of the robot is unavailable or untrustworthy. In such cases, the vehicle must simultaneously construct a map of its environment while determining its position within that map. Numerous works and studies employing various types of local sensors have been conducted over the years, striving to achieve highly accurate, robust, and

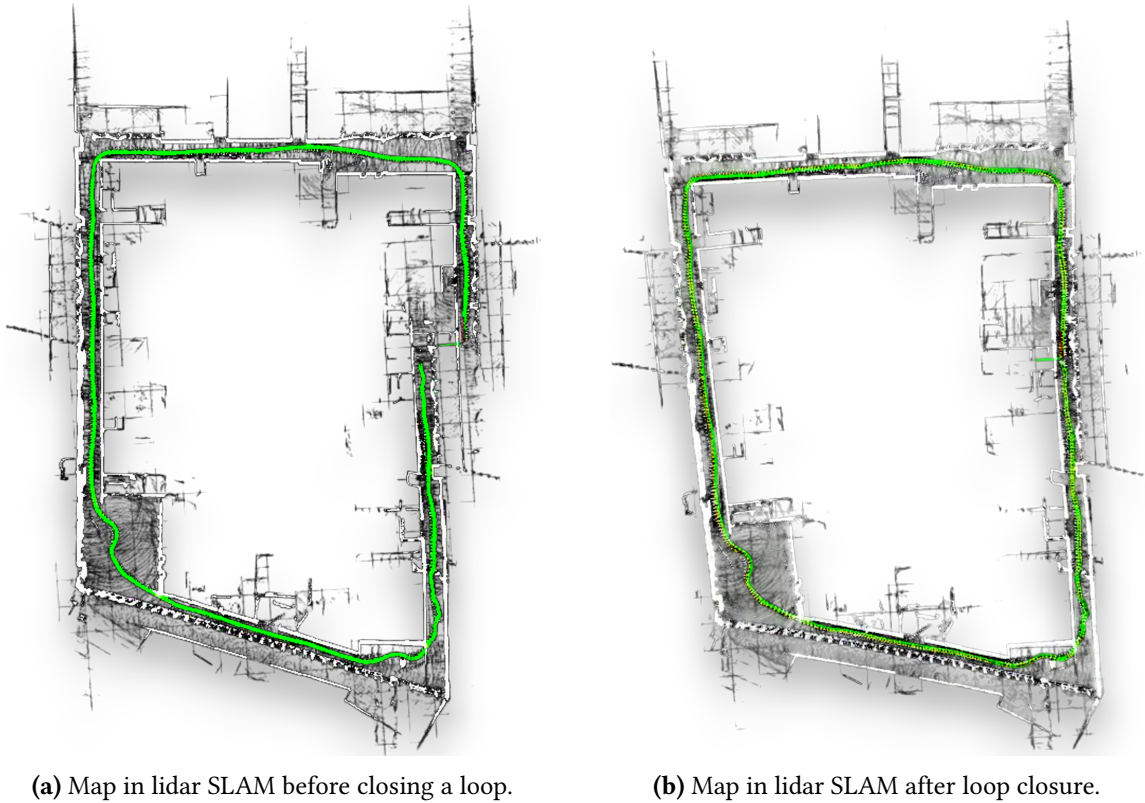


Figure A.11: Lidar map before and after fusing the loop-closure constraints in SLAM.

efficient SLAM algorithms, as highlighted in Cadena et al. [247], Sahili et al. [248], and Ebadi et al. [249].

Generally, SLAM algorithms strive to achieve *full SLAM*, where the estimated trajectory remains globally consistent, effectively eliminating drift over the entire operational lifetime. In scenarios where globally consistent measurements (e.g., GNSS) are unavailable, SLAM algorithms aim to acquire pseudo-global constraints, known as *loop-closure constraints*, by recognizing previously visited locations in the simultaneously built map. A basic loop-closure constraint involves estimating a global pose x_t^w in the world frame w , which is derived, for example, by matching the current scan with the map. Figure A.11¹⁴ illustrates the SLAM process before and after incorporating loop-closure constraints, demonstrating how these constraints effectively eliminate the accumulated drift and restore global consistency in the estimated trajectory.

However, our intention in introducing the SLAM techniques in this thesis is to provide a clear perspective on **the limitations of SLAM algorithms**, mainly when applied to outdoor and large-scale environments.

¹⁴Source <https://www.youtube.com/watch?v=OV6wNr62nqQ>

Vehicles operating in outdoor environments typically traverse long routes and endure extended periods of operation, making the occurrence of loop-closure constraints a rare event. As there are other sensing opportunities, such as GNSS and other radio sources, are available, SLAM techniques do not remain the sole solution for vehicle localization, particularly in scenarios where computational power is limited, and real-time solutions are required. Therefore, SLAM techniques are generally regarded as complementary options, enhancing the system's observability when global references, such as GNSS, are temporarily unavailable. Additionally, planning and control algorithms can leverage SLAM to construct a local environment model for the vehicle, enabling the geometric representation of static and dynamic obstacles to ensure safe autonomous operations.

In this thesis, I implemented a local SLAM using lidar odometry that matches lidar features to the local map, aiming to obtain locally consistent state observations.

A.3.7 Performance Metrics of Vehicle Localization

This section provides a brief introduction to the relevant performance metrics used to validate and evaluate the proposed state estimators in this thesis.

Vehicle localization approaches that rely on local sensors (e.g., LiDARs) often suffer from trajectory drift over time. Consequently, these approaches typically distinguish between the definitions of *accuracy* and *precision*. An estimated trajectory is considered *accurate* when it closely aligns with the true trajectory on a global scale, indicating that systematic localization errors and drifts have been effectively mitigated. *Precision*, on the other hand, focuses on the repeatability and consistency of relative motion estimates between consecutive poses, even if the trajectory itself deviates from the true path. For instance, the estimated trajectory of LiDAR-based SLAM approaches may exhibit high precision but lack global accuracy if no global constraints are available, as illustrated in Figure A.11.

The most commonly used error metrics to validate localization approaches include the *absolute trajectory error* (ATE) and *relative pose error* (RPE) [250].

The APE aligns the estimated trajectory with the ground-truth trajectory and measures the deviation between them in a global coordinate frame. Therefore, it is often used to evaluate the global accuracy of the localization algorithms. The mean absolute error can be calculated using the RMSE of the APE over K -timestamps as

$$\text{ATE} = \sqrt{\frac{1}{K} \sum_{k=1}^K \left\| \mathbf{x}_k^{\text{gt}} (\mathbf{x}_k^{\text{est}})^{-1} \right\|^2}, \quad (\text{A.119})$$

where the estimated and ground-truth pose in the same reference frame (e.g., ECEF) are denoted as $\mathbf{p}_k^{\text{est}}$ and $\mathbf{p}_k^{\text{gt}}$, respectively.

On the contrary, the RPE measures the error in relative transformations (position and orientation) between consecutive poses. It is thus oftentimes used to evaluate the trajectory precision (local accuracy). The RMSE of the RPE over K -timestamps can be computed as

$$\text{RPE} = \sqrt{\frac{1}{K} \sum_{k=1}^{K-1} \left\| (\mathbf{x}_k^{\text{gt}})^{-1} \mathbf{x}_{k+1}^{\text{gt}} ((\mathbf{x}_k^{\text{est}})^{-1} \mathbf{x}_{k+1}^{\text{est}})^{-1} \right\|^2}. \quad (\text{A.120})$$

Besides the error metrics, I employ Pythagoras' theorem implemented in the Open Motion Planning Library¹⁵ (OMPL) to calculate the trajectory smoothness (contrary to trajectory roughness) for all test sequences, aiming to provide a relative performance metric.

The smoothness is given as the sum of angles between all path segments in the local-world frame, as denoted in (A.121), where the variables a_i , b_i , and c_i are the lengths of the trajectory segments containing three successive vehicle positions in the Euclidean frame. For the same test sequence with k vehicle positions, a smaller s shows a higher smoothness of the trajectory. In this thesis, I used the propagated states from the state publisher at a high frequency to calculate the mean smoothness, s :

$$s = \sqrt{\frac{1}{K-1} \sum_{k=2}^{K-1} \left(\frac{2(\pi - \arccos \frac{a_k^2 + b_k^2 - c_k^2}{2a_k b_k})}{a_k + b_k} \right)^2}. \quad (\text{A.121})$$

¹⁵<https://ompl.kavrakilab.org/>

A.4 Primer on Machine Learning

The final section of Chapter 2 provides a brief introduction to the basic concepts and models in machine learning and deep learning that are essential for understanding the methods used in Chapter 5. While I do not aim to present a comprehensive review of all related models, our goal is to offer insights into those models specifically tailored to address the research objectives of this thesis.

A.4.1 Learning from Data

We have discussed multiple models that explicitly represent the mathematical foundations of system dynamics and sensing processes for solving state estimation problems in Section A.2. However, models of dynamic systems often become unknown or unobservable due to incomplete information, nonlinearity, or unpredictable uncertainties when interacting with their environments.

As a result, learning these unknown models from their related data forms a cornerstone of modern scientific inquiry. This principle dates back to the 18th century, where the foundations of probabilistic reasoning and statistical inference were established [195, 251].

Over time, the concept of *learning from data* has evolved significantly, branching into various disciplines that can be broadly categorized into classic machine learning [56] and deep learning [252]. Classic machine learning focuses on algorithms such as regression, decision trees, and support vector machines, which rely on handcrafted features and statistical principles. In contrast, deep learning utilizes multi-layered neural networks to automatically learn hierarchical representations from data, enabling breakthroughs in fields such as computer vision, natural language processing, and robotics.

Models fitted from data are referred to as *data-driven* or *black-box* models, which often lack the explicit interpretability of conventional models built from, e.g., ordinary differential or geometric equations. A data-driven model can be obtained through three primary learning mechanisms: *supervised*, *unsupervised*, and *reinforcement* learning.

In supervised learning, the model is trained on labeled data, where the input-output relationships are explicitly provided, enabling the model to learn mappings or predictions, such as classification or regression tasks. On the other hand, unsupervised learning deals with unlabeled data, focusing on uncovering hidden patterns or structures within the data, such as clustering or dimensionality reduction. Lastly, reinforcement learning involves training a model, or agent, to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.

Each mechanism is tailored to address specific problems and data scenarios, yet these mechanisms can be combined to formulate more extensive and versatile learning paradigms,

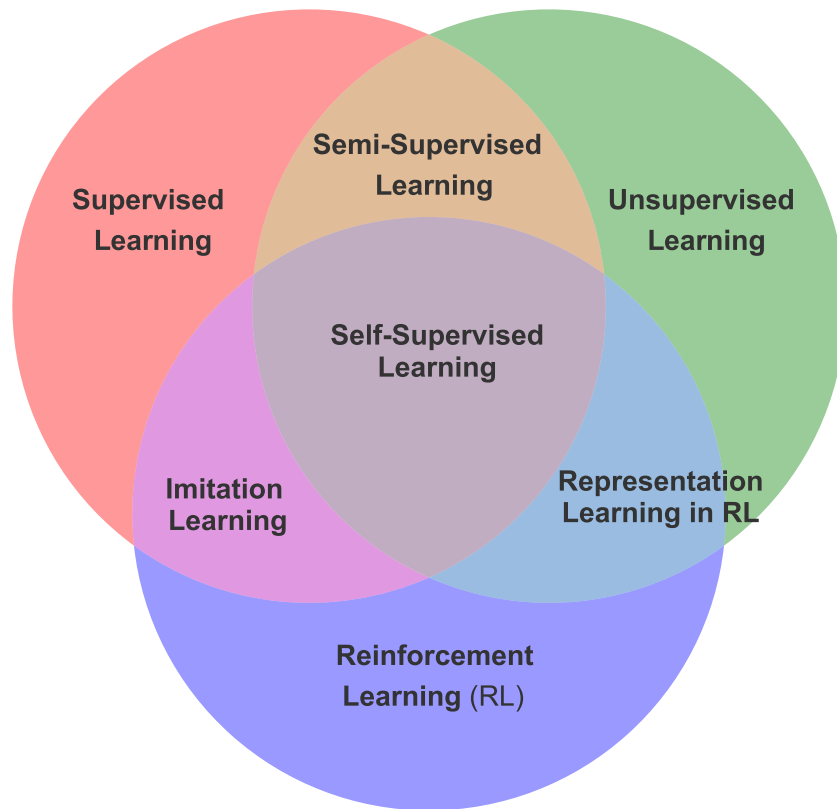


Figure A.12: Venn diagram of learning mechanisms to build data-driven models.

such as *self-supervised* learning. Figure A.12 illustrates the learning mechanisms in a Venn diagram, highlighting the relationships and overlaps among them.

In Chapter 5, I will utilize supervised learning to train various data-driven models for classifying faulty GNSS observations and predicting pseudorange errors, thereby highlighting learning-based methods for handling measurement outliers in state estimation. The following sections will introduce these models.

A.4.2 Classic Machine Learning

Classic machine learning models are foundational algorithms for pattern recognition and prediction, which rely on statistical and mathematical principles to address regression for continuous data, classification for discrete labels, clustering for grouping data, and dimensionality reduction for simplifying datasets. Popular machine-learning approaches include *decision tree*, *boosting*, and *support vector machine*, which are used in this thesis.

A.4.2.1 XGBoost

A popular model proposing an ensemble learning technique is called *boosting*, which combines multiple weak learners, typically decision trees, to create a strong predictive model by sequentially correcting errors made by previous models. Among all boosting derivatives, XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of boosting that uses gradient-based optimization to minimize errors, incorporates regularization to prevent overfitting, and supports parallel computation for faster training [253].

As a gradient-boosting ensemble algorithm, XGBoost iteratively builds decision trees to predict the residuals of previous predictions, refining the model until a stopping criterion, such as the maximum number of trees or minimal loss improvement, is met.

Renowned for its efficiency and scalability [253], XGBoost excels with large datasets and many features, leveraging incremental learning to easily update and generalize models with new data. It also utilizes parallel computation on multi-core CPUs and distributed computation on clusters, significantly enhancing training speed. Figure A.13 illustrates the basic concept of XGBoost.

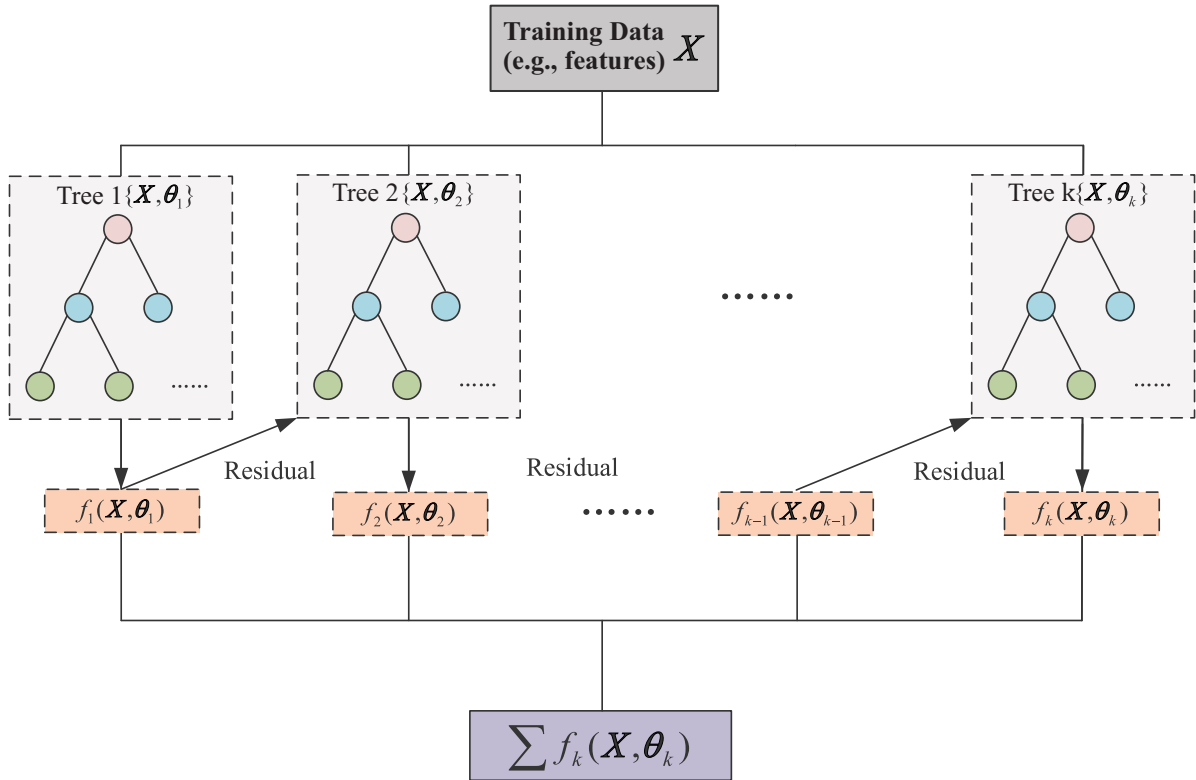


Figure A.13: Basic concept of XGBoost following Guo et al. [254].

XGBoost is highly efficient, scalable, and performs well on large datasets, effectively handling both numerical and categorical features. It incorporates regularization to reduce overfitting and supports parallel and distributed computation for faster training. However, XGBoost can be complex to tune due to the numerous hyperparameters it has, may require significant computational resources, and can struggle with interpretability compared to simpler models.

A.4.2.2 Support Vector Machine

Support Vector Machine (SVM) is a classical supervised machine learning algorithm widely used for classification and regression [255]. It identifies a hyperplane that separates data into classes by maximizing the margin between the hyperplane and the closest data points from each class, as shown in Figure A.14, ensuring better classification performance. This hyperplane serves as a single dividing line for binary classification, whereas multi-class classification employs multiple hyperplanes. When data is not linearly separable, non-linear kernel functions transform it into a higher-dimensional space, allowing for separation.

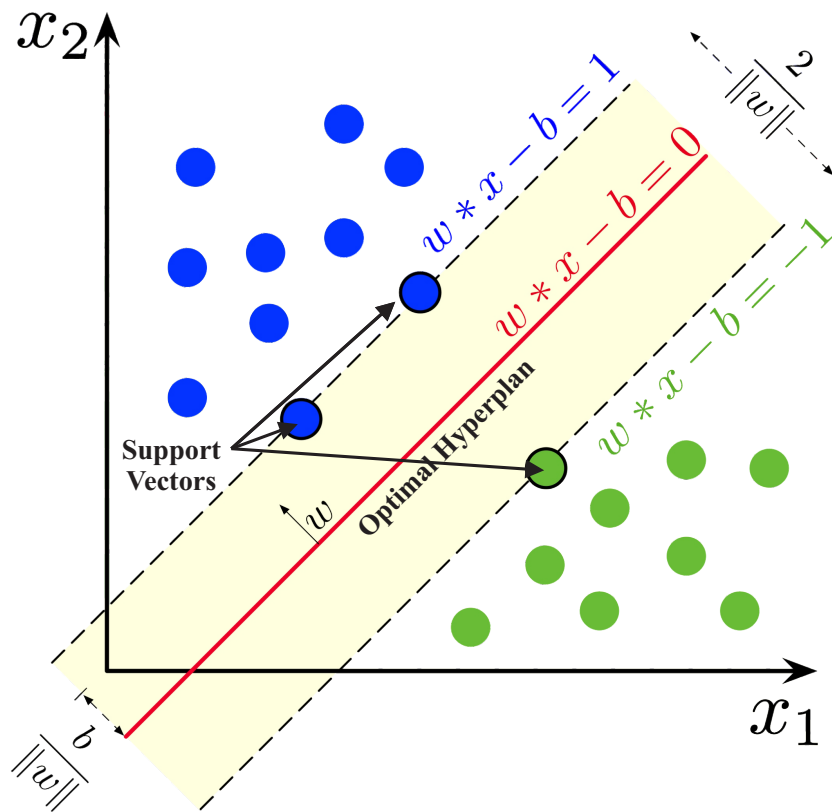


Figure A.14: Basic concept of support vector machine.

SVMs require solving a quadratic programming problem under constraints defined by the Lagrangian formulation and Karush-Kuhn-Tucker conditions. However, they can become computationally expensive for large datasets, particularly with non-linear kernels [256]. Additionally, SVMs are sensitive to hyperparameters, such as kernel choice, and may overfit when margins are small or kernels are overly complex. For tasks such as multipath error prediction, which are nonlinear and complex, careful kernel design is crucial for achieving optimal performance.

A.4.3 Deep Learning

Deep learning has a long and rich history, evolving into a leading learning paradigm since the 1990s, driven by advancements in computational power, the availability of large datasets, and the development of efficient algorithms, such as backpropagation.

Unlike classic models, Deep learning is needed to overcome the limitations of classic machine learning models, which struggle with high-dimensional data, complex patterns, and the need for manual feature engineering [257]. Deep learning techniques typically construct models with larger receptive fields, enabling them to capture broader context, hierarchical patterns, and sufficient spatial information within the data.

Popular models of deep learning include convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) and their variants like long short-term memory (LSTM) and gated recurrent units (GRUs) for sequential data, transformers for natural language processing and vision tasks, generative adversarial networks (GANs) for generating data, and autoencoders for unsupervised learning and dimensionality reduction. This section provides a brief introduction to the crucial components and key models of deep learning relevant to this thesis.

A.4.3.1 Multilayer Perceptron

A *perceptron* forms the foundation of artificial neural networks and can be viewed as a single-layer feedforward neural network designed for binary classification tasks, as illustrated in Fig. A.15. Developed in the 1950s by Rosenblatt [258], it is one of the earliest and simplest models of neural networks.

The perceptron takes a set of inputs, applies weights, and passes the weighted sum through an activation function to produce, e.g., a binary output (0 or 1) for classification or a continuous value for regression. It can be modeled using the following equation

$$\hat{y} = \varphi\left(\sum_{i=1}^N w_i x^i + b\right) = \varphi(\mathbf{w}^T \mathbf{x} + b), \quad (\text{A.122})$$

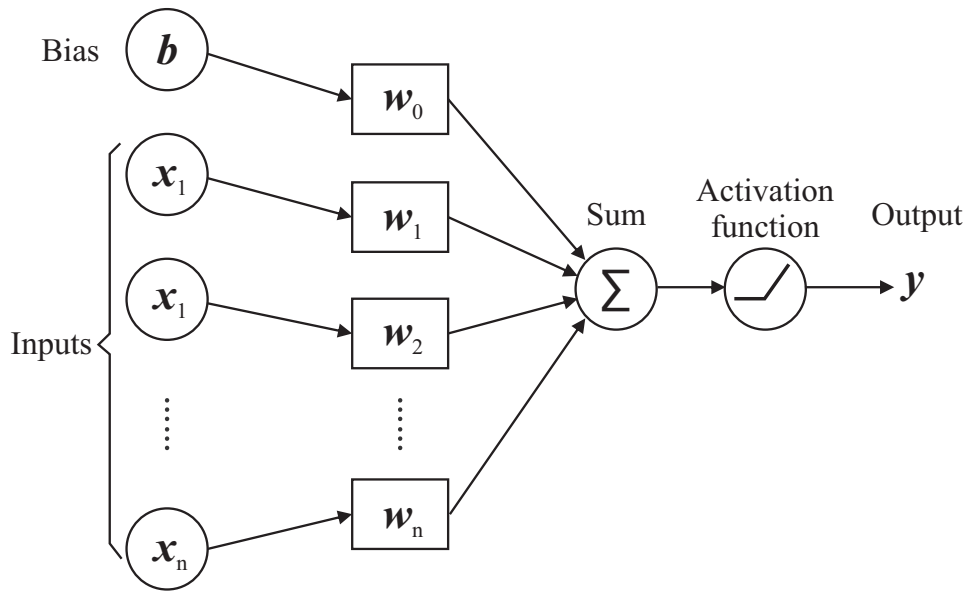


Figure A.15: Concept of perceptron.

where the vector $\hat{y}$ denotes the output vector, the function $\varphi(\cdot)$ represents the activation function. The input vector x is weighted by the weighting vector w . Additionally, a bias vector b is added to adjust the weighted input sum. During training, weights are adjusted using gradient descent to minimize the loss, which measures the error between predictions and target values.

In practice, multiple perceptrons are structured together to form a *multilayer perceptron* (MLP), a fundamental feedforward neural network as shown in Figure A.16 [257]. Each node, except those in the input layer, functions as a perceptron, applying weights and non-linear activation functions. MLPs comprise one or more hidden layers that process inputs and transform them into target outputs. Each layer contains multiple nodes, with every node fully connected to nodes in the preceding and succeeding layers, ensuring the outputs of one layer serve as inputs to the next.

Though MLPs are widely used in supervised learning tasks, they often struggle to achieve satisfactory prediction results as standalone models. Therefore, MLPs are typically embedded within larger deep learning architectures to transform data, extract significant features from raw or intermediate outputs, or organize intermediate information to produce classification or regression results effectively.

A.4.3.2 Gradient Descent and Backpropagation

Gradient descent is the most commonly used optimization algorithm in machine learning and deep learning for minimizing the loss function and finding the optimal trainable parameters

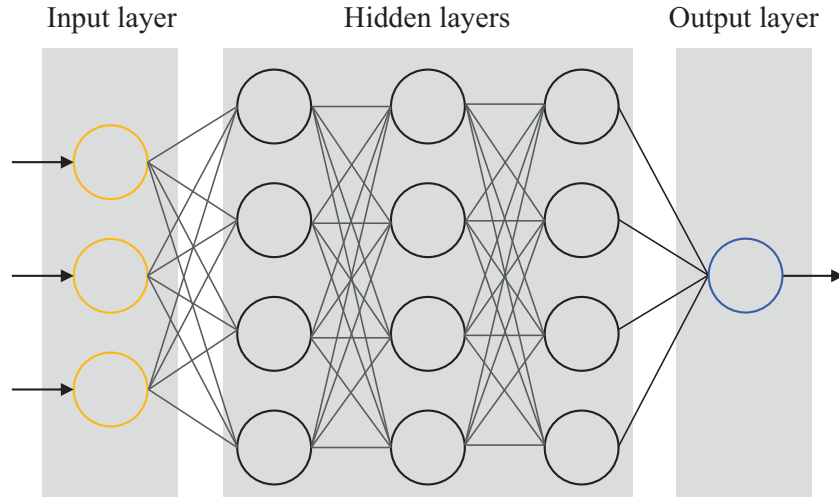


Figure A.16: Basic concept of MLP.

of a model. The core idea is to iteratively adjust the model parameters in the direction of the negative gradient of the loss function to those parameters [257].

Gradient descent can be mathematically described as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta}_t), \quad (\text{A.123})$$

where the vector $\boldsymbol{\theta}$ denotes the model parameters at successive timestamps t and $t + 1$. The variable α is the learning rate. The variable $\nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta}_t)$ represents the gradient of the objective function (loss function) concerning the model parameters. The to-be-minimized objective function is denoted as $\mathbf{J}(\boldsymbol{\theta}_t)$.

Several variants of gradient descent are available to establish the learning process, including *batch gradient descent*, *stochastic gradient descent*, and *mini-batch gradient descent*. Batch gradient descent computes the gradient over the entire training set, which is computationally expensive and unsuitable for large datasets. Stochastic gradient descent computes the gradient on a single training example, resulting in faster updates but a noisy convergence curve that may lead to sub-optimal solutions. Mini-batch gradient descent strikes a balance by computing the gradient on small batches of training examples, combining the efficiency of stochastic updates with the stability of batch processing. In this thesis, the mini-batch gradient descent method will be applied for its balanced advantages [259].

The learning rate is a critical hyperparameter in gradient descent, significantly affecting both the convergence speed and the final optimization result. Using an adaptive learning rate can help the model achieve better performance during training [260]. Therefore, in this thesis, the Adaptive Moment Estimation (Adam) optimizer, which incorporates an adaptive learning rate, will be employed for the gradient descent process.

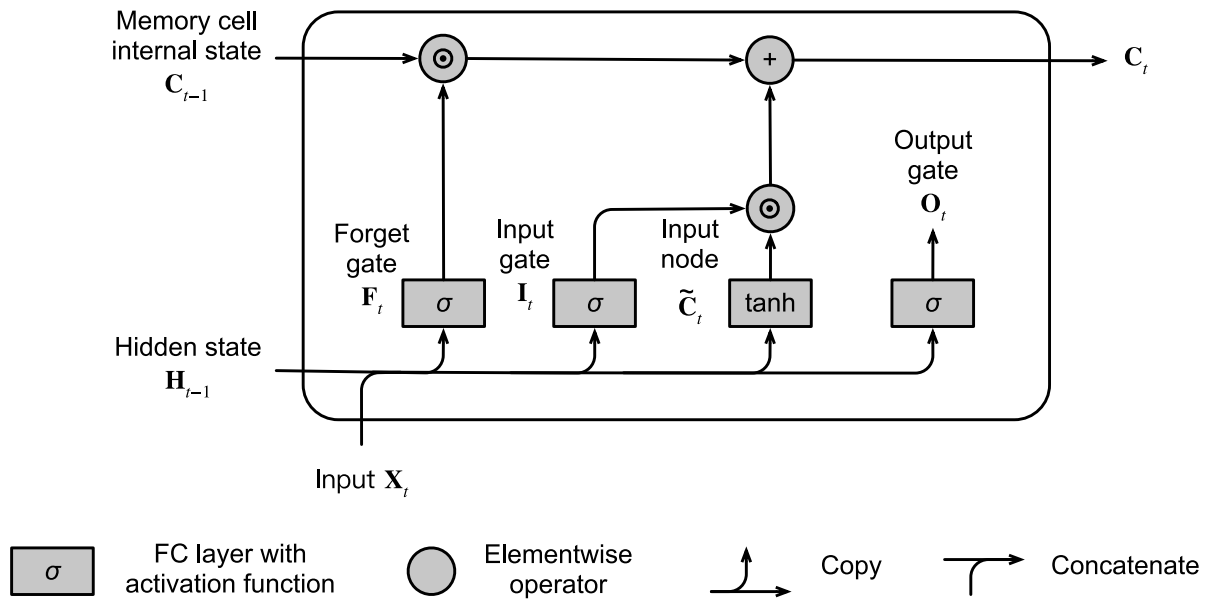


Figure A.17: LSTM Architecture.

To train a neural network using gradient descent, the gradients of the trainable parameters must be computed. The *backpropagation* algorithm [261], leveraging the chain rule of calculus, efficiently computes the gradient of the loss function with respect to the network's weights. Backpropagation propagates the model's loss backward through the network, layer by layer, calculating partial derivatives for all trainable parameters. These gradients are then used to optimize the parameters, ensuring the network improves its performance.

A.4.3.3 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) is a specialized type of recurrent neural network (RNN) designed for processing sequential data, such as time series. Unlike traditional feed-forward neural networks, RNNs share parameters across time steps and learn temporal dependencies. However, RNNs often face the vanishing gradient problem when handling long sequences. LSTMs address this issue with a unique architecture that includes a memory cell to store information over time and three gates—input, forget, and output gates—to regulate the flow of information.

Figure A.17 illustrates the basic architecture of LSTM. The input gate manages the information entering the memory cell from the input and previous hidden state. The forget gate selectively retains or discards previous memory cell state information. The output gate determines what information flows from the memory cell to the output. The computations

within the basic LSTM model can be presented by the following equations

$$\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_x^i + \mathbf{H}_{t-1} \mathbf{W}_h^i + \mathbf{b}^i), \\
\mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_x^f + \mathbf{H}_{t-1} \mathbf{W}_h^f + \mathbf{b}^f), \\
\mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_x^o + \mathbf{H}_{t-1} \mathbf{W}_h^o + \mathbf{b}^o), \\
\tilde{\mathbf{C}}_t &= \tanh(\mathbf{X}_t \mathbf{W}_x^c + \mathbf{H}_{t-1} \mathbf{W}_h^c + \mathbf{b}^c), \\
\mathbf{C}_t &= \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t, \\
\mathbf{H}_t &= \mathbf{O}_t \odot \tanh(\mathbf{C}_t),
\end{aligned} \tag{A.124}$$

where we denote the input, forget, and output gate at timestamp t as $\mathbf{W}_t$, $\mathbf{O}_t$, and $\mathbf{F}_t$, respectively. The hidden state is represented as $\mathbf{H}_t$. The cell state $\mathbf{C}_t$ serves as the memory of the network, while the candidate cell state $\tilde{\mathbf{C}}_t$ stores new information from the new input that is considered for inclusion in the cell state.

To calculate different gates and states (left side in (A.124)) in LSTM, the input and hidden states are multiplied by the associated weight matrices $\mathbf{W}$ and summed with the corresponding bias vectors $\mathbf{b}$ within the sigmoid activation function $\sigma(\cdot)$. Once the gates and states are computed, the memory cell state $\mathbf{C}_t$ and the hidden state $\mathbf{H}_t$ are updated accordingly, as shown in (A.124). The operator $\odot$ represents element-wise multiplication, while the hyperbolic tangent function $\tanh(\cdot)$ introduces non-linearity to calculate the hidden state.

By incorporating memory cells and gating mechanisms, LSTMs effectively capture long-term dependencies in sequential data, overcoming the limitations of traditional RNNs, such as the vanishing gradient problem [262]. In Chapter 5, I will adapt the LSTM network to address the GNSS faults prediction problem, as it is crucial to capture and utilize the temporal dependency relationships in satellite data. Given its strength in processing time series data, LSTM will be employed in this thesis to construct a robust and effective model.

A.4.3.4 Attention Mechanism

The attention mechanism was introduced to enhance neural networks' ability to handle long-range dependencies in sequential data, first popularized in machine translation by Vaswani et al. [186]. It operates as a weighted average of sequence elements, where the weights are dynamically computed based on the similarity between an input query and the elements' keys, allowing the model to effectively capture and prioritize relevant information.

The attention mechanism is usually constructed through four components:

- **Query $\mathbf{Q}$:** The query is a feature vector that defines what information is being sought in the sequence, determining which parts of the input should receive attention.

- **Key $\mathcal{K}$:** The key is a feature vector generated for each input element, describing the information produced by the element or what might be significant. Keys are designed to help identify the input elements relevant to the query.
- **Value $\mathcal{V}$:** For each input element, a value vector is generated, which is combined with the attention matrix to produce the attention-weighted output.
- **Score Function:** This function determines the importance of each query-key pair by computing attention scores, which form the attention matrix. The score function typically uses similarity metrics, such as scaled dot products, to measure alignment between queries and keys.

Using these components, we can formulate a *self-attention* module to compute the attention-weighted representation of the input sequence, where the query, key, and value vectors are all derived from the same input sequence. This enables the model to compute attention weights based on relationships between different input parts, effectively allowing the network to "pay attention" to key features within the data.

The attention mechanism described above can operate in parallel to produce sets of queries, keys, and values, facilitating the efficient computation of attention scores. This process is known as single-head attention. By applying multiple single-head attentions to the same input sequence, we can construct a *multi-head self-attention* module, which offers enhanced attention capabilities by attending to a wider range of significant information simultaneously, compared to using a single attention head (see Figure A.18).

The multi-head self-attention mechanism can be mathematically described as follows

$$\begin{aligned}\mathcal{Q} &= \mathbf{XW}^{\mathcal{Q}}, \\ \mathcal{K} &= \mathbf{XW}^{\mathcal{K}}, \\ \mathcal{V} &= \mathbf{XW}^{\mathcal{V}}, \\ \text{MH}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) &= \text{concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O,\end{aligned}\tag{A.125}$$

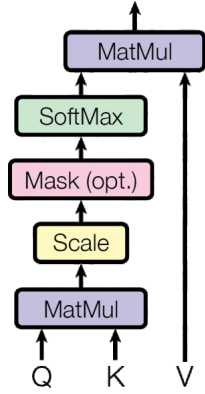
where the query, key, and value of the attention module are computed by projecting the feature map $\mathbf{X}$ with associated projection matrix $\mathbf{W}$. The multi-head attention is defined by the operator $\text{MH}(\dots)$, which concatenates all attention headers and multiplies by the output weighting matrix $\mathbf{W}^O$. The attention headers are calculated as

$$\text{head}_i = \text{Attention}(\mathcal{QW}_i^{\mathcal{Q}}, \mathcal{KW}_i^{\mathcal{K}}, \mathcal{VW}_i^{\mathcal{V}}), \quad i \in \{1, \dots, h\}.\tag{A.126}$$

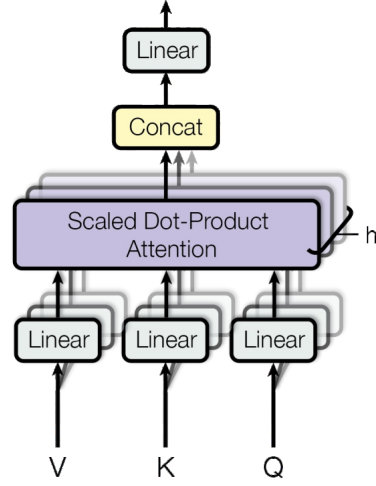
In contrast to single-head attention, multi-head attention projects the queries, keys, and values into h -times attention headers in (A.126) using the projection matrices $\mathbf{W}_i$, $i \in \{1, \dots, h\}$. Each non-projected attention header can be calculated using

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{\mathcal{QK}^T}{\sqrt{d_k}}\right) \mathcal{V},\tag{A.127}$$

Scaled Dot-Product Attention



Multi-Head Attention

**Figure A.18:** Multi-head attention architecture following Vaswani et al. [186].

where the dimension of the values is denoted as d_k .

The self-attention mechanism is widely used in tasks such as natural language processing and time-series analysis due to its ability to capture long-range dependencies and efficiently parallelize computations. While self-attention offers flexibility and scalability, it can be computationally expensive for large sequences and may require substantial memory resources, particularly in transformer models [263]. In this thesis, I propose a transformer-analogous attention mechanism to prioritize relevant environmental context information, supporting GNSS fault prediction in Chapter 5.

A.4.4 Performance Metrics of Learning-based Methods

Machine learning model evaluation metrics are quantitative measures used to assess and compare the performance of trained models. These metrics differ depending on the type of task being addressed:

- **Classification tasks** (e.g., spam detection, image recognition) commonly use metrics like accuracy, precision, recall, and F1 score to evaluate how well the model distinguishes between classes.
- **Regression tasks** (e.g., predicting house prices, stock market trends) rely on metrics like mean squared error (MSE), and mean absolute error (MAE) to measure the difference between predicted and actual values.

In the following sections, I present the performance metrics utilized in this thesis to evaluate and validate the effectiveness of the proposed learning-based methods.

A.4.4.1 Classification Metrics

The calculation of classification metrics is based on the following categories of predicted cases:

- True Positives (TP): The number of correctly predicted positive instances.
- True Negatives (TN): The number of correctly predicted negative instances.
- False Positives (FP): The number of instances incorrectly predicted as positive.
- False Negatives (FN): The number of instances incorrectly predicted as negative.

Accuracy

Accuracy describes the proportion of correctly classified instances out of the total number of instances. While it is simple and intuitive, accuracy can be misleading for imbalanced datasets, where one class significantly outweighs the other. The accuracy is calculated as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (\text{A.128})$$

Precision

Precision represents the proportion of true positives (correctly classified positive instances) out of all instances predicted as positive. Precision evaluates how often the model's positive predictions are correct, and it is computed as

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (\text{A.129})$$

Recall

Recall (aka sensitivity or true positive rate) measures the model's ability to correctly identify all positive instances in the dataset. It is the proportion of true positives out of all actual positive cases

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (\text{A.130})$$

F1-score

The F1-score is a performance metric that combines precision and recall into a single measure, providing a balanced evaluation of a model's accuracy. It is particularly useful when dealing with imbalanced datasets, where focusing solely on precision or recall can provide a skewed perspective. The F1-score is calculated as the harmonic mean of precision and recall, ensuring that both metrics are equally weighted

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (\text{A.131})$$

A.4.4.2 Regression metrics

In regression tasks, evaluation metrics are used to measure the difference between the predicted values and the actual values. These metrics assess how well a regression model captures the relationships in the data and predicts continuous outcomes. Below are the commonly used regression metrics:

Mean Absolute Error (MAE)

The Mean Absolute Error is the average of the absolute differences between the predicted and actual values:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|. \quad (\text{A.132})$$

Lower MAE values indicate better model performance. The MAE is suitable when all prediction errors are equally important.

Mean Squared Error (MSE)

The Mean Squared Error is the average of the squared differences between the predicted and actual values:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2. \quad (\text{A.133})$$

Smaller MSE values indicate a better fit. It is ideal for highlighting significant errors that require additional attention.

In this thesis, the Mean Absolute Error (MAE) is utilized as the regression metric to evaluate the predicted pseudorange error.

APPENDIX B

Useful Mathematical Expressions

B.1	Practical Transformations	203
B.1.1	Transformation from <i>ECEF</i> to <i>WGS84</i> in Closed Form	203
B.1.2	Transformation from <i>ECEF</i> to <i>WGS84</i> Iteratively	204
B.2	Supplementary Calculations for the GP Motion Priors	207
B.2.1	Derivative of the GP-WNOA Motion Prior	207
B.2.2	Derivative of the GP-WNOA Interpolation	208
B.2.3	Derivative of the GP-WNOJ Motion Prior	210
B.2.4	Derivative of the GP-WNOJ Interpolation	211
B.3	Factor Formulation	215
B.3.1	Pre-Integrated IMU Factor	215
B.3.2	IMU-Bias Factor	215
B.3.3	Between-Pose Factor	216
B.3.4	Velocity Factor	216
B.3.5	GNSS-PVT Factor	216
B.3.6	Pseudorange and Doppler-shift (PrDo) Factor	217
B.3.7	GNSS Receiver Clock Error Factor	217
B.4	Noise Models	217

B.1 Practical Transformations

This section presents useful coordinate transformations that supplement the fundamentals in Section A.3.1.

B.1.1 Transformation from *ECEF* to *WGS84* in Closed Form

To calculate the geodetic coordinates in *WGS84* from the associated global coordinates in the *ECEF* frame, Heikkinen [221] introduced a simplified, closed-form method. I exemplify this method in Algorithm 9 to calculate the geodetic coordinate of the satellite illustrated in Figure B.1.

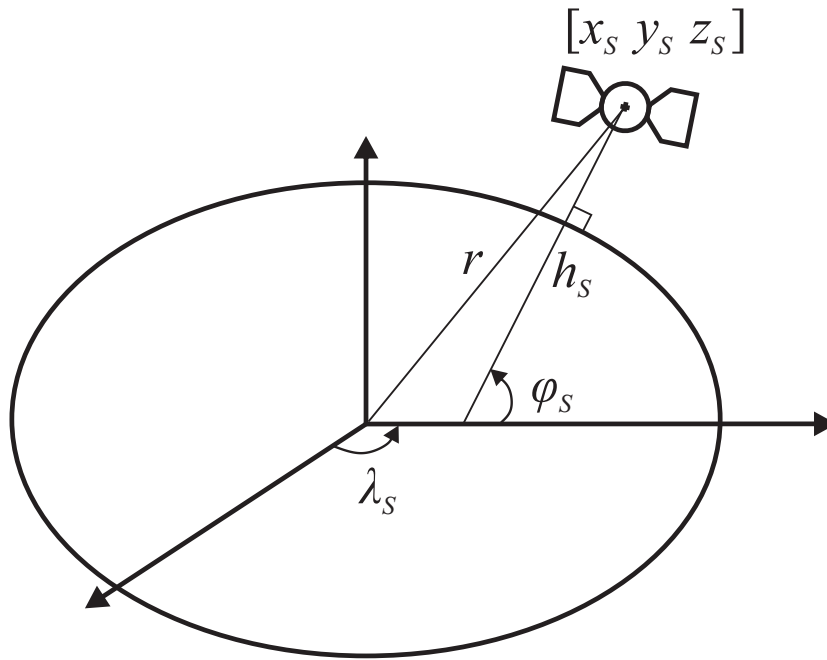


Figure B.1: Example of the transformation from *ECEF* to *WGS84* using Heikkinen [221].

B.1.2 Transformation from *ECEF* to *WGS84* Iteratively

In addition, I also provide the implementation in Algorithm 10 to transform the *ECEF*-coordinates into the geodetic frame using the numerical method, which is generally used in many applications [222].

Algorithm 9: Transformation from *ECEF* to *WGS84* in closed form.

Input : Initial Cartesian coordinate: $\mathbf{p}_S^{\text{ECEF}} = [x_S \ y_S \ z_S]^T$
Output: Geodetic coordinate $\mathbf{p}_S^{\text{LLH}} = [\varphi_S \ \lambda_S \ h_S]^T$

```

1  /* Calculate the relative  $e$  and numeric  $e'$  eccentricity using the
   semi-major axis  $a$  and semi-minor axis  $b$  of the ellipsoid */
2   $e^2 = \frac{a^2 - b^2}{a^2}, \quad e'^2 = \frac{a^2 - b^2}{b^2}, \quad E_2^{a^2 - b^2}$ 
3  /* Calculate some help variables: */
4   $r = \sqrt{x_S^2 + y_S^2}$ 
5   $F = 54 \cdot b^2 \cdot z_S^2$ 
6   $G = r^2 + (1 - e^2) \cdot z_S^2 - e^2 E^2$ 
7   $c = \frac{e^4 F r^2}{G^3}$ 
8   $s = \sqrt[3]{1 + c + \sqrt{c^2 + 2c}}$ 
9   $P = \frac{F}{3(s + \frac{1}{s} + 1)^2 G^2}$ 
10  $Q = \sqrt{1 + 2e^4 P}$ 
11  $r_0 = -\frac{P e^2 r}{1 + Q} + \sqrt{\frac{a^2}{2} (1 + \frac{1}{Q}) - \frac{P(1 - e^2) z_S^2}{Q(1 + Q)} - \frac{P r^2}{2}}$ 
12  $U = \sqrt{(r - e^2 r_0)^2 + z_S^2}$ 
13  $V = \sqrt{(r - e^2 r_0)^2 + (1 - e^2) z_S^2}$ 
14  $z_0 = \frac{b^2 z_S}{a V}$ 
15  $\sin \varphi_S = \frac{z_S + e'^2 z_0}{\sqrt{r^2 + (z_S + e'^2 z_0)^2}}, \quad \cos \varphi_S = \frac{r}{\sqrt{r^2 + (z_S + e'^2 z_0)^2}}$ 
16  $\sin \lambda_S = \frac{y_S}{r}, \quad \cos \lambda_S = \frac{x_S}{r}$ 
17 /* Calculate latitude  $\varphi_S$ , longitude  $\lambda_S$ , and height  $h_S$ : */
18  $\varphi_S = \arctan2\left(\frac{\sin \varphi_S}{\cos \varphi_S}\right) = \arctan2\left(\frac{z_S + e'^2 z_0}{r}\right)$ 
19  $\lambda_S = \arctan2\left(\frac{\sin \lambda_S}{\cos \lambda_S}\right) = \arctan2\left(\frac{y_S}{x_S}\right)$ 
20  $h_S = U \left(1 - \frac{b^2}{a V}\right)$ 
21 return  $\mathbf{p}_S^{\text{LLH}} = [\varphi_S \ \lambda_S \ h_S]^T$ 

```

Algorithm 10: Transformation from *ECEF* to *WGS84* using iterative method.

Input : Initial Cartesian coordinate: $\mathbf{p}_S^{\text{ECEF}} = [x_S \ y_S \ z_S]^T$
Output: Geodetic coordinate $\mathbf{p}_S^{\text{LLH}} = [\varphi_S \ \lambda_S \ h_S]^T$

- 1 **Initialization:**
- 2 $h_S = 0, \quad h'_S = 0, \quad N = a, \quad p = \sqrt{x_S^2 + y_S^2}$
- 3 **Perform the following iteration until $h_S - h'_S \leq 1e - 12$:**
- 4 $h'_S = h_S$
- 5 $\sin \lambda_S = \frac{z_S}{N(1-e^2)+h}$
- 6 $\lambda_S = \arctan2\left(\frac{z_S+e^2 \cdot N \sin \lambda_S}{p}\right)$
- 7 $N(\lambda) = \frac{a}{\sqrt{1-e^2 \sin^2 \lambda_S}}$
- 8 $h = \frac{p}{\cos \lambda_S} - N$
- 9 **return** $\mathbf{p}_S^{\text{LLH}} = [\varphi_S \ \lambda_S \ h_S]^T$

B.2 Supplementary Calculations for the GP Motion Priors

In this section, I provide mathematical formulations to calculate the Jacobian matrices of both GP motion priors and their interpolator introduced in Section 3.3.3 and Section 3.3.4. The Jacobian matrices are crucial in solving optimization problems using the Gauss-Newton method.

B.2.1 Derivative of the GP-WNOA Motion Prior

Recalling the error term of a GP-WNOA motion prior in (3.20), we get

$$\mathbf{e}_{ij}^{\text{wnoa}} = \begin{bmatrix} \ln(\mathbf{T}_{j,i})^\vee - (t_j - t_i)\boldsymbol{\varpi}_i \\ \mathcal{J}(\ln(\mathbf{T}_{j,i})^\vee)^{-1}\boldsymbol{\varpi}_j - \boldsymbol{\varpi}_i \end{bmatrix}, \quad t_i < t_\tau < t_j. \quad (\text{B.1})$$

To calculate the derivatives of the error term with respect to the state variable, only the term involving the inverse of the Jacobian, $\mathcal{J}(\ln(\mathbf{T}_{j,i})^\vee)^{-1}\boldsymbol{\varpi}_j$, requires further consideration.

Following the previous work Barfoot and Furgale [149, Appendix A], when the pose increment is small between the timestamps i and j , we can update the state $\boldsymbol{\xi}$ with the pose increment as a perturbation $\delta\boldsymbol{\xi}^\wedge$ as

$$\exp(\delta\boldsymbol{\xi}^\wedge) \exp(\boldsymbol{\xi}^\wedge) \approx \exp((\boldsymbol{\xi} + \delta\boldsymbol{\epsilon})), \text{ and} \quad (\text{B.2})$$

$$\delta\boldsymbol{\xi} = \mathcal{J}(\boldsymbol{\xi})\delta\boldsymbol{\epsilon}, \quad (\text{B.3})$$

where the vector $\delta\boldsymbol{\epsilon}$ denotes the linearized perturbation to each state dimension.

Given (B.2), we can equivalently represent a state increment in the global frame as

$$\mathbf{T}_{j,i} = \exp(\boldsymbol{\xi}_{j,i}^\wedge) \approx \exp(\delta\boldsymbol{\xi}_{j,i}^\wedge) \exp(\bar{\boldsymbol{\xi}}_{j,i}^\wedge) \approx \exp((\bar{\boldsymbol{\xi}}_{j,i} + \mathcal{J}(\bar{\boldsymbol{\xi}}_{j,i})^{-1}\delta\boldsymbol{\xi}_{j,i})^\wedge), \quad (\text{B.4})$$

where the state $\bar{\boldsymbol{\xi}}_{j,i}$ represents the mean value of the increment.

And thus, we can reformulate (B.4) into

$$\ln(\mathbf{T}_{j,i})^\vee \approx \ln(\bar{\mathbf{T}}_{j,i})^\vee + \mathcal{J}(\ln(\bar{\mathbf{T}}_{j,i})^\vee)^{-1}\delta\boldsymbol{\xi}_{j,i}, \quad (\text{B.5})$$

which simplifies the linearization of the inverse Jacobian in (B.1) as

$$\mathcal{J}(\ln(\mathbf{T}_{j,i})^\vee)^{-1}\boldsymbol{\varpi}_j \approx \mathcal{J}(\ln(\bar{\mathbf{T}}_{j,i})^\vee + \mathcal{J}(\ln(\bar{\mathbf{T}}_{j,i})^\vee)^{-1}\delta\boldsymbol{\xi}_{j,i})^{-1}(\bar{\boldsymbol{\varpi}}_j + \delta\boldsymbol{\varpi}_j), \quad (\text{B.6})$$

$$\approx \mathcal{J}_{j,i}^{-1}\bar{\boldsymbol{\varpi}}_j - \frac{1}{2}(\mathcal{J}_{j,i}^{-1}\delta\boldsymbol{\xi}_{j,i})^\wedge \bar{\boldsymbol{\varpi}}_j + \mathcal{J}_{j,i}^{-1}\delta\boldsymbol{\varpi}_j, \quad (\text{B.7})$$

$$= \mathcal{J}_{j,i}^{-1} + \frac{1}{2}\bar{\boldsymbol{\varpi}}_j^\wedge \mathcal{J}_{j,i}^{-1}\delta\boldsymbol{\xi}_{j,i} + \mathcal{J}_{j,i}^{-1}\delta\bar{\boldsymbol{\varpi}}_j. \quad (\text{B.8})$$

Now, we can calculate the derivatives of the error term in (B.1) with respect to the local state variable $\gamma = [\xi^T \varpi^T]^T$ as

$$\frac{\partial e_{ij}^{\text{wnoa}}}{\partial \gamma_i} = \begin{bmatrix} -\mathcal{J}_{j,i}^{-1} \mathcal{T}_{j,i} & -(t_j - t_i) \mathbf{1} \\ -\frac{1}{2} \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} \mathcal{T}_{j,i} & -\mathbf{1} \end{bmatrix} \quad \text{and} \quad \frac{\partial e_{ij}^{\text{wnoa}}}{\partial \gamma_j} = \begin{bmatrix} \mathcal{J}_{j,i}^{-1} & \mathbf{0} \\ \frac{1}{2} \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} & -\mathcal{J}_{j,i}^{-1} \end{bmatrix}. \quad (\text{B.9})$$

The matrix $\mathcal{T}$ denotes the *adjoint* transformation matrix [1, Chapter 8.1.4], which is defined as

$$\mathcal{T} := \text{Ad}(\mathbf{T}) = \text{Ad} \left(\begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{R} & \mathbf{p}^\wedge \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} = \exp(\xi^\wedge), \quad (\text{B.10})$$

where the operator $\xi^\wedge$ represents the adjoint of $\xi^\wedge \in \mathfrak{se}(3)$, which can be calculated as

$$\xi^\wedge = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho^\wedge \\ \mathbf{0} & \phi^\wedge \end{bmatrix}. \quad (\text{B.11})$$

B.2.2 Derivative of the GP-WNOA Interpolation

Recalling the state interpolation using the GP-WNOA motion prior in (3.19), we get the interpolation of the pose and velocity as

$$\mathbf{T}_\tau^{\text{wnoa}} = \exp \left\{ \left[\Lambda_{12}(\tau) \varpi_i + \Omega_{11}(\tau) \ln(\mathbf{T}_{j,i})^\vee + \Omega_{12}(\tau) \mathcal{J}_j^{-1} \varpi_j \right]^\wedge \right\} \mathbf{T}_i, \quad (\text{B.12})$$

$$\varpi_\tau^{\text{wnoa}} = \mathcal{J}(\ln(\mathbf{T}(\tau) \mathbf{T}_i^{-1})^\vee) (\Lambda_{22}(\tau) \varpi_i + \Omega_{21} \delta \xi_{j,i} + \Omega_{22}(\tau) \mathcal{J}_{j,i}^{-1} \varpi_j), \quad t_i < t_\tau < t_j, \quad (\text{B.13})$$

where $\Lambda(\tau) \in \mathbb{R}^{12 \times 12}$ and $\Omega(\tau) \in \mathbb{R}^{12 \times 12}$ are vehicle transition matrices obtained from (3.8) and (3.9). For the sake of clarity in the calculations, we cluster the transition matrices $\Lambda(\tau)$ and $\Omega(\tau)$ into

$$\Lambda(\tau) = \begin{bmatrix} \overbrace{\Lambda_{11}(\tau)}^{\xi_i} & \overbrace{\Lambda_{12}(\tau)}^{\varpi_i} \\ \overbrace{\Lambda_{21}(\tau)}^{\xi_j} & \overbrace{\Lambda_{22}(\tau)}^{\varpi_j} \end{bmatrix}, \quad \Omega(\tau) = \begin{bmatrix} \overbrace{\Omega_{11}(\tau)}^{\xi_i} & \overbrace{\Omega_{12}(\tau)}^{\varpi_i} \\ \overbrace{\Omega_{21}(\tau)}^{\xi_j} & \overbrace{\Omega_{22}(\tau)}^{\varpi_j} \end{bmatrix}, \quad (\text{B.14})$$

where the subblocks $\Lambda_{mn}(\tau) \in \mathbb{R}^{6 \times 6}$ and $\Omega_{mn}(\tau) \in \mathbb{R}^{6 \times 6}$. The Jacobian of the term $\ln(\mathbf{T}(\tau) \mathbf{T}_i^{-1})^\vee$ can be computed for the velocity interpolation as

$$\mathcal{J}(\ln(\mathbf{T}(\tau) \mathbf{T}_i^{-1})^\vee) = \mathcal{J}(\Lambda_{12} \varpi_i + \Omega_{11} \delta \xi_{j,i} + \Omega_{1,2} \mathcal{J}_{j,i}^{-1} \varpi_j). \quad (\text{B.15})$$

To calculate the derivative of the interpolated state $\mathbf{T}_\tau^{\text{wnoa}}$, we perform the linearization of (B.12) by introducing an intermediate variable

$$\boldsymbol{\theta}_i^{\text{wnoa}}(\tau) := \{\mathbf{T}_{\tau,i}, \boldsymbol{\varpi}_i(\tau)\} = \{\mathbf{T}_i(\tau)\mathbf{T}_i^{-1}, \boldsymbol{\varpi}_i(\tau)\}, \quad t_i < t_\tau, \quad \delta\boldsymbol{\theta}_i^{\text{wnoa}}(\tau) = \begin{bmatrix} \delta\boldsymbol{\xi}_{\tau,i} \\ \delta\boldsymbol{\varpi}_i(\tau) \end{bmatrix}, \quad (\text{B.16})$$

which projects the interpolated state $\mathbf{T}_\tau$ into a local frame fixed at $\mathbf{T}_i$ [264]. This projection enables the following linearization of a local state variable

$$\boldsymbol{\gamma}_i(\tau) = \begin{bmatrix} \boldsymbol{\xi}_i(\tau) \\ \boldsymbol{\varpi}_i(\tau) \end{bmatrix} \approx \begin{bmatrix} \ln(\bar{\mathbf{T}}_i(\tau)\bar{\mathbf{T}}_i)^\wedge \\ \mathcal{J}_i(\tau)^{-1}\bar{\boldsymbol{\varpi}}_i(\tau) \end{bmatrix} + \begin{bmatrix} \mathcal{J}_i(\tau)^{-1} & \mathbf{0} \\ \frac{1}{2}\bar{\boldsymbol{\varpi}}_i(\tau)^\wedge \mathcal{J}_i(\tau)^{-1} & \mathcal{J}_i(\tau)^{-1} \end{bmatrix} \delta\boldsymbol{\theta}_i(\tau). \quad (\text{B.17})$$

Thus, the derivative of the interpolated state $\mathbf{T}_\tau^{\text{wnoa}}$ can be calculated using the perturbation state $\delta\boldsymbol{\theta}_i^{\text{wnoa}}$ of the intermediate variable $\boldsymbol{\theta}_i^{\text{wnoa}}$ by utilizing the product rule and chain rule, which gives us

$$\frac{\partial \boldsymbol{\xi}_\tau}{\partial \delta \boldsymbol{\xi}_i} \approx \frac{\delta \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_i} + \boldsymbol{\tau}_{\tau,i} \frac{\partial \boldsymbol{\xi}_i}{\partial \delta \boldsymbol{\xi}_i} = \frac{\delta \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_i} + \boldsymbol{\tau}_{\tau,i} = -\frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}} \boldsymbol{\tau}_{j,i} + \boldsymbol{\tau}_{\tau,i}, \quad (\text{B.18})$$

$$\frac{\partial \boldsymbol{\xi}_\tau}{\partial \delta \boldsymbol{\varpi}_i} = \boldsymbol{\Lambda}_{12}(\tau) \boldsymbol{\mathcal{J}}_{\tau,i}, \quad (\text{B.19})$$

$$\frac{\partial \boldsymbol{\xi}_\tau}{\partial \delta \boldsymbol{\xi}_j} \approx \frac{\delta \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_j} + \boldsymbol{\tau}_{\tau,i} \frac{\delta \boldsymbol{\xi}_i}{\partial \delta \boldsymbol{\xi}_j} = \frac{\delta \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_j} = \frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}}, \quad (\text{B.20})$$

$$\frac{\partial \boldsymbol{\xi}_\tau}{\partial \delta \boldsymbol{\varpi}_j} = \boldsymbol{\Omega}_{12}(\tau) \boldsymbol{\mathcal{J}}_{\tau,i} \boldsymbol{\mathcal{J}}_{j,i}^{-1}, \quad (\text{B.21})$$

where the partial derivative $\frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}}$ is calculated as

$$\frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}} = \boldsymbol{\Omega}_{11}(\tau) \boldsymbol{\mathcal{J}}_{\tau,i} \boldsymbol{\mathcal{J}}_{j,i}^{-1} + \frac{1}{2} \boldsymbol{\Omega}_{12} \boldsymbol{\mathcal{J}}_{\tau,i} \bar{\boldsymbol{\varpi}}_j^\wedge \boldsymbol{\mathcal{J}}_{j,i}^{-1}. \quad (\text{B.22})$$

Collecting the equations above together, we get the derivative of the interpolated pose $\mathbf{T}_\tau^{\text{wnoa}} = \mathbf{T}_{\tau,i} \mathbf{T}_i = \exp((\boldsymbol{\gamma}_i(\tau))^\wedge) \mathbf{T}_i$ with the respect to the local state $\boldsymbol{\gamma} = [\boldsymbol{\xi}^T \boldsymbol{\varpi}^T]^T$:

$$\frac{\partial \mathbf{T}_\tau^{\text{wnoa}}}{\partial \boldsymbol{\gamma}_i} = \mathbf{J}_{\exp(\boldsymbol{\gamma}_i(\tau))\mathbf{T}_i} \begin{bmatrix} -\frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}} \boldsymbol{\tau}_{j,i} + \boldsymbol{\tau}_{\tau,i} \\ \boldsymbol{\Lambda}_{12}(\tau) \boldsymbol{\mathcal{J}}_{\tau,i} \end{bmatrix}, \quad \text{and} \quad (\text{B.23})$$

$$\frac{\partial \mathbf{T}_\tau^{\text{wnoa}}}{\partial \boldsymbol{\gamma}_j} = \mathbf{J}_{\exp(\boldsymbol{\gamma}_i(\tau))\mathbf{T}_i} \begin{bmatrix} \frac{\partial \boldsymbol{\xi}_{\tau,i}}{\partial \delta \boldsymbol{\xi}_{j,i}} \\ \boldsymbol{\Omega}_{12}(\tau) \boldsymbol{\mathcal{J}}_{\tau,i} \boldsymbol{\mathcal{J}}_{j,i}^{-1} \end{bmatrix}. \quad (\text{B.24})$$

where the Jacobian $\mathbf{J}_{\exp(\gamma_i(\tau))\mathbf{T}_i}$ presents the derivative of the exponential map $\exp((\gamma_i(\tau))^\wedge)$, which can be found in Barfoot [1, Chapter 8].

Since the interpolated velocity in (B.13) is in the local frame, we do not need to retract it and compose to the global frame. Thus, the derivative of the interpolated velocity $\varpi_\tau^{\text{wnoa}}$ with the respect to the local state $\gamma = [\xi^T \varpi^T]^T$ is computed as

$$\frac{\partial \varpi_\tau^{\text{wnoa}}}{\partial \gamma_i} = \mathcal{J}(\ln(\mathbf{T}(\tau)\mathbf{T}_i^{-1})^\vee) \begin{bmatrix} -\Omega_{21} \mathcal{J}_{j,i}^{-1} \mathcal{T}_{j,i} - \frac{1}{2} \Omega_{22} \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} \mathcal{T}_{j,i} \\ \Lambda_{22} \end{bmatrix}, \quad \text{and} \quad (\text{B.25})$$

$$\frac{\partial \varpi_\tau^{\text{wnoa}}}{\partial \gamma_j} = \mathcal{J}(\ln(\mathbf{T}(\tau)\mathbf{T}_i^{-1})^\vee) \begin{bmatrix} \Omega_{21} \mathcal{J}_{j,i}^{-1} + \frac{1}{2} \Omega_{22} \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} \\ \Omega_{21} \mathcal{J}_{j,i}^{-1} \end{bmatrix}. \quad (\text{B.26})$$

B.2.3 Derivative of the GP-WNOJ Motion Prior

Similar to the GP-WNOA motion prior, the GP-WNOJ prior involves the acceleration into the dynamic model. We write

$$\mathbf{e}_{ij}^{\text{wnoj}} = \begin{bmatrix} \ln(\mathbf{T}_{j,i})^\vee - (t_j - t_i) \varpi_i - \frac{1}{2} (t_j - t_i)^2 \dot{\varpi}_i, \\ \mathcal{J}_{j,i}^{-1} \varpi_j - \varpi_i - (t_j - t_i) \dot{\varpi}_i, \\ -\underbrace{\frac{1}{2} (\mathcal{J}_{j,i}^{-1} \varpi_j)^\wedge \varpi_j}_{\mathbf{o}_1} + \underbrace{\mathcal{J}_{j,i}^{-1} \dot{\varpi}_j - \dot{\varpi}_i}_{\mathbf{o}_2} \end{bmatrix}, \quad (\text{B.27})$$

where only the third row, namely the error terms $\mathbf{o}_1$ and $\mathbf{o}_2$ with the respect to the local accelerations, requires our attention.

Following Tang et al. [103], we consider the following linearization

$$\mathbf{o}_1 = -\frac{1}{2} (\mathcal{J}_{j,i}^{-1} \varpi_j)^\wedge \varpi_j \quad (\text{B.28})$$

$$\approx -\frac{1}{2} (\mathcal{J}(\xi_{j,i} + \mathcal{J}_{j,i}^{-1} \delta \xi_{j,i})^{-1} (\varpi_j + \delta \varpi_j))^\wedge (\varpi_j + \delta \varpi_j). \quad (\text{B.29})$$

Initialized by Anderson [264], we get

$$\mathcal{J}(\xi_{j,i} + \mathcal{J}_{j,i}^{-1} \delta \xi_{j,i})^{-1} (\varpi_j + \delta \varpi_j) \approx \mathcal{J}_{j,i}^{-1} \varpi_j + \frac{1}{2} \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} \delta \xi_{j,i} + \mathcal{J}_{j,i}^{-1} \delta \varpi_j. \quad (\text{B.30})$$

Inserting (B.30) in (B.28), we write

$$\begin{aligned}
 o_1 &\approx -\frac{1}{2}(\mathcal{J}(\xi_{j,i} + \mathcal{J}_{j,i}^{-1}\delta\xi_{j,i})^{-1}(\varpi_j + \delta\varpi_j))^\wedge(\varpi_j + \delta\varpi_j) \\
 &= -\frac{1}{2}\left((\mathcal{J}_{j,i}\varpi_j)^\wedge + \frac{1}{2}(\varpi_j^\wedge \mathcal{J}_{j,i}^{-1}\delta\xi_{j,i})^\wedge + (\mathcal{J}_{j,i}^{-1}\delta\varpi_j)^\wedge\right)(\varpi_j + \delta\varpi_j) \\
 &= -\frac{1}{2}\left((\mathcal{J}_{j,i}\varpi_j)^\wedge - \frac{1}{2}(\mathcal{J}_{j,i}^{-1}\delta\xi_{j,i})^\wedge + (\mathcal{J}_{j,i}^{-1}\delta\varpi_j)^\wedge\right)(\varpi_j + \delta\varpi_j) \\
 &\approx -\frac{1}{2}\left((\mathcal{J}_{j,i}^{-1}\varpi_j)^\wedge + (\mathcal{J}_{j,i}^{-1}\varpi_j)^\wedge\delta\varpi_j - \varpi_j^\wedge \mathcal{J}_{j,i}^{-1}\delta\varpi_j - \frac{1}{2}\varpi_j^\wedge \varpi_j^\wedge \mathcal{J}_{j,i}^{-1}\delta\xi_{j,i}\right).
 \end{aligned} \tag{B.31}$$

For the error term o_2 , we again utilize (B.30) and compute

$$\mathcal{J}_{j,i}^{-1}\dot{\varpi}_j \approx \mathcal{J}_{j,i}^{-1}\dot{\varpi}_j + \frac{1}{2}\dot{\varpi}_j^\wedge \mathcal{J}_{j,i}^{-1}\delta\xi_{j,i} + \mathcal{J}_{j,i}^{-1}\delta\dot{\varpi}_j. \tag{B.32}$$

In summary, we can calculate the derivatives of the error term in (B.27) with respect to the local state variable $\gamma = [\xi^T \varpi^T \dot{\varpi}^T]^T$ as

$$\frac{\partial e_{ij}^{\text{wnoj}}}{\partial \gamma_i} = \begin{bmatrix} -\mathcal{J}_{j,i}^{-1}\mathcal{T}_{j,i} & -(t_j - t_i)\mathbf{1} & -\frac{1}{2}((t_j - t_i))^2\mathbf{1} \\ -\frac{1}{2}\varpi_j^\wedge \mathcal{J}_{j,i}^{-1}\mathcal{T}_{j,i} & -\mathbf{1} & -(t_j - t_i)\mathbf{1} \\ -\frac{1}{4}\varpi_j^\wedge \varpi_j^\wedge \mathcal{J}_{j,i}^{-1}\mathcal{T}_{j,i} - \frac{1}{2}\dot{\varpi}_j^\wedge \mathcal{J}_{j,i}^{-1}\mathcal{T}_{j,i} & \mathbf{0} & -\mathbf{1} \end{bmatrix}, \tag{B.33}$$

$$\frac{\partial e_{ij}^{\text{wnoj}}}{\partial \gamma_j} = \begin{bmatrix} \mathcal{J}_{j,i}^{-1} & \mathbf{0} & \mathbf{0} \\ \frac{1}{2}\varpi_j^\wedge \mathcal{J}_{j,i}^{-1} & \mathcal{J}_{j,i}^{-1} & \mathbf{0} \\ \frac{1}{4}\varpi_j^\wedge \varpi_j^\wedge \mathcal{J}_{j,i}^{-1} + \frac{1}{2}\dot{\varpi}_j^\wedge \mathcal{J}_{j,i}^{-1} & -\frac{1}{2}(\mathcal{J}_{j,i}^{-1}\varpi_j)^\wedge + \frac{1}{2}\varpi_j^\wedge \mathcal{J}_{j,i}^{-1} & \mathcal{J}_{j,i}^{-1} \end{bmatrix}. \tag{B.34}$$

B.2.4 Derivative of the GP-WNOJ Interpolation

Recalling the interpolation using the GP-WNOJ motion prior, we write¹

$$\begin{aligned}
 \mathbf{T}_\tau^{\text{wnoj}} &= \exp \left\{ \left[\Lambda_{12}(\tau)\varpi_i + \Omega_{13}(\tau)\dot{\varpi}_i + \Sigma_{11}(\tau)\ln(\mathbf{T}_{i+1,i})^\vee + \Omega_{12}(\tau)\mathcal{J}_{i+1}^{-1}\varpi_{i+1} + \right. \right. \\
 &\quad \left. \left. + \Omega_{13}(\tau)\left(-\frac{1}{2}(\mathcal{J}_{i+1}^{-1}\varpi_{i+1})^\wedge\varpi_{i+1} + \mathcal{J}_{i+1}^{-1}\dot{\varpi}_{i+1}\right) \right]^\wedge \right\} \mathbf{T}_i,
 \end{aligned} \tag{B.35}$$

$$\begin{aligned}
 \varpi_\tau^{\text{wnoj}} &= \mathcal{J}(\ln(\mathbf{T}(\tau)\mathbf{T}_i^{-1})^\vee)(\Lambda_{22}(\tau)\varpi_i + \Lambda_{23}(\tau)\dot{\varpi}_i + \Omega_{21}(\tau)\delta\xi_{j,i} \\
 &\quad + \Omega_{22}(\tau)\mathcal{J}_{j,i}^{-1}\varpi_j + \Omega_{23}(\tau)\left(-\frac{1}{2}(\mathcal{J}_{j,i}^{-1}\varpi_j)^\wedge\varpi_j + \mathcal{J}_{j,i}^{-1}\dot{\varpi}_j\right),
 \end{aligned} \tag{B.36}$$

$$\dot{\varpi}_\tau^{\text{wnoj}} = \mathcal{J}_{\tau,i}(\dot{\varpi}_{\tau,k} + \frac{1}{2}\dot{\varpi}_{\tau,j}^\wedge \mathcal{J}_{\tau,i}\dot{\varpi}_{\tau,j}), \tag{B.37}$$

¹The original resource of the expressions can be found at <https://github.com/utiasASRL/steam>.

where we shorten the Jacobian $\mathcal{J}(\ln(\mathbf{T}(\tau)\mathbf{T}_i^{-1})^\vee)$ as $\mathcal{J}_{\tau,i}$ for the sake of clarity. The transition matrices $\Lambda(\tau) \in \mathbb{R}^{18 \times 18}$ and $\Omega(\tau) \in \mathbb{R}^{18 \times 18}$ for the GP-WNOJ prior are reshaped as

$$\Lambda(\tau) = \begin{bmatrix} \overbrace{\Lambda_{11}(\tau)}^{\xi_i} & \overbrace{\Lambda_{12}(\tau)}^{\varpi_i} & \overbrace{\Lambda_{13}(\tau)}^{\dot{\varpi}_i} \\ \overbrace{\Lambda_{21}(\tau)}^{\xi_j} & \overbrace{\Lambda_{22}(\tau)}^{\varpi_j} & \overbrace{\Lambda_{23}(\tau)}^{\dot{\varpi}_j} \\ \overbrace{\Lambda_{31}(\tau)}^{\xi_k} & \overbrace{\Lambda_{32}(\tau)}^{\varpi_k} & \overbrace{\Lambda_{33}(\tau)}^{\dot{\varpi}_k} \end{bmatrix}, \quad \Omega(\tau) = \begin{bmatrix} \overbrace{\Omega_{11}(\tau)}^{\xi_i} & \overbrace{\Omega_{12}(\tau)}^{\varpi_i} & \overbrace{\Omega_{13}(\tau)}^{\dot{\varpi}_i} \\ \overbrace{\Omega_{21}(\tau)}^{\xi_j} & \overbrace{\Omega_{22}(\tau)}^{\varpi_j} & \overbrace{\Omega_{23}(\tau)}^{\dot{\varpi}_j} \\ \overbrace{\Omega_{31}(\tau)}^{\xi_k} & \overbrace{\Omega_{32}(\tau)}^{\varpi_k} & \overbrace{\Omega_{33}(\tau)}^{\dot{\varpi}_k} \end{bmatrix}. \quad (\text{B.38})$$

The intermediate variables in (B.37) are given as

$$\begin{aligned} \dot{\varpi}_{\tau,i} &= \Lambda_{12}(\tau)\varpi_i + \Lambda_{13}(\tau)\dot{\varpi}_i + \Omega_{11}(\tau)\delta\xi_{j,i} + \Omega_{12}(\tau)\mathcal{J}_{j,i}^{-1}\varpi_j \\ &\quad + \Omega_{13}(\tau)\left(-\frac{1}{2}(\mathcal{J}_{i+1}^{-1}\varpi_{i+1})^\lambda + \mathcal{J}_{i+1}^{-1}\dot{\varpi}_j\right), \\ \dot{\varpi}_{\tau,j} &= \Lambda_{22}(\tau)\varpi_i + \Lambda_{23}(\tau)\dot{\varpi}_i + \Omega_{21}(\tau)\delta\xi_{j,i} + \Omega_{22}(\tau)\mathcal{J}_{j,i}^{-1}\varpi_j \\ &\quad + \Omega_{23}(\tau)\left(-\frac{1}{2}(\mathcal{J}_{i+1}^{-1}\varpi_{i+1})^\lambda + \mathcal{J}_{i+1}^{-1}\dot{\varpi}_j\right), \\ \dot{\varpi}_{\tau,k} &= \Lambda_{32}(\tau)\varpi_i + \Lambda_{33}(\tau)\dot{\varpi}_i + \Omega_{31}(\tau)\delta\xi_{j,i} + \Omega_{32}(\tau)\mathcal{J}_{j,i}^{-1}\varpi_j \\ &\quad + \Omega_{33}(\tau)\left(-\frac{1}{2}(\mathcal{J}_{i+1}^{-1}\varpi_{i+1})^\lambda + \mathcal{J}_{i+1}^{-1}\dot{\varpi}_j\right). \end{aligned}$$

Similar to the GP-WNOA interpolator presented in Section B.2.2, the derivative of the interpolated state $\mathbf{T}_\tau^{\text{wnoj}}$ can be calculated using the perturbation state $\delta\theta_i^{\text{wnoj}}$ of the intermediate variable θ_i^{wnoj} , where

$$\theta_i^{\text{wnoj}}(\tau) := \{\mathbf{T}_{\tau,i}, \varpi_i(\tau), \dot{\varpi}_i(\tau)\} = \{\mathbf{T}_i(\tau)\mathbf{T}_i^{-1}, \varpi_i(\tau), \dot{\varpi}_i(\tau)\}, \quad t_i < t_\tau, \quad (\text{B.39})$$

$$\delta\theta_i^{\text{wnoj}}(\tau) = \begin{bmatrix} \delta\xi_{\tau,i} \\ \delta\varpi_i(\tau) \\ \delta\dot{\varpi}_i(\tau) \end{bmatrix}. \quad (\text{B.40})$$

By utilizing the product rule and chain rule, we get the following extension

$$\frac{\partial \xi_\tau}{\partial \delta \dot{\varpi}_i} = \Lambda_{13}(\tau)\mathcal{J}_{\tau,i}, \quad (\text{B.41})$$

$$\frac{\partial \xi_\tau}{\partial \delta \varpi_j} = \Omega_{12}(\tau)\mathcal{J}_{\tau,i}\mathcal{J}_{j,i}^{-1} - \frac{1}{2}\Omega_{13}(\tau)\mathcal{J}_{\tau,i}((\mathcal{J}_{j,i}^{-1}\varpi_j)^\lambda - \varpi_j^\lambda \mathcal{J}_{j,i}^{-1}), \quad (\text{B.42})$$

$$\frac{\partial \xi_\tau}{\partial \delta \dot{\varpi}_j} = \Omega_{13}(\tau)\mathcal{J}_{\tau,i}\mathcal{J}_{j,i}^{-1}, \quad (\text{B.43})$$

where the partial derivative $\frac{\partial \xi_{\tau,i}}{\partial \delta \xi_{j,i}}$ is calculated as

$$\frac{\partial \xi_{\tau,i}}{\partial \delta \xi_{j,i}} = \Omega_{11}(\tau) \mathcal{J}_{\tau,i} \mathcal{J}_{j,i}^{-1} + \frac{1}{2} \Omega_{12}(\tau) \mathcal{J}_{\tau,i} \bar{\omega}_j^\wedge \mathcal{J}_{j,i}^{-1} + \frac{1}{4} \Omega_{13}(\tau) \mathcal{J}_{\tau,i} \bar{\omega}_j^\wedge \bar{\omega}_j^\wedge \mathcal{J}_{j,i}^{-1} \quad (\text{B.44})$$

$$+ \frac{1}{2} \Omega_{13}(\tau) \bar{\omega}_j^\wedge \mathcal{J}_{j,i}^{-1}. \quad (\text{B.45})$$

Collecting the equations above together, we get the derivative of the interpolated pose $\mathbf{T}_\tau^{\text{wnoj}} = \mathbf{T}_{\tau,i} \mathbf{T}_i = \exp((\gamma_i(\tau))^\wedge) \mathbf{T}_i$ with the respect to the local state $\gamma = [\xi^T \ \varpi^T \ \dot{\varpi}^T]^T$:

$$\frac{\partial \mathbf{T}_\tau^{\text{wnoj}}}{\partial \gamma_i} = \mathbf{J}_{\exp(\gamma_i(\tau)) \mathbf{T}_i} \begin{bmatrix} -\frac{\partial \xi_{\tau,i}}{\partial \delta \xi_{j,i}} \mathcal{T}_{j,i} + \mathcal{T}_{\tau,i} \\ \Lambda_{12}(\tau) \mathcal{J}_{\tau,i} \\ \Lambda_{13}(\tau) \mathcal{J}_{\tau,i} \end{bmatrix}, \quad \text{and} \quad (\text{B.46})$$

$$\frac{\partial \mathbf{T}_\tau^{\text{wnoj}}}{\partial \gamma_j} = \mathbf{J}_{\exp(\gamma_i(\tau)) \mathbf{T}_i} \begin{bmatrix} \frac{\partial \xi_{\tau,i}}{\partial \delta \xi_{j,i}} \\ \Omega_{12}(\tau) \mathcal{J}_{\tau,i} \mathcal{J}_{j,i}^{-1} - \frac{1}{2} \Omega_{13}(\tau) \mathcal{J}_{\tau,i} ((\mathcal{J}_{j,i}^{-1} \bar{\omega}_j)^\wedge - \bar{\omega}_j^\wedge \mathcal{J}_{j,i}^{-1}) \\ \Omega_{13}(\tau) \mathcal{J}_{\tau,i} \mathcal{J}_{j,i}^{-1} \end{bmatrix}. \quad (\text{B.47})$$

where the Jacobian $\mathbf{J}_{\exp(\gamma_i(\tau)) \mathbf{T}_i}$ presents the derivative of the exponential map $\exp((\gamma_i(\tau))^\wedge)$, which can be found in Barfoot [1, Chapter 8].

Although the interpolation of the velocity in (B.36) presents a complex calculation, it is basically driven from

$$\varpi(\tau) = \mathcal{J}(\xi_\tau) \dot{\xi}_\tau. \quad (\text{B.48})$$

To calculate the derivative of the interpolated velocity $\varpi_\tau^{\text{wnoj}}$ with the respect to the local state $\gamma = [\xi^T \ \varpi^T \ \dot{\varpi}^T]^T$, we firstly take this basic formulation, and compute its derivative with respect to an arbitrary local pose² ξ_t , $t \neq \tau$ using the product and chain rule

$$\frac{\partial \varpi}{\partial \xi_t} = \mathcal{J}_{\tau,t} \frac{\partial \dot{\xi}_t}{\partial \xi_t} - \frac{1}{2} \dot{\xi}_t^\wedge \frac{\partial \xi_\tau}{\partial \xi_t}. \quad (\text{B.49})$$

Analogously, the interpolated acceleration in (B.37) takes the basic formulation

$$\dot{\varpi}(\tau) \approx \mathcal{J}(\xi_\tau) (\ddot{\xi}_\tau + \frac{1}{2} \dot{\xi}_\tau^\wedge \varpi(\tau)). \quad (\text{B.50})$$

²Only the derivative with respect to the local pose needs to be considered here due to its nonlinearity.

By leveraging the product and chain rule, the derivative of (B.50) with respect to an arbitrary local pose ξ_t , $t \neq \tau$ is given as

$$\begin{aligned}
 \frac{\partial \dot{\varpi}(\tau)}{\partial \xi_t} &= \mathcal{J}_{\tau,t} \frac{\partial}{\partial \xi_t} \left(\ddot{\xi}_\tau + \frac{1}{2} \dot{\xi}_\tau^\lambda \varpi(\tau) \right) - \frac{1}{2} \left(\ddot{\xi}_\tau + \frac{1}{2} \dot{\xi}_\tau^\lambda \varpi(\tau) \right)^\lambda \frac{\partial \xi_\tau}{\partial \xi_t} \\
 &= \mathcal{J}_{\tau,t} \left(\frac{\partial \ddot{\xi}_\tau}{\partial \xi_t} - \frac{1}{2} \varpi_\tau^\lambda \frac{\partial \dot{\xi}_\tau}{\partial \xi_t} + \frac{1}{2} \dot{\xi}_\tau^\lambda \left(\mathcal{J}_{\tau,t} \frac{\partial \dot{\xi}_\tau}{\partial \xi_t} - \frac{1}{2} \dot{\xi}_\tau^\lambda \frac{\partial \xi_\tau}{\partial \xi_t} \right) \right) - \frac{1}{2} \left(\ddot{\xi}_\tau + \frac{1}{2} \dot{\xi}_\tau^\lambda \varpi_\tau \right)^\lambda \frac{\partial \xi_\tau}{\partial \xi_t} \\
 &= \mathcal{J}_{\tau,t} \frac{\partial \ddot{\xi}_\tau}{\partial \xi_t} + \mathcal{J}_{\tau,t} \left(-\frac{1}{2} \varpi_\tau^\lambda + \frac{1}{2} \dot{\xi}_\tau^\lambda \mathcal{J}_{\tau,t} \right) \frac{\partial \dot{\xi}_\tau}{\partial \xi_t} + \\
 &\quad + \left(-\frac{1}{4} \mathcal{J}_{\tau,t} \dot{\xi}_\tau^\lambda \dot{\xi}_\tau^\lambda - \frac{1}{2} \left(\ddot{\xi}_\tau + \frac{1}{2} \dot{\xi}_\tau^\lambda \varpi_\tau \right)^\lambda \right) \frac{\partial \xi_\tau}{\partial \xi_t}.
 \end{aligned} \tag{B.51}$$

For the sake of convenience, I do not provide full expansions of (B.49) and (B.51) to calculate the derivative of the interpolated velocity and acceleration³. All necessary calculations have already been presented in the above sections.

To implement (B.49) and (B.51), one only needs to insert the related partial derivatives of the interpolated pose ξ_τ with respect to the associated local pose ξ_t , where the timestamp t varies between i and j in our GP priors. The reader can also find our implementation in the code.

³As I presented in Section 3.3, I do not estimate the accelerations of the GP-WNOJ prior in this thesis. Therefore, the transition matrices $\Lambda(\tau)$ and $\Omega(\tau)$ calculated in the applications present a dimension of $\mathbb{R}^{12 \times 12}$. However, I still provide the full implementation of the acceleration interpolation and its derivative in this section.

B.3 Factor Formulation

B.3.1 Pre-Integrated IMU Factor

Following [233], we define the error function of the IMU factor between two consecutive state variables at timestamps t_i, t_j as

$$\|e_{ij}^{\text{imu}}\|^2 = \left\| \begin{bmatrix} \mathbf{r}_{\Delta R_{ij}}^T & \mathbf{r}_{\Delta v_{ij}}^T & \mathbf{r}_{\Delta p_{ij}}^T \end{bmatrix} \right\|_{\Sigma^{\text{imu}}}^2, \quad (\text{B.52})$$

where

$$\mathbf{r}_{\Delta R_{ij}} = \text{Log}(\Delta \tilde{\mathbf{R}}_{ij}(\mathbf{b}_i^g)) \mathbf{R}_i^T \mathbf{R}_j, \quad (\text{B.53})$$

$$\mathbf{r}_{\Delta v_{ij}} = \mathbf{R}_i^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a), \quad (\text{B.54})$$

$$\mathbf{r}_{\Delta p_{ij}} = \mathbf{R}_i^T (\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta t_{ij} - \frac{1}{2} \mathbf{g} \Delta t_{ij}^2) - \Delta \tilde{\mathbf{p}}_{ij}(\mathbf{b}_i^g, \mathbf{b}_i^a). \quad (\text{B.55})$$

In (B.53) to (B.55), I omit the bias derivatives that can be ignored between two state variables. The motion increments $\{\Delta \tilde{\mathbf{R}}_{ij} \Delta \tilde{\mathbf{v}}_{ij} \Delta \tilde{\mathbf{p}}_{ij}\}$ are provided by the IMU pre-integration with

$$\Delta \tilde{\mathbf{R}}_{ij} = \prod_{k=i}^{j-1} \text{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g - \boldsymbol{\eta}_k^g) \Delta t), \quad (\text{B.56})$$

$$\Delta \tilde{\mathbf{v}}_{ij} = \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{ik} (\tilde{\mathbf{a}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^a) \Delta t, \quad (\text{B.57})$$

$$\Delta \tilde{\mathbf{p}}_{ij} = \sum_{k=i}^{j-1} \left[\Delta \mathbf{v}_{ik} \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} (\tilde{\mathbf{a}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^a) \Delta t^2 \right], \quad (\text{B.58})$$

where the raw vehicle acceleration $\tilde{\mathbf{a}}$ and rotation rate $\tilde{\boldsymbol{\omega}}$ from the IMU are integrated. The pre-defined noise parameters $\{\boldsymbol{\eta}_a \boldsymbol{\eta}_g\}$ are propagated to acquire the covariance matrix Σ^{imu} [233]. The gravity vector is updated according to the current position in the e frame for each pre-integration.

B.3.2 IMU-Bias Factor

As in [233], I estimate the accelerometer and gyroscope biases with the Brownian motion model by formulating the bias error function as

$$\|e_{ij}^b\|^2 = \|\mathbf{b}_j^a - \mathbf{b}_i^a\|_{\Sigma^{\text{ba}}}^2 + \|\mathbf{b}_j^g - \mathbf{b}_i^g\|_{\Sigma^{\text{bg}}}^2. \quad (\text{B.59})$$

B.3.3 Between-Pose Factor

For the relative odometry observations $\Delta\tilde{\mathbf{T}}_{i,j}^e = \{\Delta\tilde{\mathbf{R}}_{i,j}^e, \Delta\tilde{\mathbf{p}}_{i,j}^e\}$, I follow the original implementation in [265] and formulate the between pose factor represented as

$$\|\mathbf{e}_{i,j}^{\text{bp}}\|^2 = \left\| \ln(\mathbf{T}_i^{e,-1} \mathbf{T}_j^e \Delta\tilde{\mathbf{T}}_{i,j}^e)^\vee \right\|_{\Sigma^{\text{bp}}}^2, \quad (\text{B.60})$$

where the pose $\mathbf{T}_i^e$ and $\mathbf{T}_j^e$ are queried using timestamps associated with two successive lidar scans.

B.3.4 Velocity Factor

I use the 2-D observations $\tilde{\mathbf{v}}_t^b$ to formulate the navigation velocity factor. As the measured velocity can be directly associated with the velocity in state variables, as denoted in (A.118), I formulate the error function for the velocity observations considering the lever arm $\mathbf{l}^{b,\text{vel}}$ from the body frame to the sensor center as

$$\|\mathbf{e}_i^{\text{vel}}\|^2 = \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot (\mathbf{R}_{e,i}^b \mathbf{v}_{b,i}^e + \boldsymbol{\omega}_i^{b\wedge} \mathbf{l}^{b,\text{vel}}) - \tilde{\mathbf{v}}_i^b \right\|_{\Sigma^{\text{vel}}}^2. \quad (\text{B.61})$$

B.3.5 GNSS-PVT Factor

I propose a generalized implementation of the GNSS-PVT factor for the observed antenna position $\tilde{\mathbf{p}}_{\text{ant}}^e$ and the velocity $\tilde{\mathbf{v}}_{\text{ant}}^n$. Taking into account the lever arm $\mathbf{l}_{\text{ant}}^b$ from the IMU center to the phase center of the GNSS antenna, I calculate the antenna position at timestamp t_i as $\mathbf{p}_{\text{ant},i}^e = \mathbf{p}_{b,i}^e + \mathbf{R}_{b,i}^e \mathbf{l}_{\text{ant}}^b$ and velocity as $\mathbf{v}_{\text{ant},i}^e = \mathbf{v}_{b,i}^e + \mathbf{R}_{b,i}^e (\boldsymbol{\omega}_i^b)^\wedge \mathbf{l}_{\text{ant}}^b$. Thus, the error function can be derived as

$$\|\mathbf{e}_i^{\text{pvt}}\|^2 = \left\| [\mathbf{r}_{p_i}^T \mathbf{r}_{v_i}^T]^T \right\|_{\Sigma^{\text{pvt}}}^2, \quad (\text{B.62})$$

with

$$\mathbf{r}_{p_i} = \mathbf{p}_{\text{ant},i}^e - \tilde{\mathbf{p}}_{\text{ant},i}^e, \quad (\text{B.63})$$

$$\mathbf{r}_{v_i} = \mathbf{R}_{e,i}^n \mathbf{v}_{\text{ant},i}^e - \tilde{\mathbf{v}}_{\text{ant},i}^n, \quad (\text{B.64})$$

where the rotation matrix $\mathbf{R}_{e,i}^n$ is given in (A.90) by substituting the geodetic coordinate of the main antenna $\mathbf{p}_{\text{ant},i}^{\text{LLH}}$ as the origin. I use the measured standard deviations in the GNSS solutions to formulate the covariance matrix Σ^{pvt} .

B.3.6 Pseudorange and Doppler-shift (PrDo) Factor

I derive the error function for the pre-processed pseudorange and Doppler-shift observations with (A.116) and (A.117) as

$$\|e_i^{\text{PrDo}}\|^2 = \left\| \begin{bmatrix} r_i^{\text{Pr}} & r_i^{\text{Do}} \end{bmatrix}^T \right\|_{\Sigma^{\text{PrDo}}}^2, \quad (\text{B.65})$$

where

$$r_i^{\text{Pr}} = \left\| \mathbf{p}_{\text{ant},i}^e - \mathbf{p}_{\text{sat},k,i}^e \right\| + c_{b,i} - \rho_{k,i}, \quad (\text{B.66})$$

$$r_i^{\text{Do}} = (\mathbf{u}_{\text{ant},i}^{\text{sat}})^T \left(\mathbf{v}_{\text{ant},i}^e - \mathbf{v}_{\text{sat},k,i}^e \right) + c_{d,i} + \lambda_c \Delta f_{k,i}. \quad (\text{B.67})$$

I consider a scaled carrier-to-noise ratio (C/N_0) with hyperparameters λ_ρ and $\lambda_{\Delta f_k}$ to represent the variance of pseudorange and Doppler-shift observations, which is denoted as

$$\eta_\rho^2 = \lambda_\rho 10^{-\frac{C/N_0}{10}} \text{ and } \eta_{\Delta f}^2 = \lambda_{\Delta f} 10^{-\frac{C/N_0}{10}}. \quad (\text{B.68})$$

B.3.7 GNSS Receiver Clock Error Factor

In the tight coupling of the raw GNSS observations, the unknown receiver clock bias and drift (cbd) are estimated in the state variable by assuming a constant drifting model, which can be fused as

$$\|e_i^{\text{cbd}}\|^2 = \left\| \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_{b,i-1} \\ c_{d,i-1} \end{bmatrix} - \begin{bmatrix} c_{b,i} \\ c_{d,i} \end{bmatrix} \right\|_{\Sigma^{\text{cbd}}}^2. \quad (\text{B.69})$$

B.4 Noise Models

In this thesis, I formulate the covariance matrices of the GNSS-related factors using noise values provided in GNSS observations, as presented in Section B.3.5 and Section B.3.6.

The IMU noise is characterized by the Allan noise parameters, which is used to calculate $\Sigma^{\text{imu}} \in \mathbb{R}^{6 \times 6}$, $\Sigma^{\text{ba}} \in \mathbb{R}^{3 \times 3}$, and $\Sigma^{\text{bg}} \in \mathbb{R}^{3 \times 3}$.

Because there are no noise indicators for lidar odometry, speed sensor, and receiver clock errors, I formulate the covariance matrices $\Sigma^{\text{bp}} \in \mathbb{R}^{6 \times 6}$, $\Sigma^{\text{vel}} \in \mathbb{R}^{2 \times 2}$, and $\Sigma^{\text{cbd}} \in \mathbb{R}^{2 \times 2}$ manually as diagonal matrices using ad hoc parameters.

Noise models of different factors can be configured to use M-estimators [1]. In the experiments, I use the M-estimator with *Cauchy* loss [162] to adapt the noise model of the factors, such as GNSS-PVT factors, which may be affected by outlier measurements due to strong corruption in urban areas.

APPENDIX C

Supplementary Materials for Factor Graph Optimization

C.1	Implementation Details	219
C.1.1	Application I: GNSS/INS Fusion	219
C.1.2	Application II: GNSS-Multisensor Fusion	220
C.2	Extensive Experimental Analysis of GNSS/INS Fusion	221
C.2.1	General Solver Configurations	221
C.2.2	Computation Time	223
C.2.3	Size of Fixed Lag	223
C.3	Extensive Analysis of GNSS/Multisensor Fusion	226
C.3.1	Analysis of Challenging Scenarios	226
C.3.2	Smoother Type and Computation Time	229
C.3.3	GP-WNOA/WNOJ Motion Model	231

C.1 Implementation Details

C.1.1 Application I: GNSS/INS Fusion

C.1.1.1 Implementation

This application is implemented in C++ using the open-source library GTSAM¹, which provides core implementations of factor formulations and nonlinear factor graph optimization.

I assume that all data, including IMU measurements and pre-processed GNSS observations, are available. Consequently, the application operates in an offline manner, enabling efficient experimental evaluation of different motion priors and error models.

¹<https://gtsam.org>

C.1.1.2 Parameterization and Initialization

The initial state is parameterized using the Position, Velocity, and Time (PVT) solution from the GNSS receiver and is used to formulate the prior factor for the first optimization epoch. The initial state's standard deviations (std.) are given in Table C.1. The sensor noise parameters are shown in Table C.2.

Table C.1: Initial Uncertainty Parameterization in Application I (reproduced from [M1] with permission of IEEE).

State	σ_0
Position error N/E	5 m
Position error D	10 m
Velocity error	1 m s ⁻¹
Orientation error	5 deg
Acc. bias	0.1 m s ⁻²
Gyro. bias	0.01 deg/s
Receiver clock bias	1000 m
Receiver clock drift	10 m s ⁻¹

Table C.2: Noise Parameters in Application I (reproduced from [M1] with permission of IEEE).

Sensor	η
Acc.	$0.8 \times 10^{-3} \text{ m s}^{-2}$
Gyro.	$0.52 \times 10^{-3} \text{ rad s}^{-1}$
Acc. bias	$0.4 \times 10^{-3} \text{ m s}^{-2}$
Gyro. bias	$0.87 \times 10^{-3} \text{ rad s}^{-1}$
Pseudorange	2400 m
Deltarange	8 m s ⁻¹

C.1.2 Application II: GNSS-Multisensor Fusion

C.1.2.1 Implementation

I implemented my approach in C++ using Robot Operating System ROS2². The open-source software library GTSAM was extended to implement the graph and factor formulations.

I adopted the software solution for lidar odometry from LIO-SAM³, where only the front-end feature extraction and association were adapted in this work. I used the positioning and

²<https://docs.ros.org/en/humble/index.html>

³<https://github.com/TixiaoShan/LIO-SAM>

orientation estimation solution from a dual-antenna GNSS setup to initialize the state variable $\mathbf{x}_0$.

In this work, I used a laptop with an Intel i9-9900K, 16 cores at maximum 4.7 GHz and 64 GB memory for sensor pre-processing and graph optimization in experimental studies.

C.1.2.2 Near-Zero-Velocity Detection

While the vehicle is stationary, the state estimation exhibits random pose drift. This is a known problem in vehicle localization using inertial measurements [266]. In this case, state observability degrades dramatically due to insufficient IMU excitation, resulting in unbounded error accumulation. Thus, I follow the idea proposed in Wiseth et al. [266] to detect near-zero velocity motion by voting through multiple sensors that provide velocity information. If the vehicle is deemed stationary, I temporarily pause the graph optimization and state propagation.

C.1.2.3 Optimization Frequency

By default, I extend and optimize the time-centric graph at 10 Hz in my experiments to achieve a good balance of accuracy and runtime efficiency. Although these frequencies can be flexibly configured in the proposed estimation framework, I found that optimizing the graph at 5 Hz is a threshold that avoids discontinuities (jumps) in the estimated trajectory in my application. For applications restricted by low-performance computing devices, choosing a higher frequency to extend the graph and a lower frequency for optimization can be considered.

Figure C.1 presents the qualitative results on choosing the optimization frequencies, where I have investigated different frequencies to analyze the “jump” effect in the scenario where the vehicle was turning, since the optimization frequency of my estimation framework gnsSFGO can be configured. I found that an optimization frequency of 5 Hz is the threshold to avoid the “jump” effect. However, this threshold can be relaxed if the time difference between the state variables is small (e.g., 0.1 s) when comparing Figure C.1e and Figure C.1f.

C.2 Extensive Experimental Analysis of GNSS/INS Fusion

C.2.1 General Solver Configurations

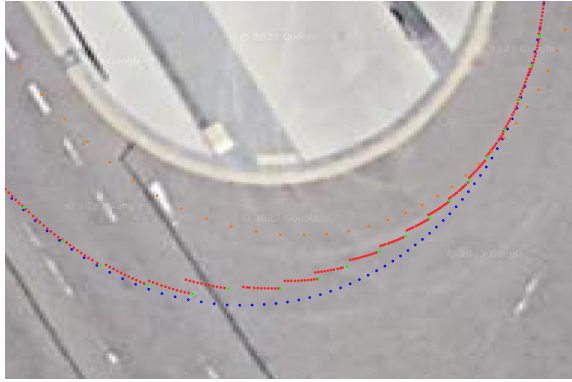
In my experiments, I evaluate continuous-time and discrete-time FGO using three solver settings: batch optimization with fixed lag, iSAM2 with and without fixed lag. As shown in



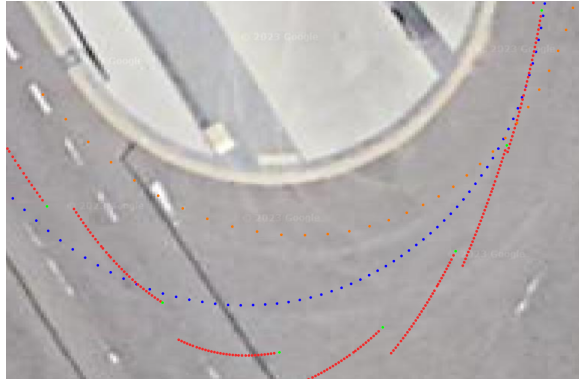
(a) Trajectory with state and optimization frequency at 10 Hz.



(b) Trajectory with state and optimization frequency at 5 Hz.



(c) Trajectory with state and optimization frequency at 3 Hz.



(d) Trajectory with state and optimization frequency at 1 Hz.



(e) Trajectory with state frequency at 10 Hz and optimization frequency at 3 Hz.



(f) Trajectory with state frequency at 10 Hz and optimization frequency at 1 Hz.

Figure C.1: Trajectories of loosely coupled fusion using different state and optimization frequencies. Blue: reference trajectory of the NovAtel GNSS receiver, orange: trajectory of the ublox-f9p GNSS receiver, red: propagated trajectory at IMU frequency, green: optimized trajectory of gnssFGO.

Table 3.1, both batch and iSAM2 solvers with a 0.5s fixed-lag (F.L.) perform well among other configurations, as discussed in Section C.2.3. However, I highlight the iSAM2 with fixed-lag as the primary solver due to its superior performance and computational time efficiency.

C.2.2 Computation Time

I calculate the elapsed time of the UKF and FGO with different settings on the same computer with an Intel i7-11800H CPU. The results are shown in Table C.3. Although the FGO with all settings requires more computation time compared to the UKF, the online application of FGO using iSAM2 with fixed lag can still be realized by taking the elapsed time of 864.9s for the entire test route of 5583.3 s as a reference.

Table C.3: Elapsed Computation Time in Application I (reproduced from [M1] with permission of IEEE).

Method	UKF ⁴	FGO - Batch		FGO - iSAM2		
Settings	-	Lag 0.5 s	Lag 10 s	w/o. Lag	Lag 0.5 s	Lag 10 s
Time (s)	603.3	1355.8	17063.6	9687.9	864.9	5520.1

C.2.3 Size of Fixed Lag

Furthermore, I experiment with the lag size in terms of accuracy and computation time, as shown in Table C.4.

I found that increasing the lag size to integrate more past factors into the FGO doesn't necessarily improve the performance in this scenario. Taking the result of route #3 in Table C.4 as an instance, the batch solver with a larger lag size shows a more significant 2D error.

The same conclusion can also be drawn by the iSAM2 solver with a fixed-lag between the lag sizes 0.5s and 10s. Because the iSAM2 solver does not update all entries in the information matrix, the lag size of 0.5 s performs better than 0.1 s. However, the general improvement gain with a larger size is not observed when the lag size exceeds 0.5 s.

In fact, marginalizing the factors out of a short lag size is sufficient to guarantee performance, as the factors in the past don't contribute additional information. Even worse, if a factor in the lag is highly inconsistent due to the measurement outliers, the solver with a larger lag size can converge more slowly.

⁴The UKF was implemented in MATLAB/Simulink.

Table C.4: Evaluation of the Fixed-Lag Size in Application I (reproduced from [M1] with permission of IEEE).

	Batch - Fixed-Lag			iSAM2 - Fixed-Lag			
	0.1 s	0.5 s	10s	0.1 s	0.5 s	10 s	30 s
2D Error #1 (m)	0.660	0.646	1.507	0.678	0.637	0.644	0.892
2D Error #2 (m)	0.325	0.316	0.244	0.358	0.325	0.314	0.312
2D Error #3 (m)	0.563	0.568	0.638	0.651	0.607	0.626	0.626

C.2.3.1 Consistency of Measurement Noise Model

For both filtering-based and optimization-based algorithms, the inconsistency of the measurement noise model due to corrupted observations significantly affects performance. In my work, the GNSS factors are extended with the M-estimator using the Cauchy loss function by default, as denoted in Section B.3. To explore the performance gain of the M-estimator, I have also evaluated the continuous-time FGO without extending the GNSS factors using the Cauchy loss function.

The position errors are shown in Table 3.1, and Figure C.2 illustrates the trajectories with different noise modeling settings.

The impact of the M-estimator is shown distinctly by comparing the trajectories 4-8 of the FGO in Figure C.2 with initial GNSS noise parameters. Without the M-estimator, both discrete/continuous-time FGO drift is considerably similar to the UKF due to the highly corrupted GNSS observations (trajectory 7-8), while the FGO with M-estimator (trajectory 5-6) shows a much smaller drift.

But even with the M-estimator, the estimated trajectories drift significantly, which is unacceptable for vehicle control and safety. To indicate the critical factor of this problem, I have manually scaled the variance of GNSS observations to $\eta_\rho = 7200$ m and $\eta_v = 192$ m/s while crossing all bridges.

The trajectories 3-4 in Figure C.2 present the results of this experiment. It can be seen clearly that the estimated trajectories are less drifted, even without the M-estimator, if the GNSS noise parameters can be adapted properly in real-time. In Figure 3.6, the best trajectories are shown for the UKF and FGO with scaled GNSS noise parameters, and the M-estimator is applied in the FGO.

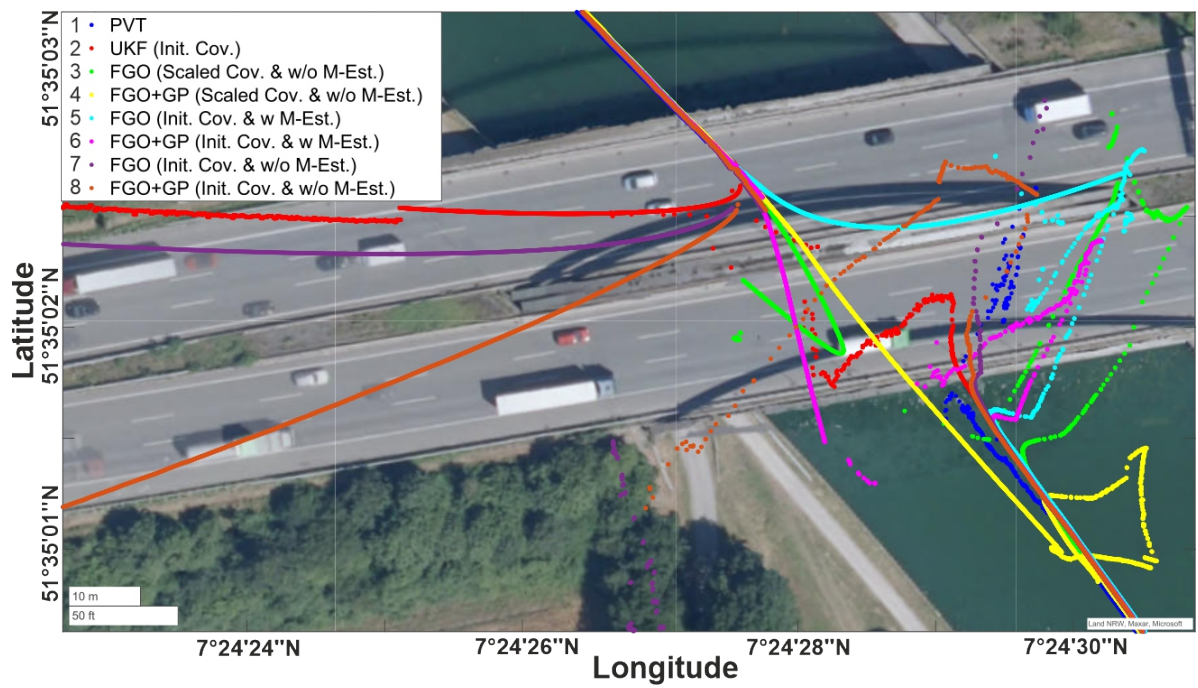


Figure C.2: Trajectories with different noise modeling while crossing bridge 9a (reproduced from [M1] with permission of IEEE).

C.3 Extensive Analysis of GNSS/Multisensor Fusion

C.3.1 Analysis of Challenging Scenarios

In this part, I propose experimental studies regarding GNSS observations, lidar odometry, and solver settings. I also evaluated the GP-WNOA and GP-WNOJ priors and discussed the hyperparameter tuning for Q_c .

C.3.1.1 Loss of GNSS Observation

Generally, losing GNSS observations in a short time interval does not lead to immediate divergence or trajectory drift if multiple state constraints, such as lidar odometry or motion prior factors, are still present. This conclusion can be drawn from my experiment in Seq. C01, where the vehicle crossed a large bridge at the central train station in Cologne, as shown in Figure 3.15.

It is also interesting to observe that the loss of GNSS observations is frequently accompanied by highly corrupted GNSS observations due to multipath effects. For example, fusion approaches may diverge when the vehicle approaches or leaves a tunnel. In this case, robustness can be significantly affected even if no local sensors are fused.

Furthermore, fusing GNSS observations in a tight coupling extends the state variables with receiver clock bias and drift $c^r = [c_b \ c_d]^T$, which become unobservable if fewer than four satellites are visible. Figure C.3 shows the estimated clock bias c_b with respect to the number of tracked satellites. The estimated clock bias drifts dramatically in the graph where only GNSS observations and IMU measurements are integrated. Even if the observability of the clock bias can be recovered, it takes some time for the state variable c^r to converge (see Figure C.3), which degrades the overall estimation performance.

Similar results can be observed in other experiments, where unobservable state variables can lead to estimation divergence and an ill-posed optimization problem. Fortunately, this problem can be eliminated in the graph fused with lidar odometry. Thanks to the between-state constraints that prevent other state variables (e.g., position and rotation) from divergence, robust trajectory estimation can be guaranteed.

In addition, a large trajectory drift can be expected if the global reference (e.g., GNSS observations) is lost over a long time interval, such as when crossing a long tunnel.

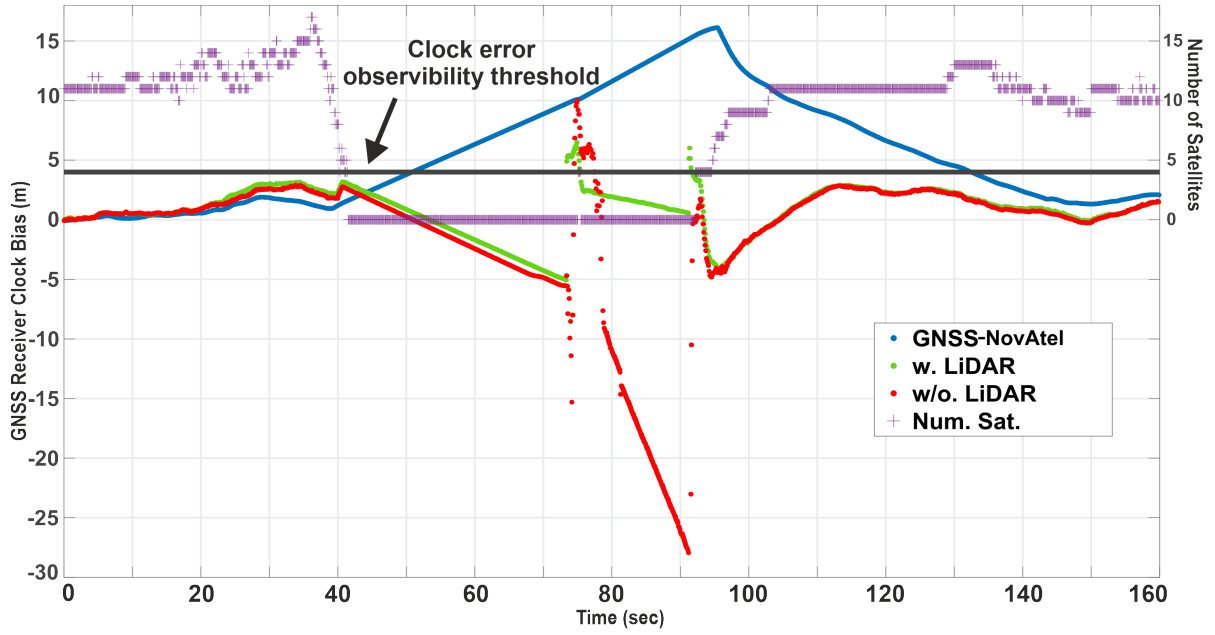


Figure C.3: Estimated GNSS receiver clock bias of Seq. C01 without near-zero-velocity detection. The receiver clock error is unobservable during tunnel crossing. In this scenario, fusing raw GNSS observations without lidar odometry cannot constrain the state estimation, resulting in trajectory drifting. The corresponding trajectories of both fusion approaches are presented in Figure 3.15 (reproduced from [M1] with permission of IEEE).

C.3.1.2 Highly Corrupted GNSS Observations

Compared to the temporary loss of GNSS observations, I emphasize that including highly corrupted GNSS observations in the graph has a greater impact on estimation performance. This conclusion can be supported by Seq. C02, where the accuracy of my proposed fusion paradigms is significantly degraded in GNSS-corrupted areas, as shown in Figure 3.12.

In Figure C.4, I plot the estimated trajectories in this scenario by transforming the coordinates in the navigation frame (ENU). A large trajectory drift up to 25 m can be observed in tightly coupled fusion without lidar odometry (see Figure C.4). Although fusing relative motion constraints, such as odometry, can effectively constrain divergence, trajectory drifts cannot be eliminated until valid global references are acquired.

C.3.1.3 Lidar Odometry Degradation

As discussed in [267], traditional lidar odometry algorithms suffer from dramatic degradation in unstructured environments and high-speed scenarios. This problem can also be observed in my experiments.

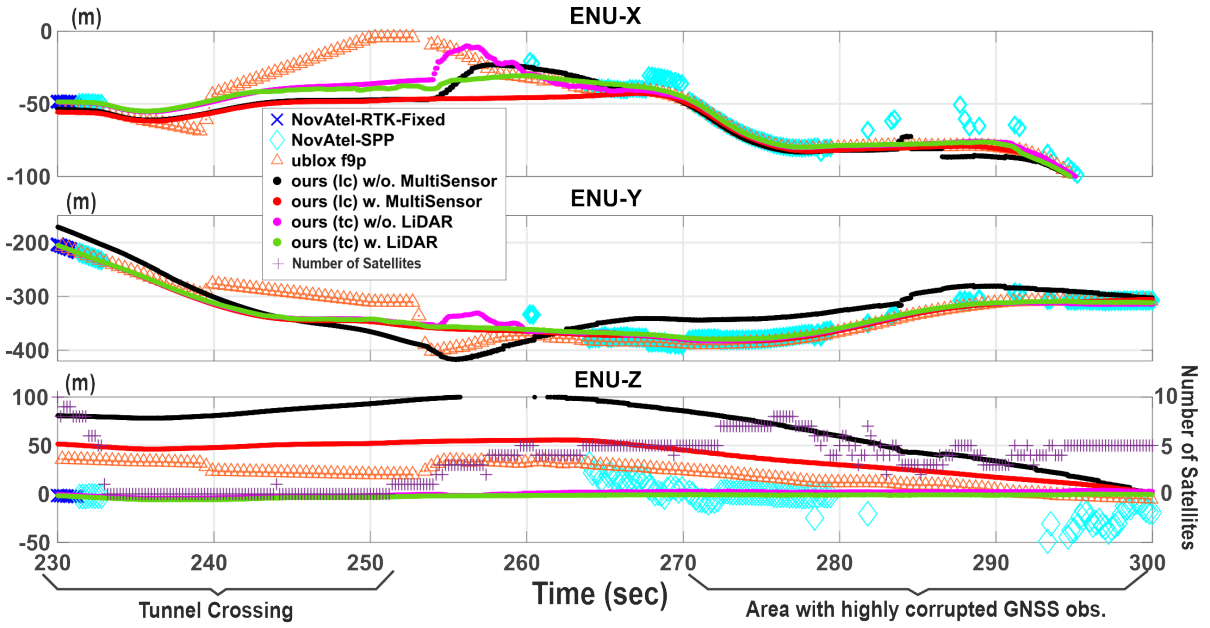


Figure C.4: Coordinates in the ENU frame of Seq. C02. The estimated trajectories become unsmooth if the GNSS observations are strongly corrupted in urban areas (reproduced from [M1] with permission of IEEE).

Figure C.5 illustrates three scenarios in which the accuracy of the lidar odometry is penalized if the vehicle is driving in featureless areas or in high-speed mode with an average vehicle speed of 125 km/h. In low-speed driving mode and open-sky areas, lidar degradation does not significantly impact the estimation performance, provided high-quality GNSS measurements are available.

However, if the vehicle moves at high speed, the lidar odometry becomes inaccurate because of motion distortion. Therefore, including lidar odometry factors in the graph can decrease localization accuracy, as reported in Table 3.3 of Seq. HS.

In scenarios with long tunnels, trajectory drifting can always be expected due to the loss of global reference. This presents the major limitation of classic lidar odometers that calculate only pose increments.

Another crucial aspect to be mentioned is the uncertainty of lidar odometry used in fusion approaches. Because no covariance is provided by classic scan matching algorithms, the pre-defined noise parameters (see Section B.4) may be over-confident. Therefore, I emphasize the importance of acquiring realistic uncertainty quantification for lidar odometry in future work.

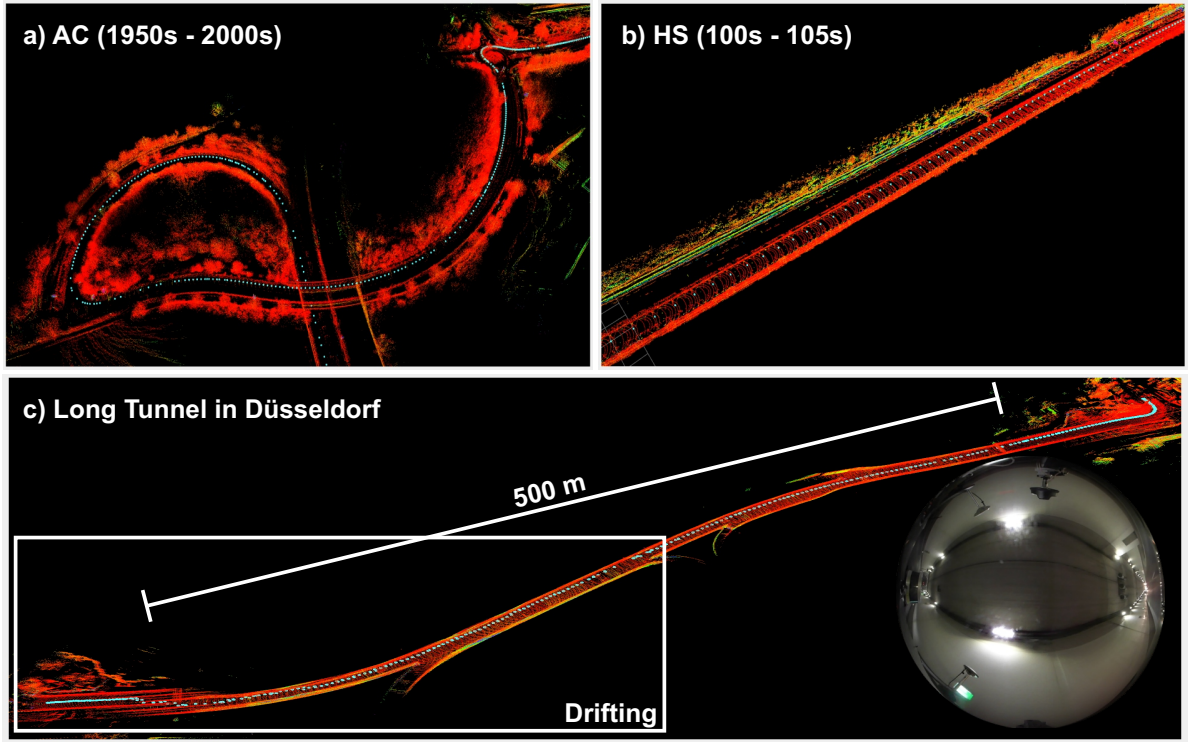


Figure C.5: Examples of lidar odometry degradation in three scenarios: a) unstructured feature-less area, b) high-speed scenario, and c) long tunnel 400 m (reproduced from [M1] with permission of IEEE).

C.3.2 Smoother Type and Computation Time

To study the impact of different smoother types and lag sizes, I evaluated batch and incremental smoother iSAM2 with different lag sizes for Seq. DUS. The performance metrics are presented in Table C.5.

Compared to an incremental smoother, solving the optimization problem with a batch optimizer does not show a considerable improvement in accuracy. This occurs because the graph structure becomes more similar to a Markov chain in large-scale localization applications with fewer loop closure constraints. In this scenario, re-linearizing all past state variables does not contribute more information that improves the accuracy. For loosely coupled fusion, the batch smoother presents a smoother trajectory. However, this advantage is absent when using the incremental smoother for fusing GNSS observations in a tight coupling.

Furthermore, the batch optimizer requires more computational resources than the incremental smoother (see Figure C.6), especially in urban areas with a higher number of measurement outliers. In online applications, estimation accuracy and trajectory smoothness can be penalized once optimization takes longer. This conclusion is supported by referring to the tightly

Table C.5: Estimation Performance of Seq. DUS using Different Solver Configurations, which are formulated as (fusion type)-(solver type)-(lag size), e.g., lc-batch-3sec (reproduced from [M1] with permission of IEEE).

Configuration	Mean 2-D Pos. Error (m)	Mean 3-D Pos. Error (m)	Mean Yaw Error (°)	Smoothness s
lc-batch-3sec	0.324	0.839	0.208	1526.4
lc-isam2-3sec	0.397	1.146	0.216	5672.8
tc-batch-3sec	1.543	2.008	0.497	1649.3
tc-isam2-10sec	1.419	1.733	0.573	1537.9
tc-isam2-3sec	1.524	1.766	0.511	1510.5

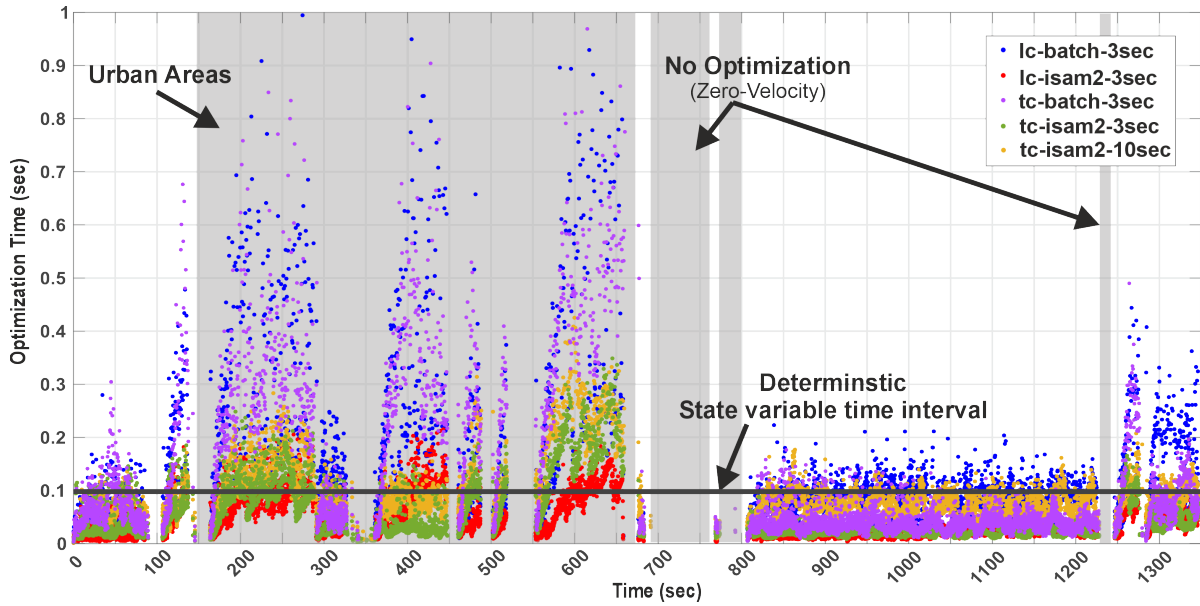


Figure C.6: Computation time with different configurations of Seq. DUS (reproduced from [M1] with permission of IEEE).

coupled fusion in Table C.5.

Similarly to the optimizer type, considering a large lag size does not contribute significantly. Moreover, even the incremental smoother with a large lag size frequently violates the desired optimization frequency, resulting in inefficient optimization procedures.

C.3.3 GP-WNOA/WNOJ Motion Model

In this section, I evaluated the continuous-time trajectory representation using the Gaussian process interpolation with both WNOJ and WNOA models. For a fair evaluation, the hyper-parameter q_c was manually tuned by penalizing the vehicle pose in Q_c with the same parameterization for pose weighting, aiming to evaluate the effect between the third-order and second-order dynamics models.

Compared to the GP-WNOJ model, a GP-WNOA model assumes that the system transition follows a constant velocity model [M1, 135]. As discussed in Tang et al. [103], representing vehicle trajectories with an approximately constant-velocity model may be insufficient in urban driving scenarios where vehicles frequently accelerate and brake.

To evaluate the performance of both GP models, I chose a part of Seq. AC containing 200 s test run in open-sky areas where the PVT solution from the high-grade GNSS receiver presents the ground-truth trajectory. I calculate the whitened error of the vehicle pose and linear velocity in the body frame and plot the results in a histogram in Figure C.7.

Because the GP-WNOJ model represents second-order system dynamics, it shows smaller errors in all linear velocity components. Both models perform similarly in position estimation, where the GP-WNOJ is more accurate in the main motion direction x -axis. For rotation, the GP-WNOJ does not present considerable improvements compared to the GP-WNOA. One possible reason supporting this result can be traced back to the rotational acceleration that cannot be directly observed using the IMU.

I have validated that the GP motion model formulates a valid continuous-time trajectory representation. However, tuning the power spectral matrix Q_c that scales the system transition in the Gaussian process kernel has a large effect on numerical stability and estimation performance [103]. Although the GP-WNOJ model presents reliable velocity estimates compared to the GP-WNOA model, it shows higher sensitivity on the power spectral matrix Q_c [103], which therefore requires more careful parameter tuning⁴ when incorporating accelerations in state propagation.

⁴In this work, I did not explicitly investigate parameter tuning for the power spectral matrix Q_c . As discussed in the original works [103, 148], this hyper-parameter can be calibrated using supervised-learning approaches. Recent work also explored this idea and showed the possibility of learning this parameter without ground-truth labels using variational Bayesian approaches [150, 268]. However, vehicle dynamics exhibit a high degree of variation in real-world driving scenarios; therefore, the parameter Q_c should be dynamically tuned in an online process, which remains my future work.

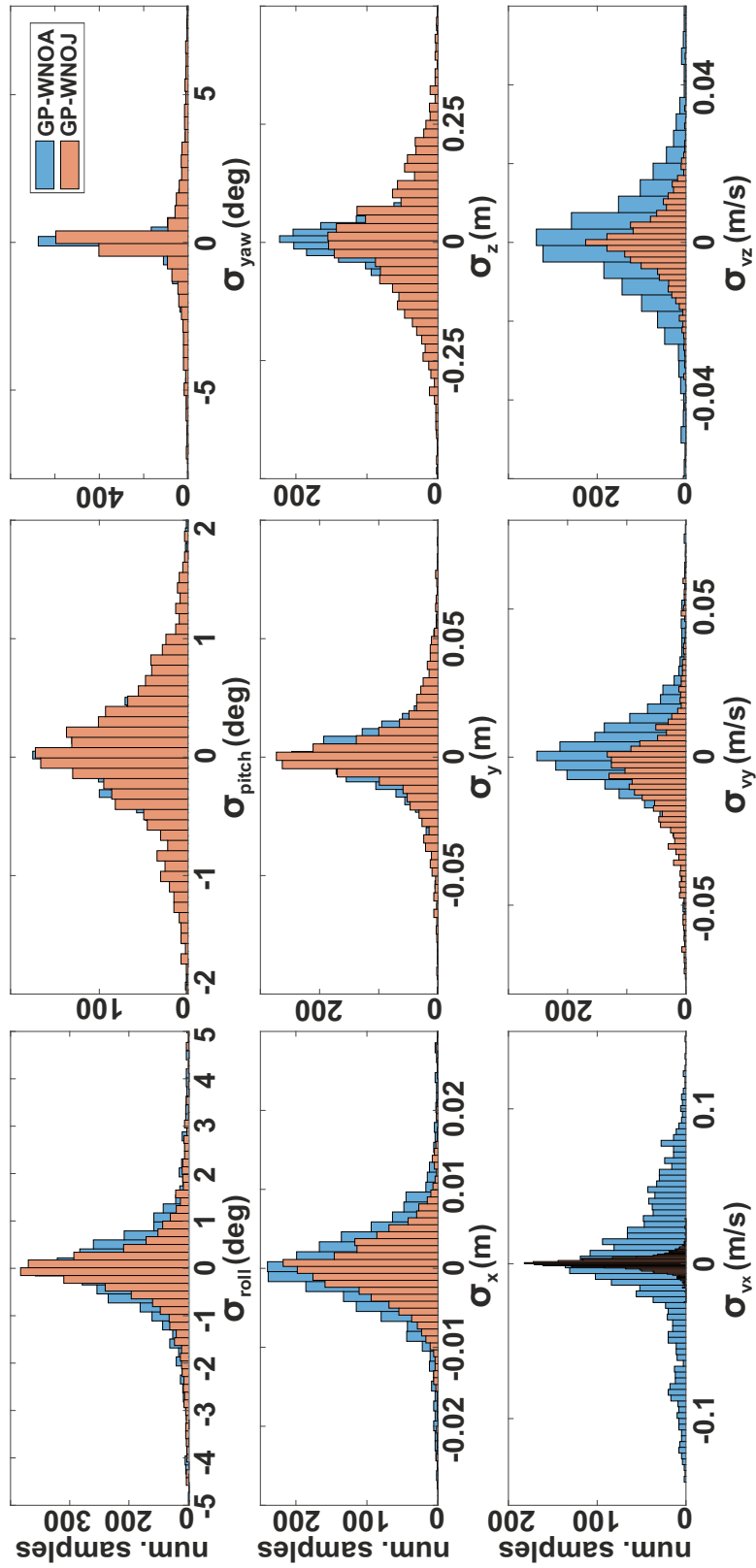


Figure C.7: Histogram of whitened state errors of GP models in an open-sky area (reproduced from [M1] with permission of IEEE).

APPENDIX D

Supplementary Materials for M-Estimation

D.1	Details of M-Estimators	233
D.1.1	Supplementary Plots of Loss Functions	233
D.1.2	Fair Estimator	235
D.1.3	Cauchy Estimator	237
D.1.4	Geman-McClure Estimator	237
D.1.5	Welsch Estimator	237
D.1.6	Huber Estimator	239
D.1.7	Tukey Estimator	239
D.2	Extensive Experimental Results	241

D.1 Details of M-Estimators

D.1.1 Supplementary Plots of Loss Functions

To support the discussions in Section 4.4.2, I attached the plots of all kernel functions at the other efficiency levels in this section.

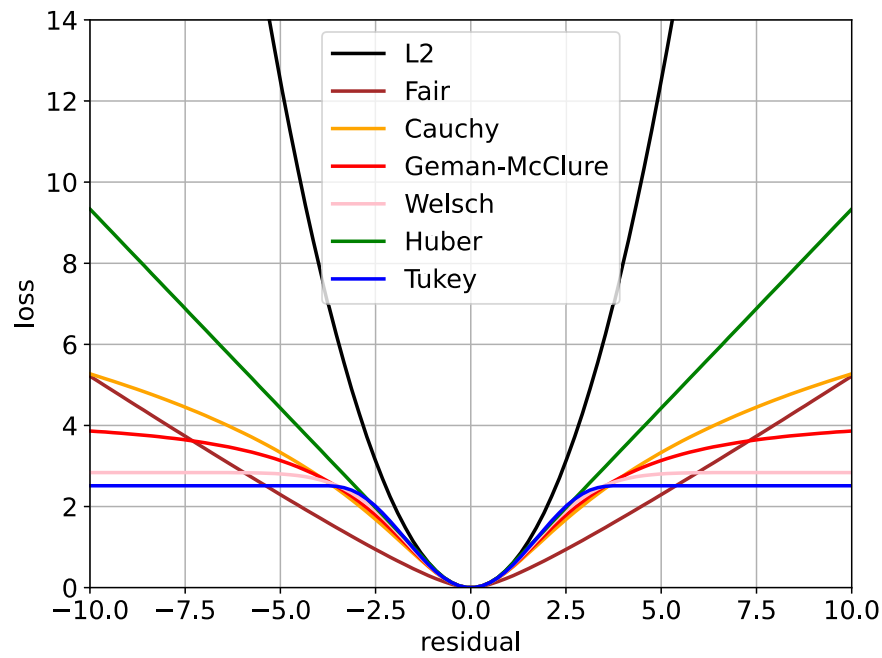


Figure D.1: Plot of loss values using different kernel functions at $E_{\text{ff}} = 90\%$.

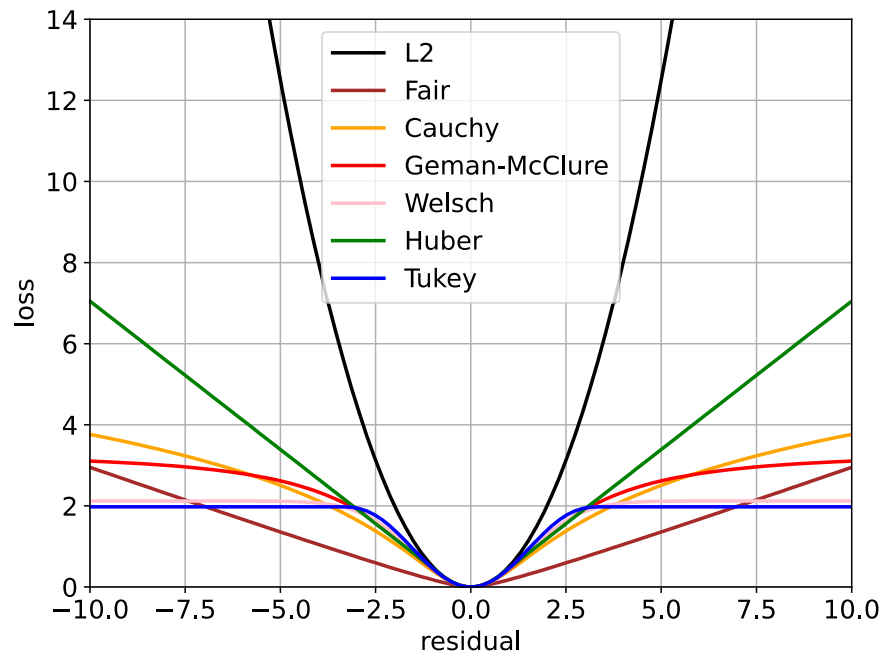


Figure D.2: Plot of loss values using different kernel functions at $E_{\text{ff}} = 85\%$.

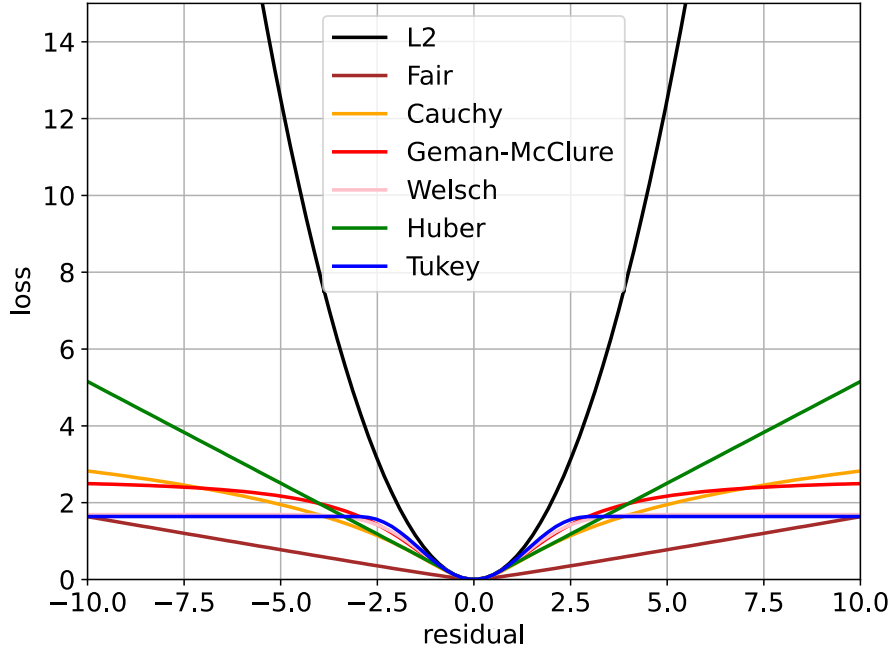


Figure D.3: Plot of loss values using different kernel functions at $E_{\text{ff}} = 80\%$.

D.1.2 Fair Estimator

The Fair estimator incorporates an $L1$ -like structure in the denominator, contrasting with the $L2$ -like structure. When solving problems such as one-dimensional location problems, datasets often lack prior knowledge of the error scale parameter. For instance, MacTavish and Barfoot [269] introduced error deflation as $\rho(r_i/d)$ to standardize tuning across loss types.

However, predicting scales accurately can be challenging, especially when homoscedasticity is not guaranteed. This makes it desirable to minimize sensitivity to the tuning constant [270].

As shown in Figure D.4, the Fair estimator is monotonous, and its unbounded property makes it less robust and convex. However, its influence function grows significantly slower than that of the $L2$ estimator.

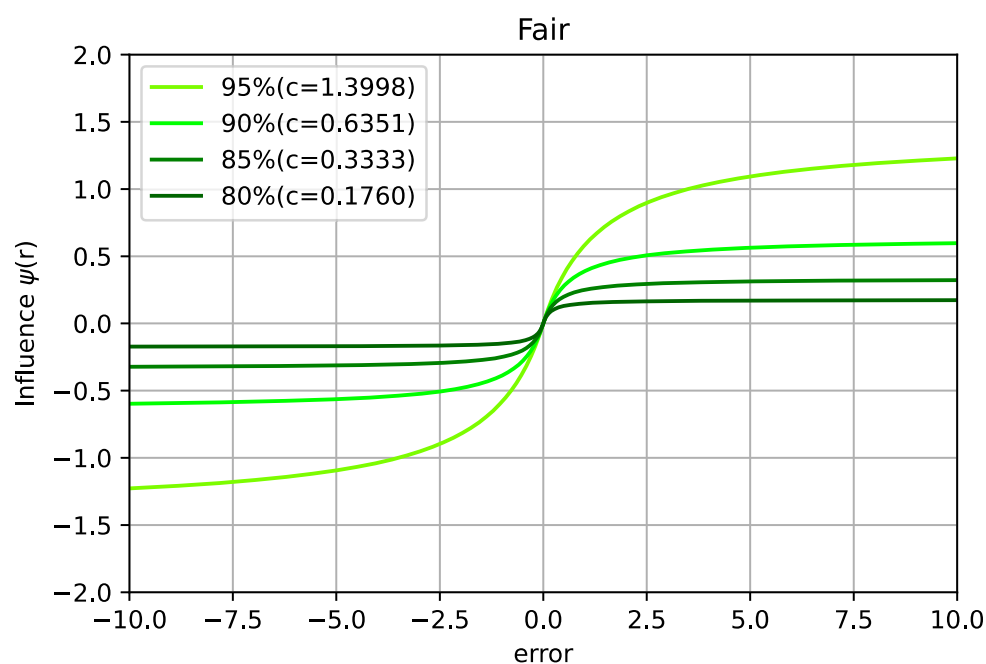


Figure D.4: Fair Influence at four levels of efficiency through tuning.

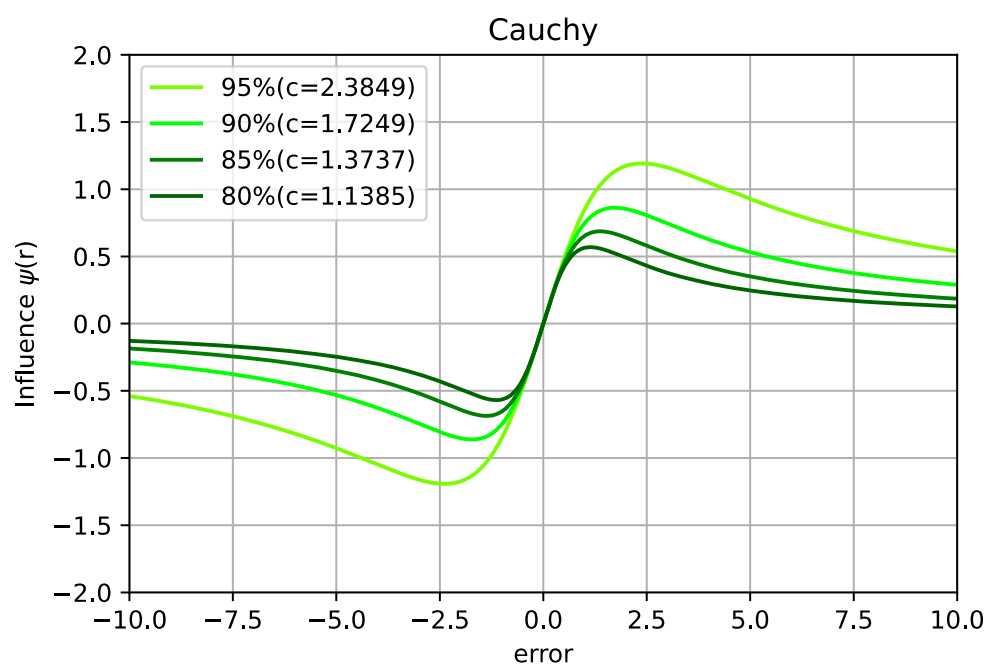


Figure D.5: Cauchy Influence function at four levels of efficiency through tuning.

D.1.3 Cauchy Estimator

The Cauchy distribution is often regarded as a "pathological" distribution in statistics, as both its expected value and variance are undefined [270].

Figure D.5 illustrates the Cauchy influence function at different efficiency levels achieved by tuning the parameter c .

The Cauchy influence function is bounded and does not feature a rejection point, classifying it as a soft-redescending estimator. Instead of nullifying the influence of outliers, it assigns them a small weight, making it robust to extreme values while maintaining some sensitivity.

D.1.4 Geman-McClure Estimator

The Geman-McClure loss function is designed to reduce the influence of outliers in data by assigning smaller penalties to large residuals.

The corresponding influence function is bounded, as shown in Figure D.6, ensuring that extreme residuals have a limited impact on the estimation process. There is a trade-off between achieving low gross-error sensitivity and maintaining high efficiency or low asymptotic variance when using this loss function.

Similar to the Cauchy estimator, it is a soft-redescending estimator but exhibits a larger gross-error sensitivity compared to Cauchy's. This property makes the Geman-McClure loss particularly effective in scenarios with significant noise or outliers, as it provides a smooth and efficient alternative to traditional quadratic loss functions.

D.1.5 Welsch Estimator

The Welsch loss function is designed to minimize the influence of outliers by heavily down-weighting large residuals.

The influence function of the Welsch estimator with four different relative efficiencies is shown in Figure D.7. As a soft-redescending estimator, the Welsch influence function does not descend to zero, meaning outliers retain minimal influence. However, unlike the Cauchy estimator, the Welsch estimator assigns a very small weight to spurious residuals, demonstrating its robustness against extreme values.

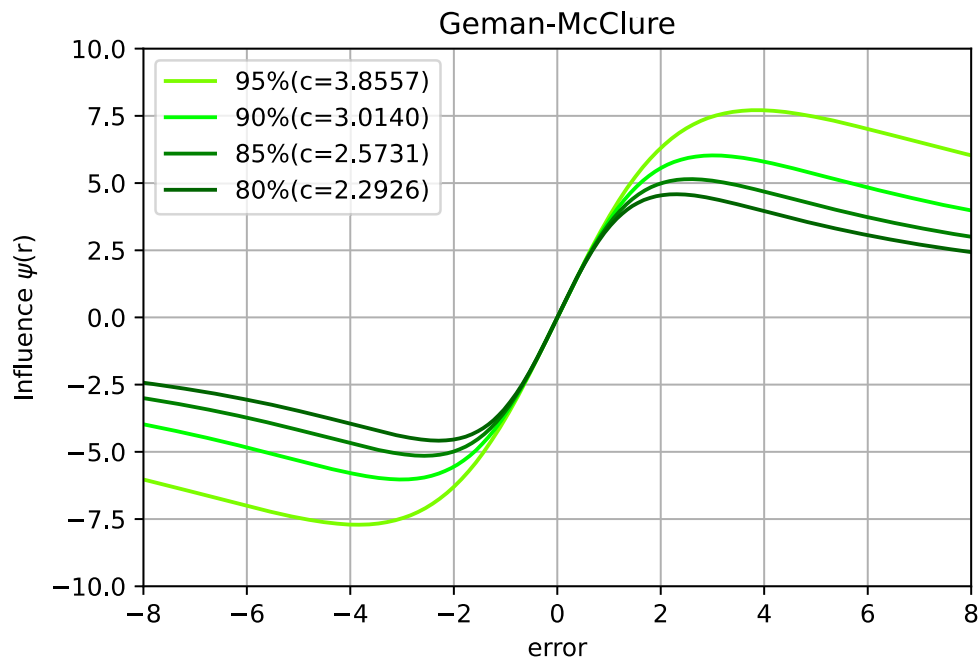


Figure D.6: Geman-McClure Influence function at four levels of efficiency.

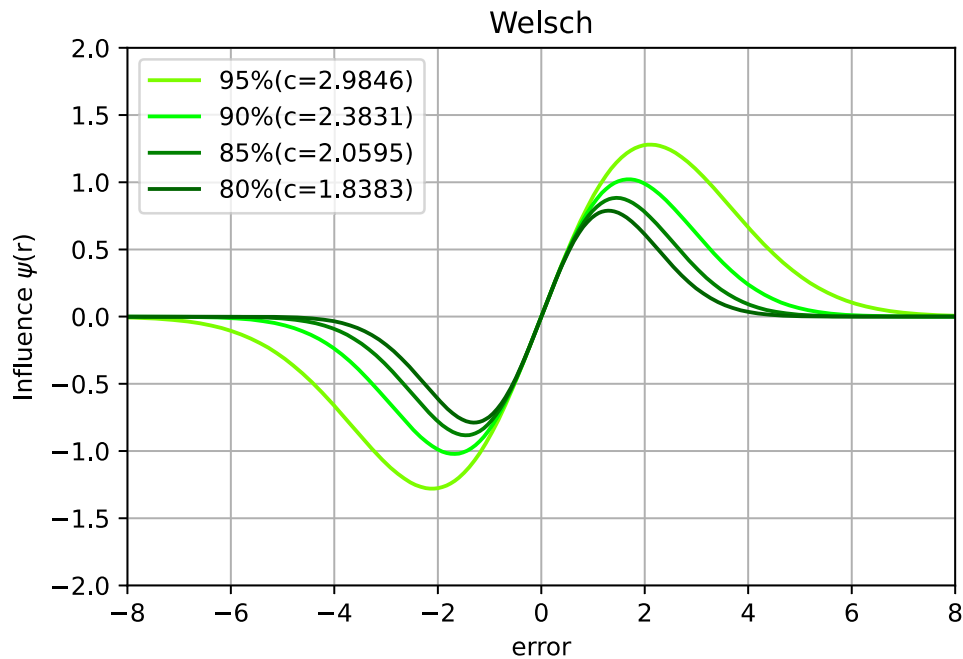


Figure D.7: Welsch Influence function at four levels of efficiency through tuning.

D.1.6 Huber Estimator

The Huber loss function was proposed as the *minimax* solution for the location problem, with a known scale parameter in the case of a contaminated normal distribution.

The Huber estimator is classified as a monotonous estimator [271]. As shown in Figure D.8, increasing efficiency results in a larger rejection point and increased gross-error sensitivity. A higher gross-error sensitivity implies reduced robustness, as potential contamination has a greater influence on the estimate.

The gross-error sensitivity also corresponds to the value of the tuning parameter, marking the boundary between L_2 and L_1 estimators. Beyond this boundary, the estimator assigns equal weight to all observations, regardless of their distance from the regression line.

D.1.7 Tukey Estimator

The Tukey (aka bisquare) loss function proposed by J. Tukey is designed to completely ignore extreme outliers by assigning them zero influence. The influence function of the Tukey estimator descends to zero beyond the threshold c , categorizing it as a hard-redescending estimator.

At higher efficiency levels, such as 95%, the estimator becomes more convex compared to lower levels like 90%, reflecting its basis in a normal distribution. As illustrated in Figure D.9, both the rejection point and gross-error sensitivity increase with efficiency. This indicates that it takes longer (in terms of standard deviation magnitude) to nullify the effect of spurious values at higher efficiency levels. For instance, a spurious value of $r = 3$ is rejected at an efficiency of 80 – 85% but remains influential at 90 – 95%.

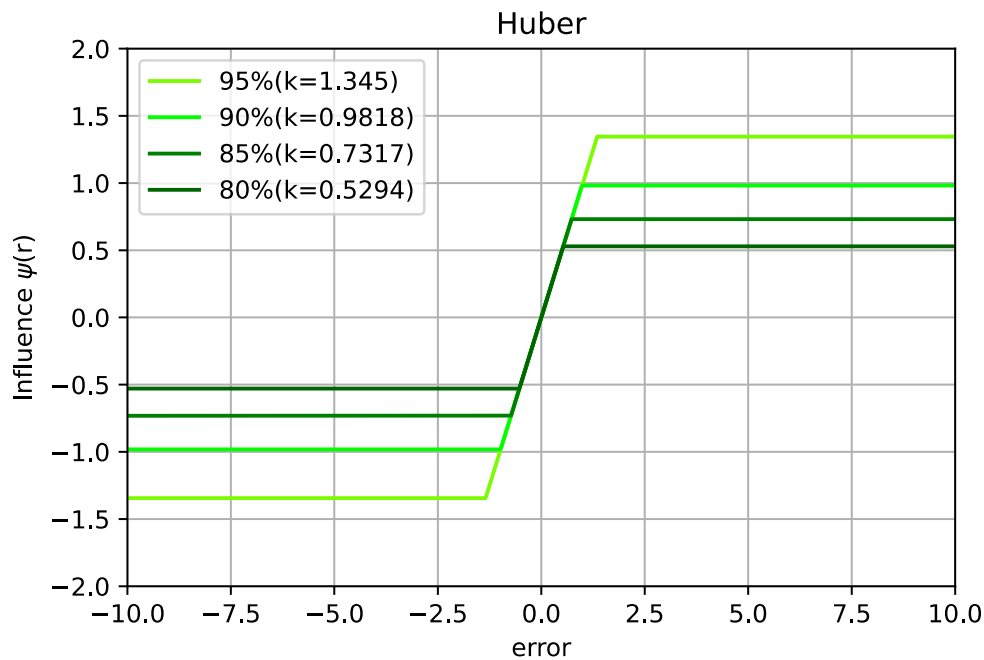


Figure D.8: Huber influence function curves at four levels of efficiency.

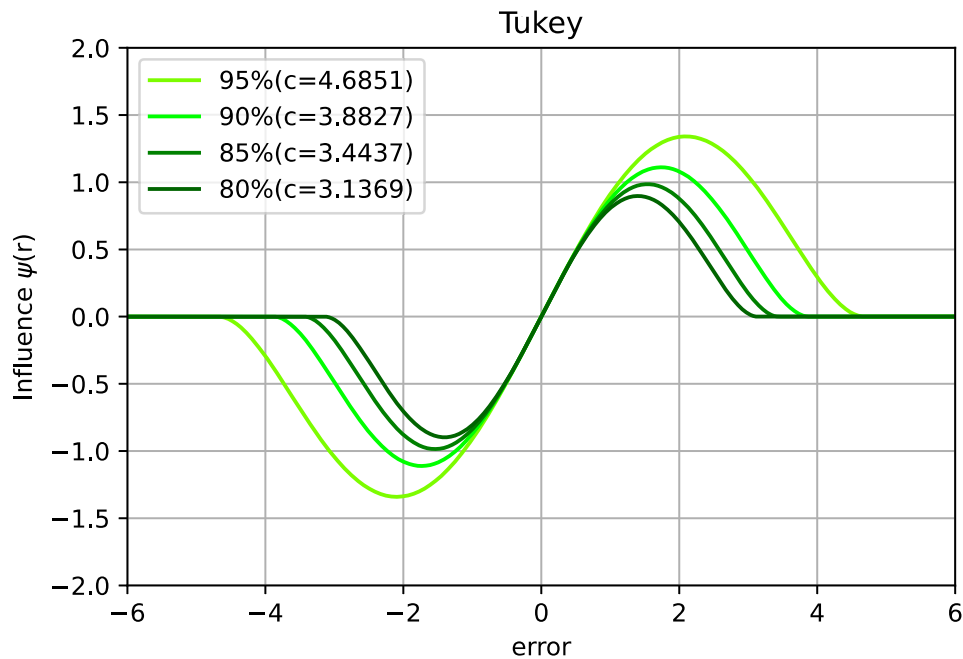


Figure D.9: Tukey influence function at four levels.

D.2 Extensive Experimental Results

In this section, I present 2-D error metric figures for each kernel function across all test sequences. These figures serve as a supplementary reference to the discussions in Section 6.2.2.

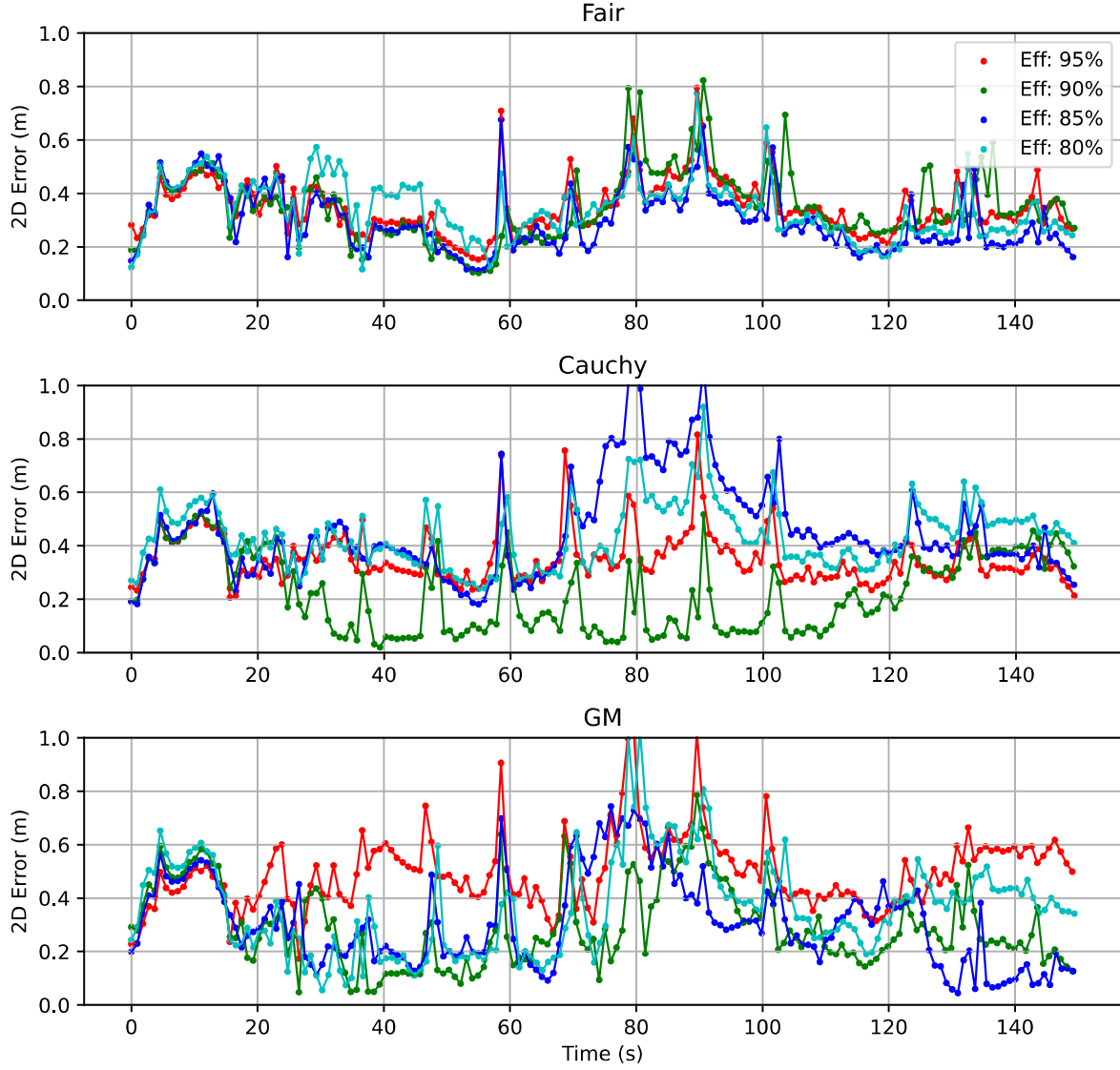


Figure D.10: Plot of 2D error of sequence #1 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval where the reported error equals zero.

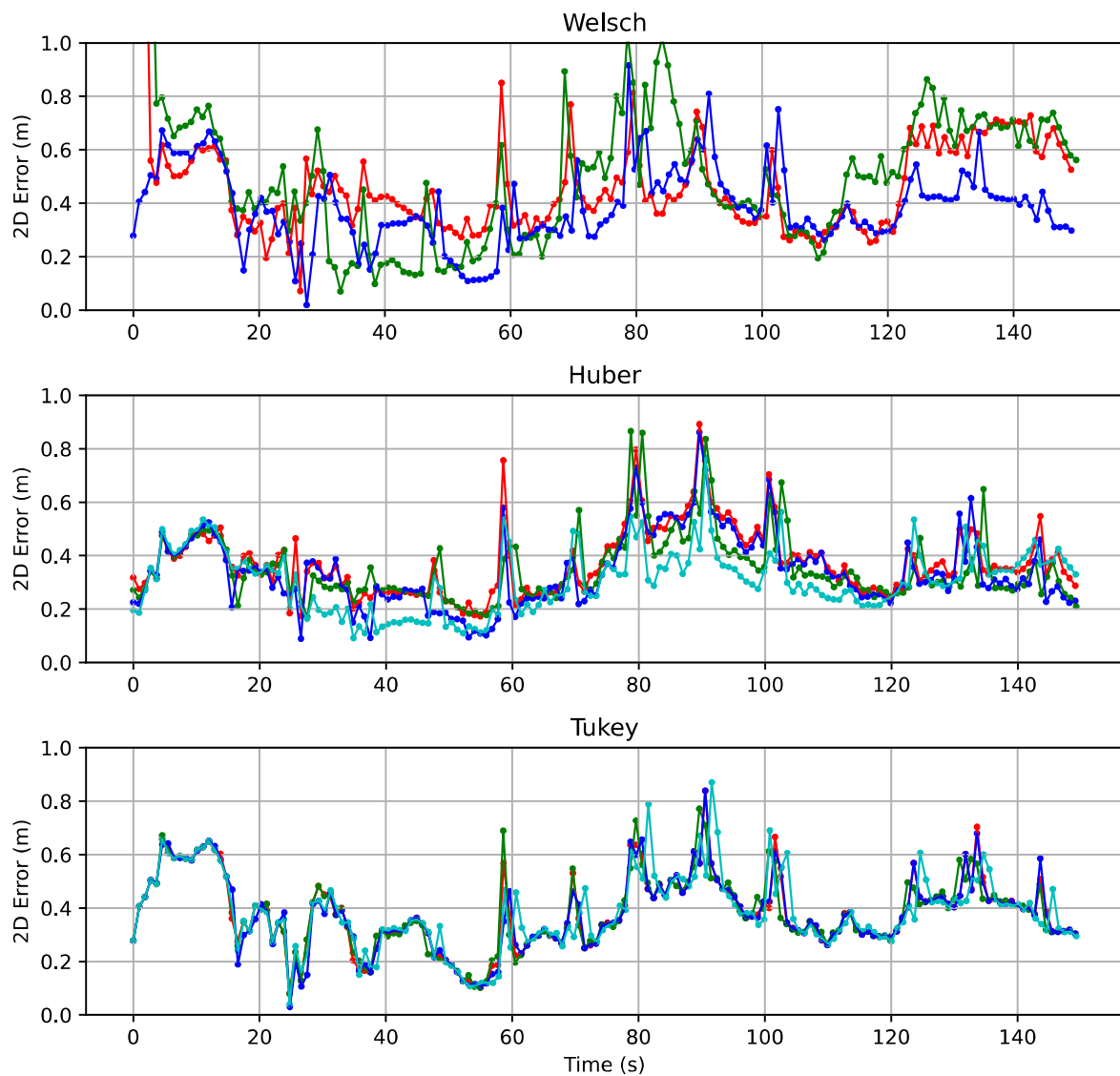


Figure D.10: (Continued) Plot of 2D error of sequence #1 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval where the reported error equals zero.

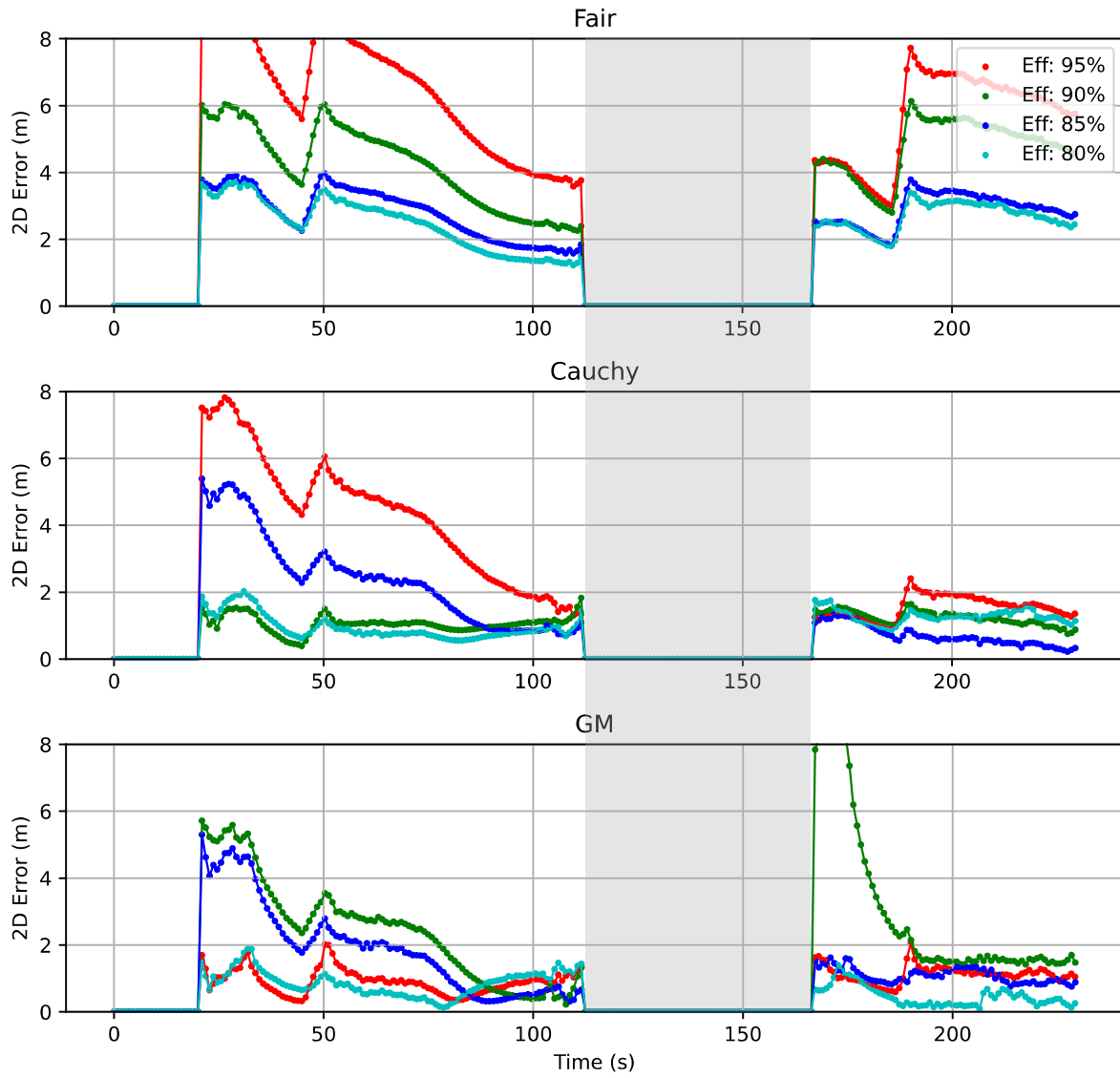


Figure D.11: Plot of 2D error of sequence #2 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

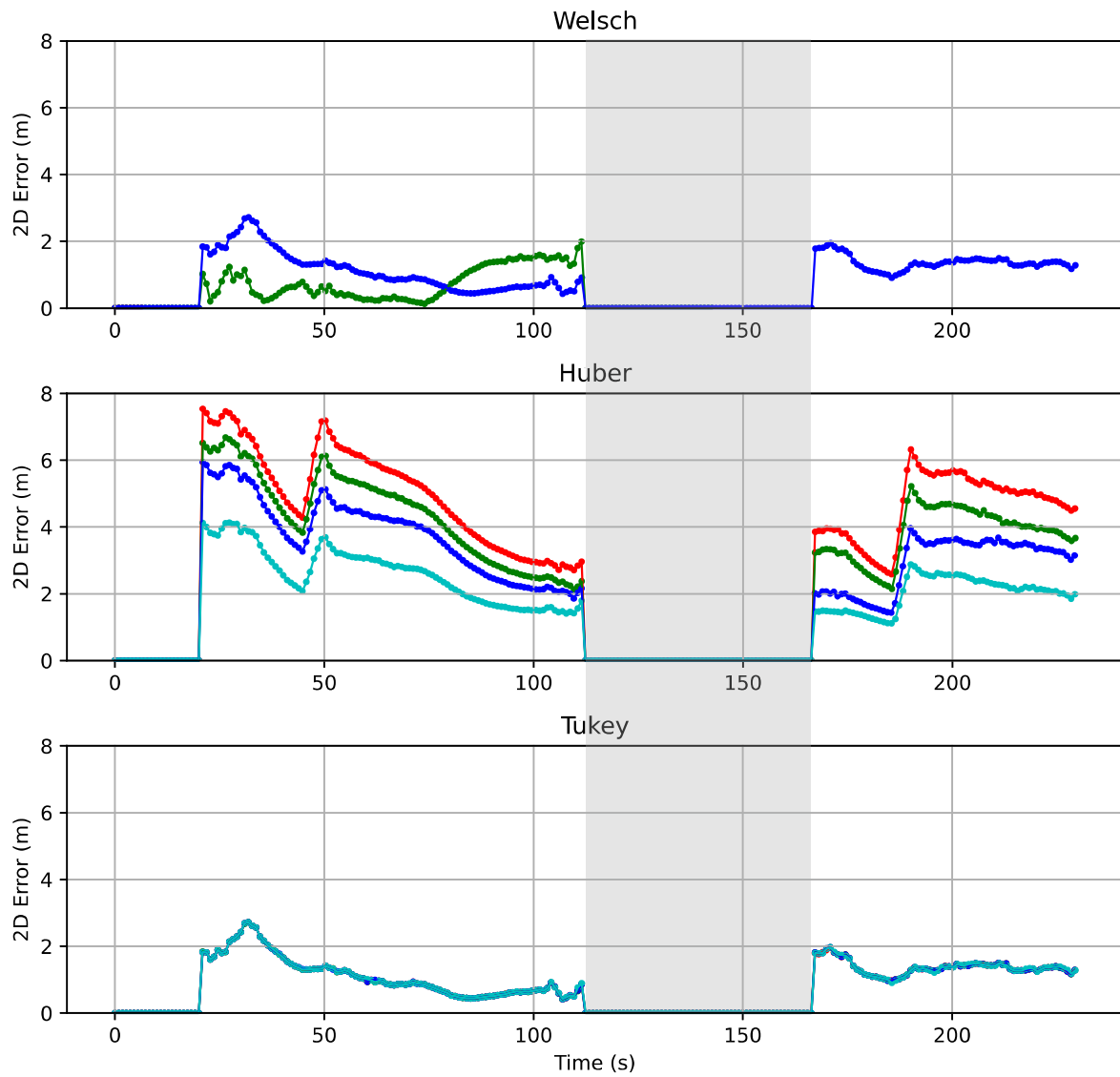


Figure D.11: (Continued) Plot of 2D error of sequence #2 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

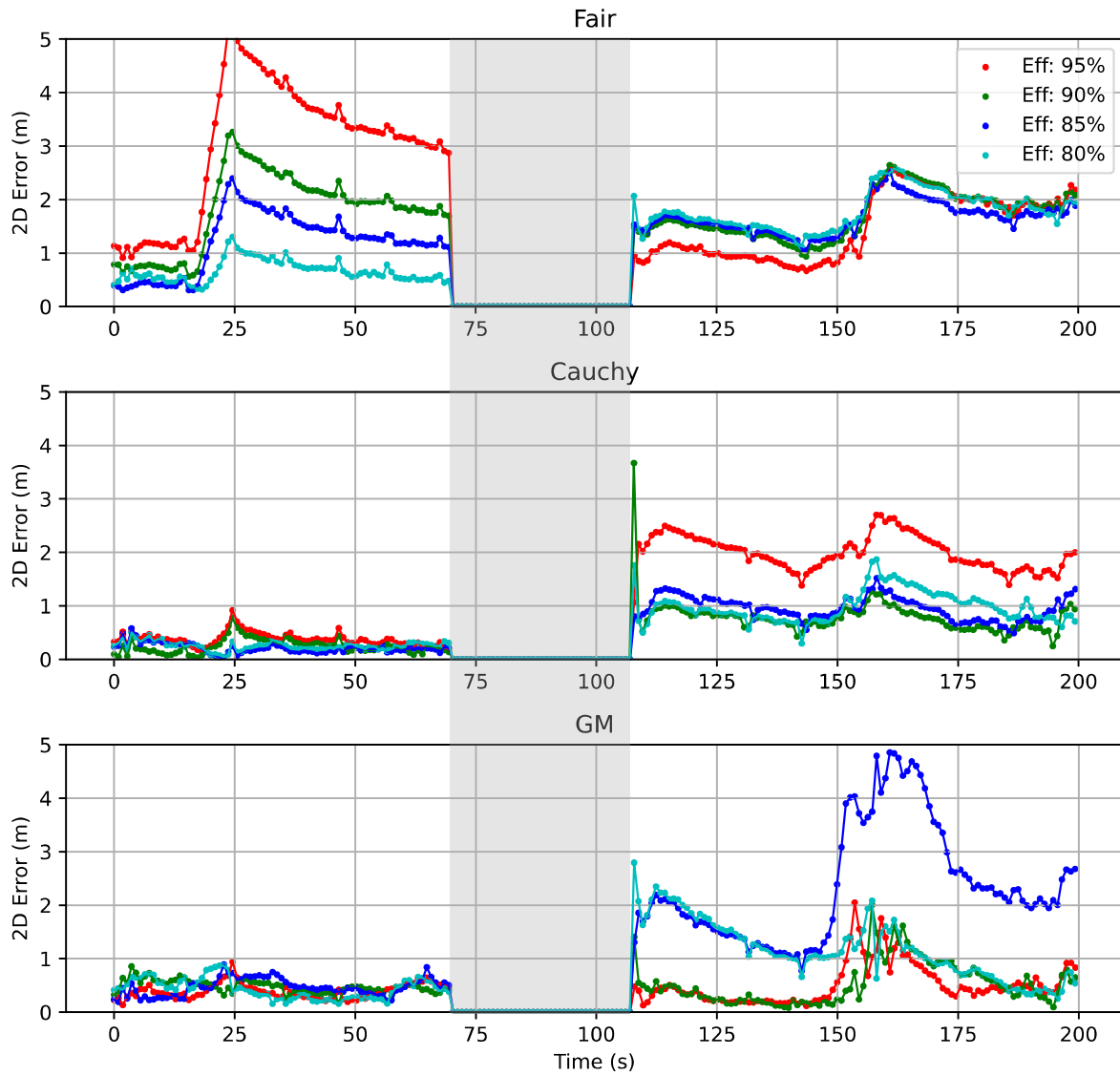


Figure D.12: Plot of 2D error of sequence #3 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

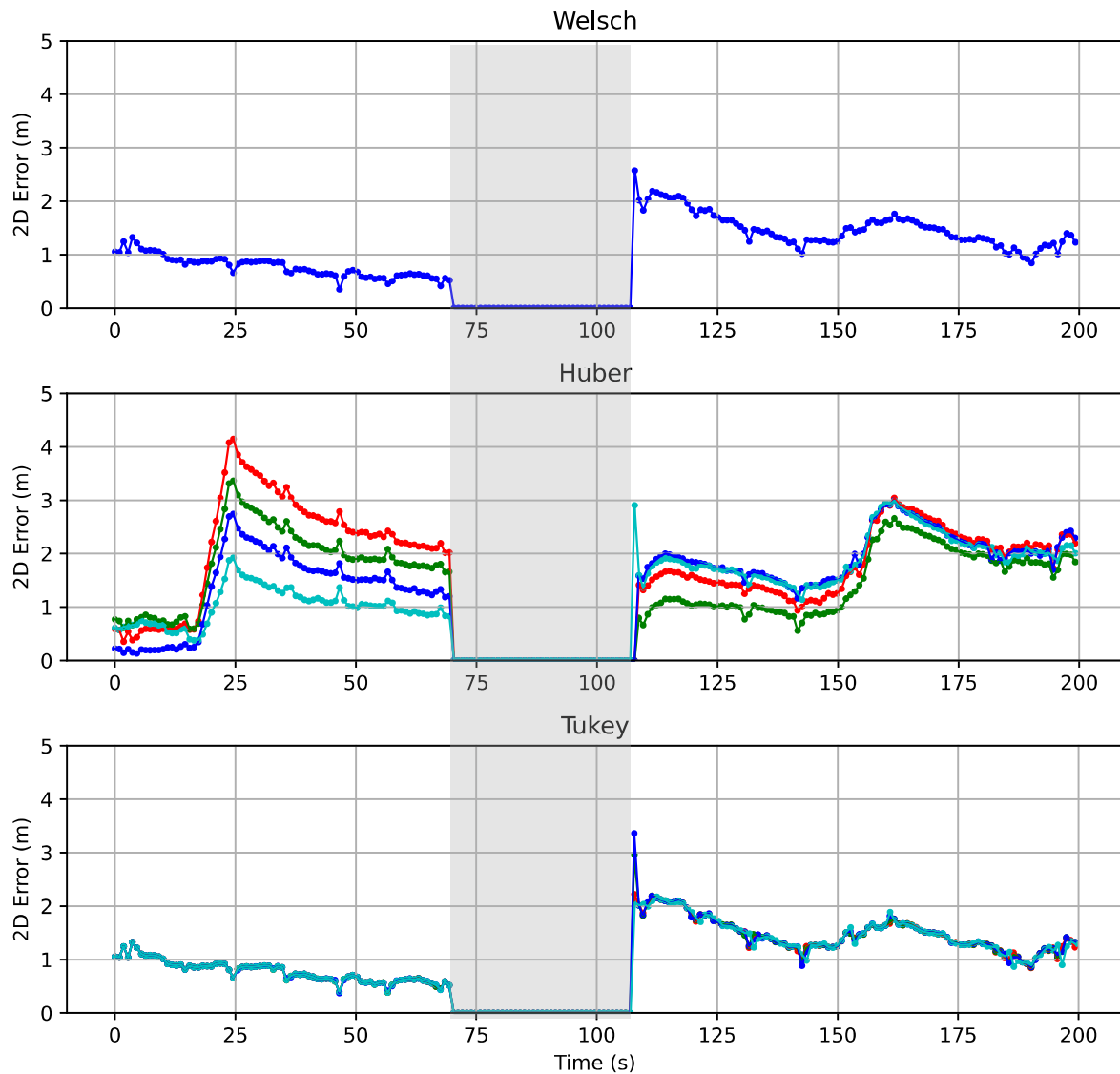


Figure D.12: (Continued) Plot of 2D error of sequence #3 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

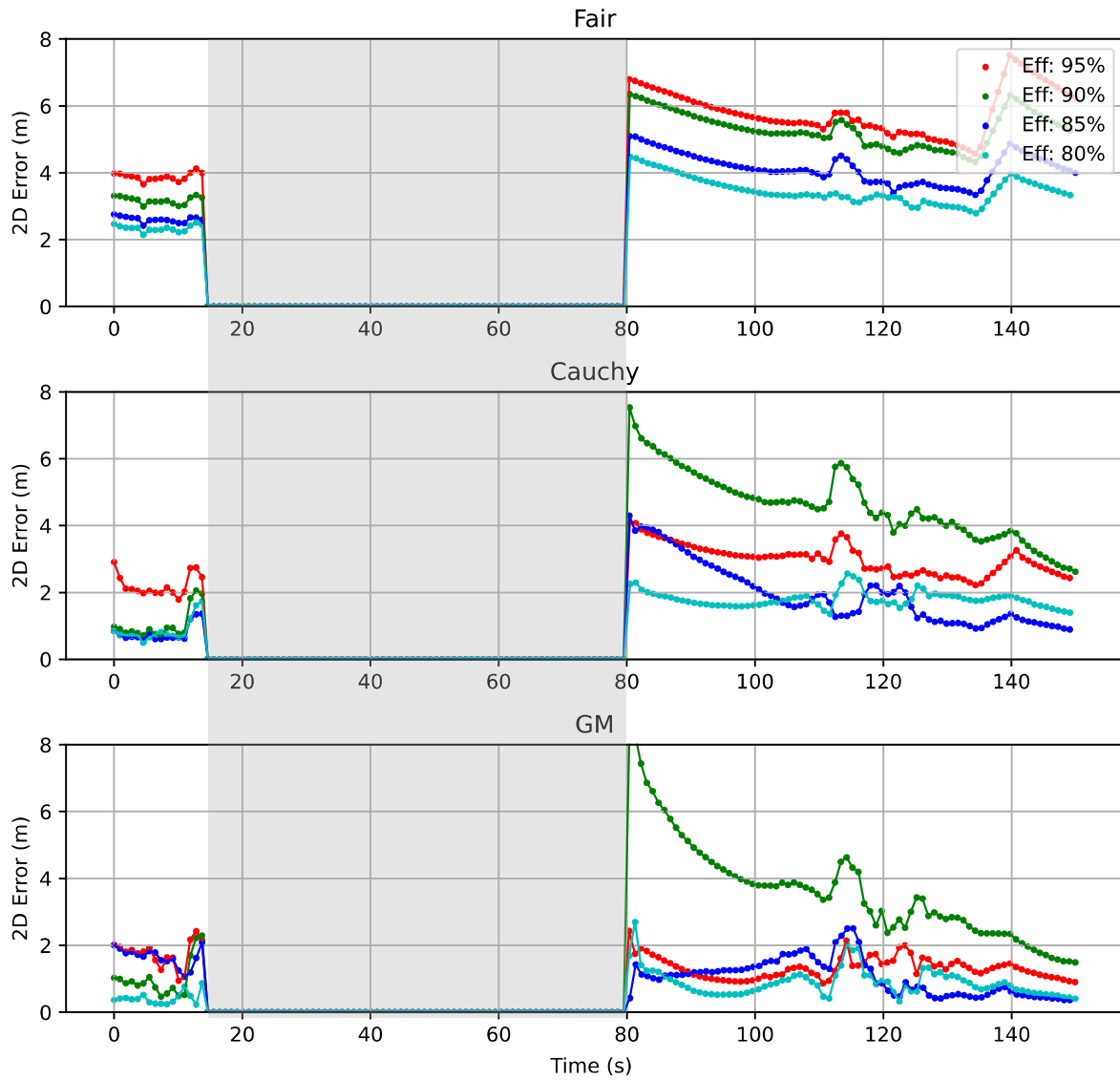


Figure D.13: Plot of 2D error of sequence #4 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

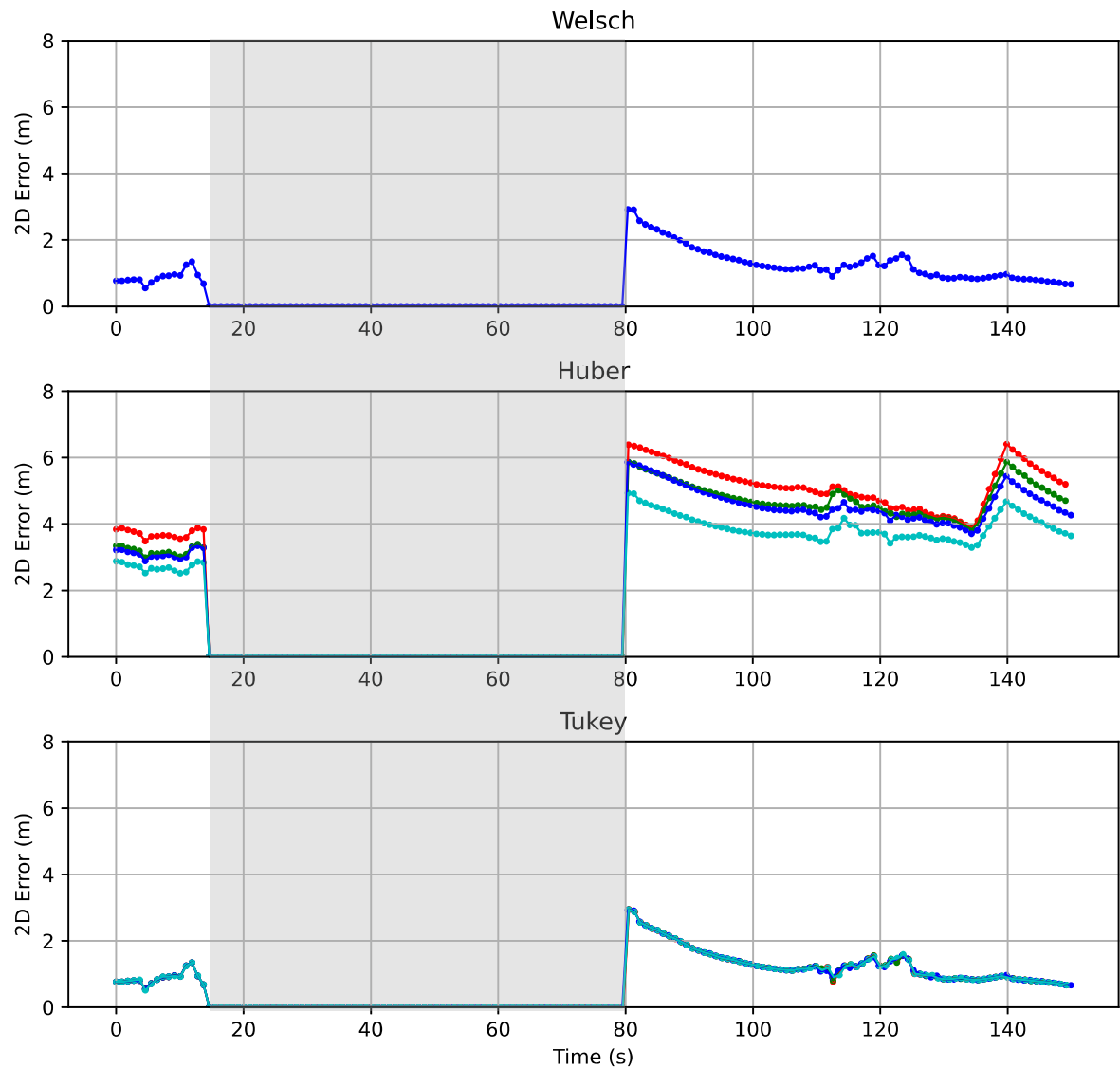


Figure D.13: (Continued) Plot of 2D error of sequence #4 in open-sky areas in kernel/efficiency view. No ground-truth data is available to calculate errors during the time interval marked in gray, where the reported error is equal to zero.

APPENDIX E

Supplementary Materials for Learning-based Methods

E.1	Implementation and Training Protocols	249
E.1.1	Data Preparation	249
E.1.2	Training Protocols	253
E.2	Extensive Experimental Analysis	254
E.2.1	NLOS Classification	255
E.2.2	Pseudorange Error Prediction	255
E.2.3	Ablation Study of Proposed Deep-Learning Model	258
E.3	Supplementary Data Specifications	259

E.1 Implementation and Training Protocols

In this thesis, all learning-based models are implemented using the PyTorch software framework in Python. For classical machine learning models, I utilize their standard implementations available in PyTorch, while both reference and proposed deep-learning models are custom-implemented.

Unlike deep-learning models, which handle the NLOS classification and error prediction tasks simultaneously by minimizing a customized loss function, classical models train these tasks separately.

In the following sections, I provide an overview of all implementation details, including data preparation and training protocols.

E.1.1 Data Preparation

In Section 2.4.3, the input data have already been extracted and analyzed. To ensure efficient model training, the input data must be properly formatted to match the model's required input shape.

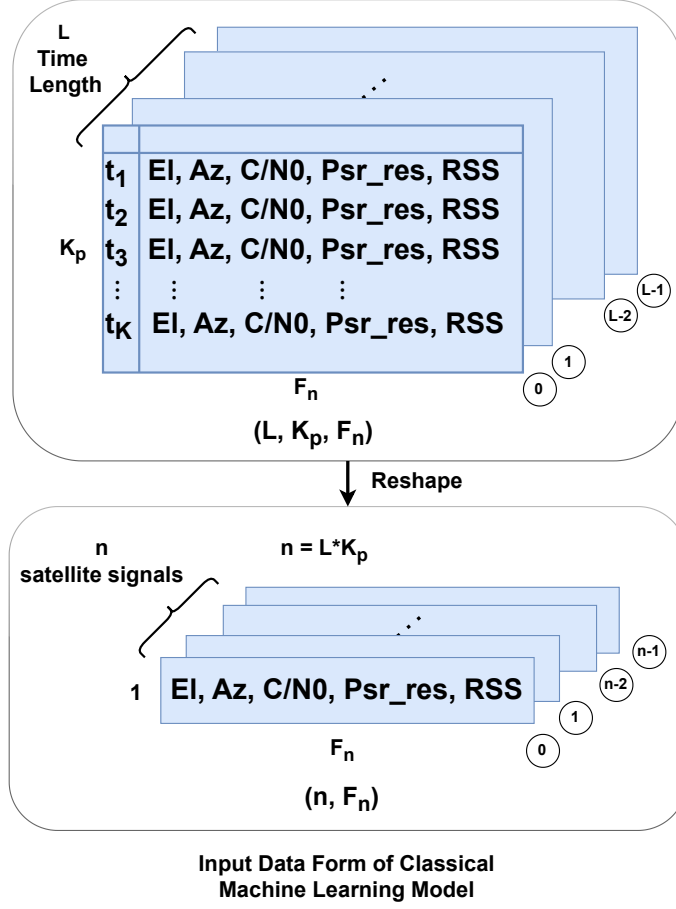


Figure E.1: Input data format for classical machine-learning models. The variable L represents the time-sequence length of the dataset, K_p denotes the number of ever-present satellites in an individual dataset, F_n indicates the number of selected features, and n is the total number of independent satellite signals.

For training classical machine learning models, the extracted data should first be reshaped into a two-dimensional format, as illustrated in Figure E.1.

However, when training deep networks, the entire dataset cannot be fed into the network at once due to computational efficiency constraints and hardware limitations. Therefore, the mini-batch gradient descent method is the most suitable approach for training deep learning models [272].

Figure E.2 illustrates the data processing and organization used to train the referenced LSTM model [90]. The left side represents a 2D batched feature tensor of the target satellite, while the right side depicts a 3D batched input data of K_C context satellites. In accordance with the referenced LSTM model architecture, both parts are synchronously fed into the

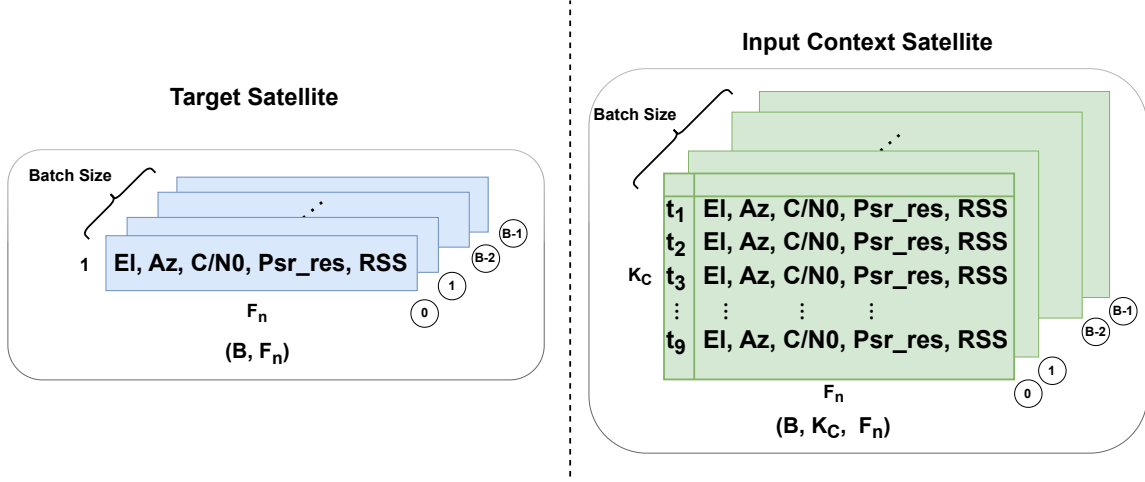


Figure E.2: Input data for the referenced LSTM model [90]. The variable B represents the batch size, T_L denotes the input time-sequence length, K_C is a hyperparameter defining the number of context satellites, and F_n indicates the number of selected features.

model to predict the visibility and pseudorange error of the target satellite, incorporating information from the context satellites.

The input to the proposed model consists of two components. The first component, illustrated in Figure E.3, is a 3D time-sequence dataset representing the target satellite, whose visibility and pseudorange error are to be predicted. The second component, also shown in Figure E.3, is a 4D context time-sequence dataset comprising multiple satellites. This dataset is constructed by concatenating several individual 3D time-sequence datasets, each formatted identically to the first component. Both input components share the same batch size and are synchronously fed into the network during training.

As analyzed in Section 2.4.3, the dataset is somewhat unbalanced, with the proportion of NLOS (Non-Line-of-Sight) signals being lower than LOS (Line-of-Sight) signals. Training on an imbalanced dataset can lead to biased results, as the model tends to favor the majority class. This occurs because the training process primarily aims to minimize the overall loss function. Since the majority class has more instances, the model is more likely to predict those labels, leading to biased classification or regression.

To mitigate this issue and improve prediction accuracy, the dataset is processed to create a balanced version. During training, both the original and balanced datasets are utilized to reduce the negative impact of the initial imbalance, ensuring a more robust and unbiased model.

In addition to data balancing, data augmentation is another effective technique used to

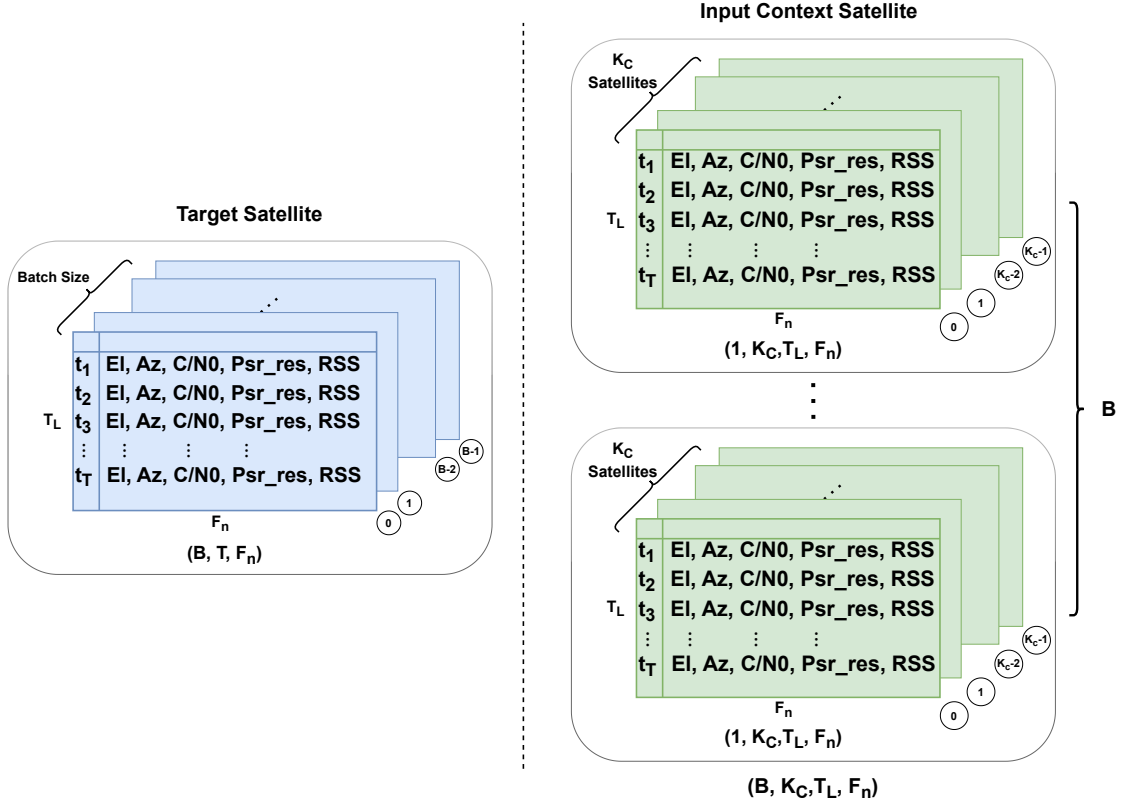


Figure E.3: Input data for context information (central input). The variable B represents the batch size, K_C is a hyperparameter defining the number of context satellites, T_L denotes the input time-sequence length, and F_n indicates the number of selected features.

enhance the performance and generalization capability of machine learning models, especially when handling unseen datasets. Data augmentation increases the size and diversity of the training dataset by generating synthetic examples from existing data. As previously mentioned, a more diverse training set helps the model achieve better performance while significantly reducing the risk of overfitting. Moreover, data augmentation can partially improve the performance of the minority class by providing additional representative samples.

Given the nature of satellite context information, variations can arise due to different satellite selections and the sequence order of the selected satellites. Data augmentation is applied by leveraging these variations in satellite context information, further enhancing model robustness and adaptability.

E.1.2 Training Protocols

E.1.2.1 Support Vector Machine (SVM)

Table E.1: Hyper Parameters for the SVM.

Penalty parameter C	Kernel
1.0	RBF

Table E.1 summarizes the penalty parameter C and the selected kernel function used in the SVM model. The penalty parameter controls the trade-off between minimizing training errors and ensuring good generalization. A smaller C allows for more misclassifications during training but can improve generalization on test data, whereas a larger C reduces training errors by creating a narrower margin, which may lead to overfitting.

The kernel function defines how the SVM transforms input data into a higher-dimensional space, with common types including linear, polynomial, and radial basis function (RBF). The choice of the kernel function depends on the nature of the data and the specific problem being addressed.

The RBF kernel is formulated in the following function as

$$\mathcal{K}_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \quad (\text{E.1})$$

where the vector $\mathbf{x}$ denotes the data point. The hyperparameter γ controls the influence of a single training example. A larger γ value leads to a more sensitive decision boundary that closely follows the training data, while a smaller γ results in a smoother decision boundary that generalizes better to unseen data.

E.1.2.2 XGBoost

Table E.2: Hyperparameters for XGBoost.

	n_estimators	max. depth	learning rate	subsampling ratio
classifier	200	None	1.0	0.8
regressor	80	None	1.0	0.8

Table E.2 presents the key hyperparameters used in the XGBoost models for both classification and regression tasks:

- **n_estimators:** determines the number of decision trees in the XGBoost model. A higher number of trees can improve performance, but increases computation time and memory usage.

- **max depth:** Specifies the maximum depth of each decision tree. Deeper trees capture more complex patterns, but may lead to overfitting.
- **learning rate:** Controls the step size at each iteration of the gradient boosting process. A smaller learning rate improves accuracy but requires more iterations to converge.
- **subsampling ratio:** Defines the fraction of training instances used to train each individual tree. Lower values help reduce overfitting, but may also limit the model’s ability to learn complex patterns.

E.1.2.3 Proposed Deep Architecture

Table E.3: Hyperparameters for the proposed deep neural network.

batch size B	sequence length T_L	num. contextual satellites K_C	learning rate l_r	num. header H_n	optimizer
32/64/128	10	9	0.001	4	Adam [260]

Table E.3 presents the hyperparameters used in the proposed deep neural network. Here, the parameters B represent the batch size, T_L denotes the input time-sequence length, and K_C indicates the number of context satellites. Additionally, the parameter H_n specifies the number of heads in the multi-head attention mechanism, and l_r is the initial learning rate.

During training, the batch size may be adjusted to achieve optimal performance. Furthermore, the learning rate is not fixed throughout the training process. Instead, a learning rate scheduler is applied to dynamically adjust the learning rate, ensuring that the loss function converges accurately and efficiently toward its minimum value over successive training steps.

E.2 Extensive Experimental Analysis

To thoroughly evaluate the effectiveness of the proposed model, extensive experiments are conducted using various datasets and configurations. This section presents a supplementary analysis of the model’s performance, comparing it with classical and state-of-the-art approaches.

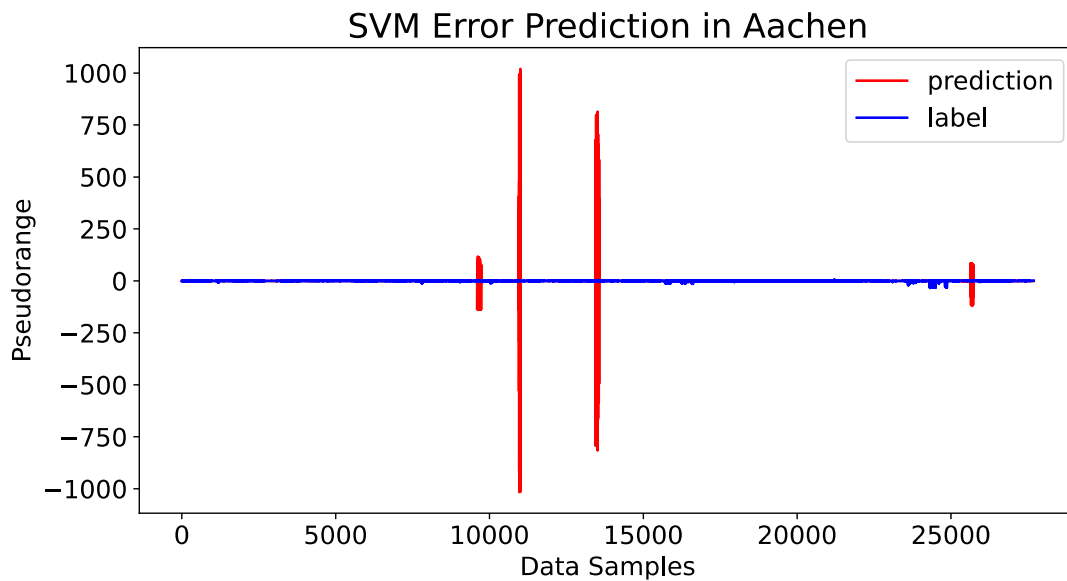
E.2.1 NLOS Classification

E.2.2 Pseudorange Error Prediction

E.2.2.1 Training and Test on Aachen Dataset

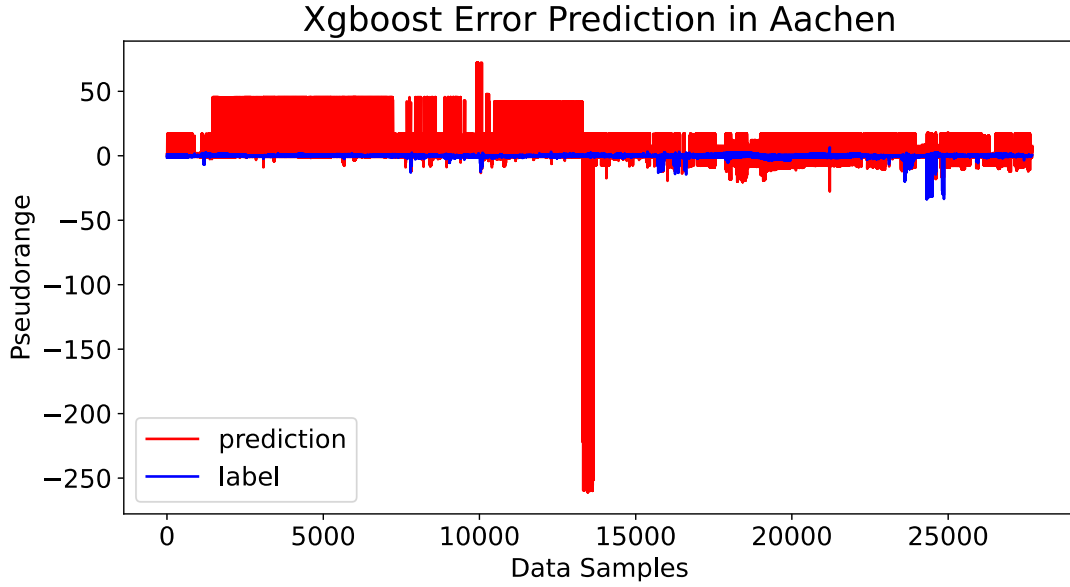
Figure E.4 illustrates the limitations of classical machine learning methods in predicting pseudorange error. Notably, XGBoost produces a significant number of predictions that deviate considerably from the actual label values, while the SVM regressor, despite achieving better overall performance than XGBoost, exhibits extreme prediction peaks that diverge drastically from the true pseudorange error.

The suboptimal performance of classical regression models can be attributed not only to algorithmic limitations but also to the nature of the pseudorange error data in Aachen. Due to enhanced preprocessing and optimization of raw GNSS data, vehicle positions are determined with greater accuracy, leading to generally lower pseudorange error values. Furthermore, the acquisition of pseudorange error labels is highly sensitive to sampling locations and receiver types. As a result, as depicted in Figure E.4, most ground-truth pseudorange errors are close to 0 m.



(a) Error prediction results using SVM with 5-feature input.

Figure E.4: Error Prediction Results of Classical Machine Learning Regressor in Aachen.



(b) Error prediction results using XGBoost with 5-feature input.

Figure E.4: (Continued) Error Prediction Results of Classical Machine Learning Regressor in Aachen.**Table E.4:** Pseudorange error prediction Average L1 loss in Aachen.

Model	Average L1 Loss
XGBoost	6.707
SVM	2.218
DL [90]	0.541
DL (ours, 5-feature)	0.597
DL (ours, 6-feature)	0.565

Besides, the mean absolute value of pseudorange error is 0.58 m, and 87.45 % of the ground-truth values falling within the range $[-1, 1]$ meters. When error magnitudes are small, noise becomes a dominant factor, significantly impacting model prediction accuracy. Beyond noise interference, both XGBoost and SVM exhibit a common limitation of classical models: sensitivity to minority labels with large absolute values.

Performance metrics in Table E.4 further highlight these limitations. Both XGBoost and SVM regressors yield an average L1 loss exceeding 2 m, whereas deep learning models significantly outperform them, achieving an average L1 loss below 0.6 m. Figure E.5 shows that all three models tend to predict values near zero, aligning with the previously observed low mean absolute pseudorange error.

However, classical models struggle to track minority labels with large absolute values, whereas deep learning models demonstrate improved performance. Notably, my proposed

model, trained with the **6-feature** configuration, achieves the lowest L1 loss and best tracks minority labels, further validating its superiority in pseudorange error regression tasks.

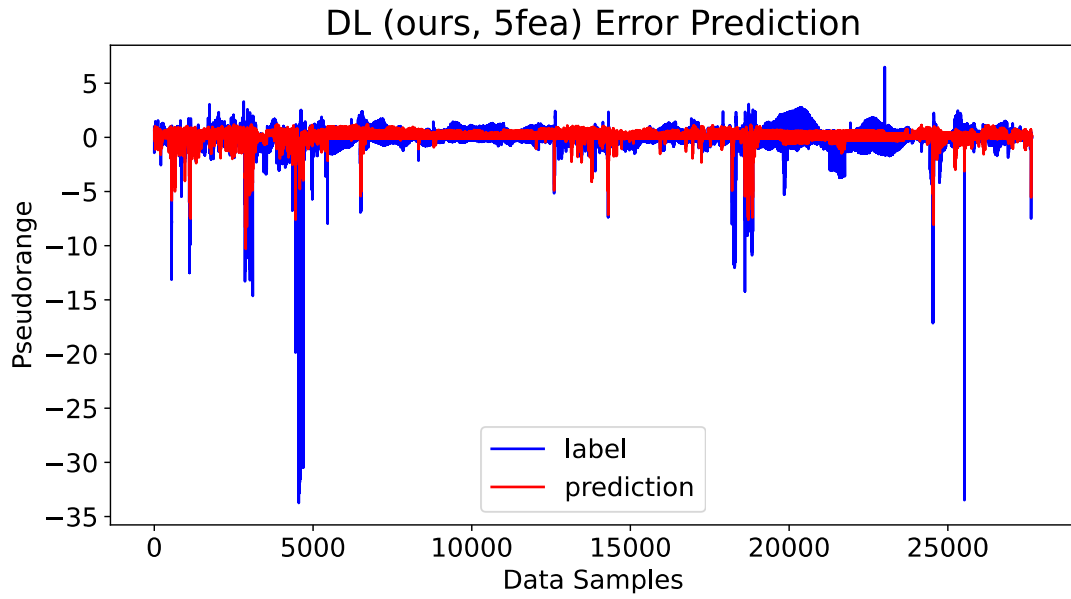
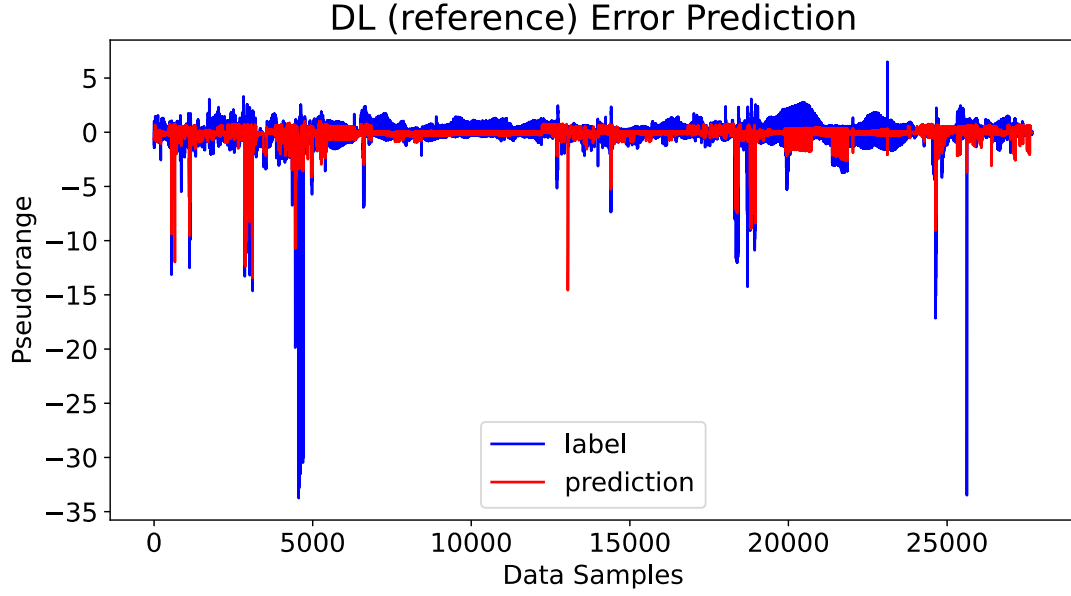
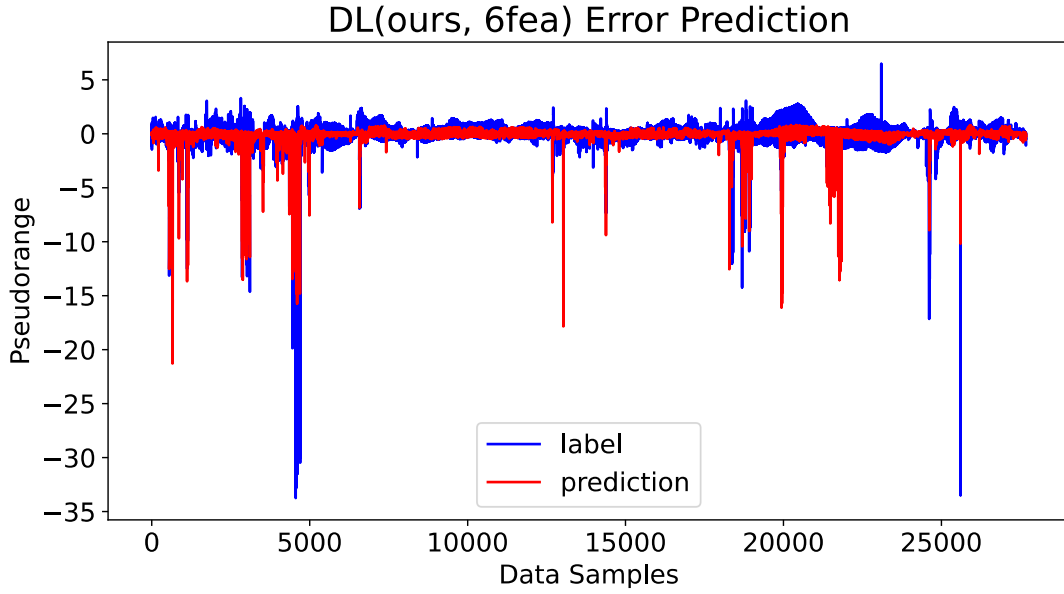


Figure E.5: Error prediction results of the deep learning regressor in Aachen.



(c) Error prediction results using the proposed deep network with 6-feature input.

Figure E.5: (Continued) Error prediction results of deep learning regressor in Aachen.

E.2.3 Ablation Study of Proposed Deep-Learning Model

In deep learning, an ablation study systematically removes specific components of a model to analyze their individual contributions to overall performance. In this section, I conduct ablation experiments on the proposed model to evaluate the impact of various components, including the transformer-based attention mechanism and the Bi-LSTM layer. The results of these experiments are presented in Table E.5 and Table E.6.

Table E.5: Performance metrics of the ablation study on the classification task for the proposed method.

Model		Precision	Recall	F1-Score	Accuracy
w/o. contextual info.	LOS	0.87	0.87	0.87	0.83
	NLOS	0.74	0.74	0.74	
w/o. attention mechanism	LOS	0.87	0.88	0.88	0.83
	NLOS	0.75	0.73	0.74	
w/o. Bi-LSTM	LOS	0.88	0.88	0.88	0.83
	NLOS	0.74	0.74	0.74	
full model (ours)	LOS	0.90	0.91	0.90	0.87
	NLOS	0.80	0.79	0.80	

As expected, incorporating spatial information via the attention mechanism enhances

Table E.6: Performance metrics of the ablation study on the regression task for the proposed method.

Model	Average L1 Loss
w/o. contextual info.	6.564
w/o. attention mechanism	6.204
w/o. Bi-LSTM	6.441
full model (ours)	5.517

both precision and recall for NLOS classification, even when the dataset contains a higher proportion of LOS observations. In my proposed architecture, the transformer-based attention mechanism processes the contextual information from multiple satellites. When this component is removed, the model is reduced to a single-satellite prediction framework without the transformer architecture, making predictions based solely on a single satellite’s time-series data. In this case, the F1-score for NLOS classification decreases from 0.80 m to 0.74 m, while the LOS F1-score drops from 0.90 m to 0.87 m. Additionally, the average L1 loss increases from 5.517 m to 6.527 m, demonstrating that contextual information plays a crucial role in improving prediction accuracy.

The Bi-LSTM layer is used to structure the extracted contextual satellite information while mitigating the bias introduced by the order of the satellite sequences before data concatenation. When Bi-LSTM is replaced with a standard LSTM, the recall for NLOS detection declines, further validating the superiority of Bi-LSTM. As established in Cheng et al. [273], Bi-LSTM enhances model capacity by preserving information from both past and future time steps, which is especially beneficial when contextual information is critical.

E.3 Supplementary Data Specifications

The following figures provide key data specifications of other received satellites in the HK dataset. The figures from the HK dataset provide key specifications for received satellites from the QZSS, GPS, GLONASS, Galileo, and BeiDou constellations. The data analysis focuses on key signal characteristics for both Line-of-Sight (LOS) and Non-Line-of-Sight (NLOS) signals, including Carrier-to-Noise Density (CN/0), Elevation, and Pseudorange Error.

The Carrier-to-Noise Density Ratio (CN/0) demonstrates a clear distinction between LOS and NLOS signals. LOS signals generally exhibit higher CN/0 values, mostly within the 30–50 dB-Hz range, indicating stronger and more reliable signal reception. In contrast, NLOS signals display significantly lower CN/0 values, typically ranging from 10 to 30 dB-Hz, reflecting weaker signal strength and increased susceptibility to noise and multipath interference.

Pseudorange error distributions further highlight the challenges posed by NLOS conditions. For LOS signals, pseudorange errors remain relatively small, mostly within the range of (-5,

5) meters, indicating minimal deviation from the true satellite-receiver distance. However, for NLOS signals, pseudorange errors exhibit much larger variations, often exceeding ± 200 meters. This demonstrates the impact of multipath propagation and obstructions in urban environments, which introduce substantial errors in satellite positioning.

The elevation angle of received signals also shows notable differences between LOS and NLOS conditions. LOS signals tend to have higher elevation angles, typically concentrated between 30° and 80° , suggesting that these signals are received directly from satellites positioned overhead. In contrast, NLOS signals typically exhibit lower elevation angles, often below 40° , indicating that they are more susceptible to reflections from buildings and other obstacles, resulting in degraded signal quality.

When comparing the performance among different satellite constellations, GPS and Galileo signals exhibit relatively stable CN/0 values in both LOS and NLOS conditions, indicating strong and consistent signal reception. On the other hand, BeiDou and GLONASS signals show a wider spread of CN/0 values, suggesting variability in signal strength. QZSS, being a regional system, provides fewer signals but generally exhibits higher CN/0 values in LOS conditions, making it a reliable option in areas where it is available.

Overall, these findings highlight the importance of contextual satellite information in predicting signal quality and pseudorange errors, especially in urban environments where NLOS conditions introduce significant errors. Incorporating deep learning models that leverage multi-satellite contextual data can help mitigate these effects by capturing spatial and temporal dependencies, ultimately improving positioning accuracy in challenging environments.

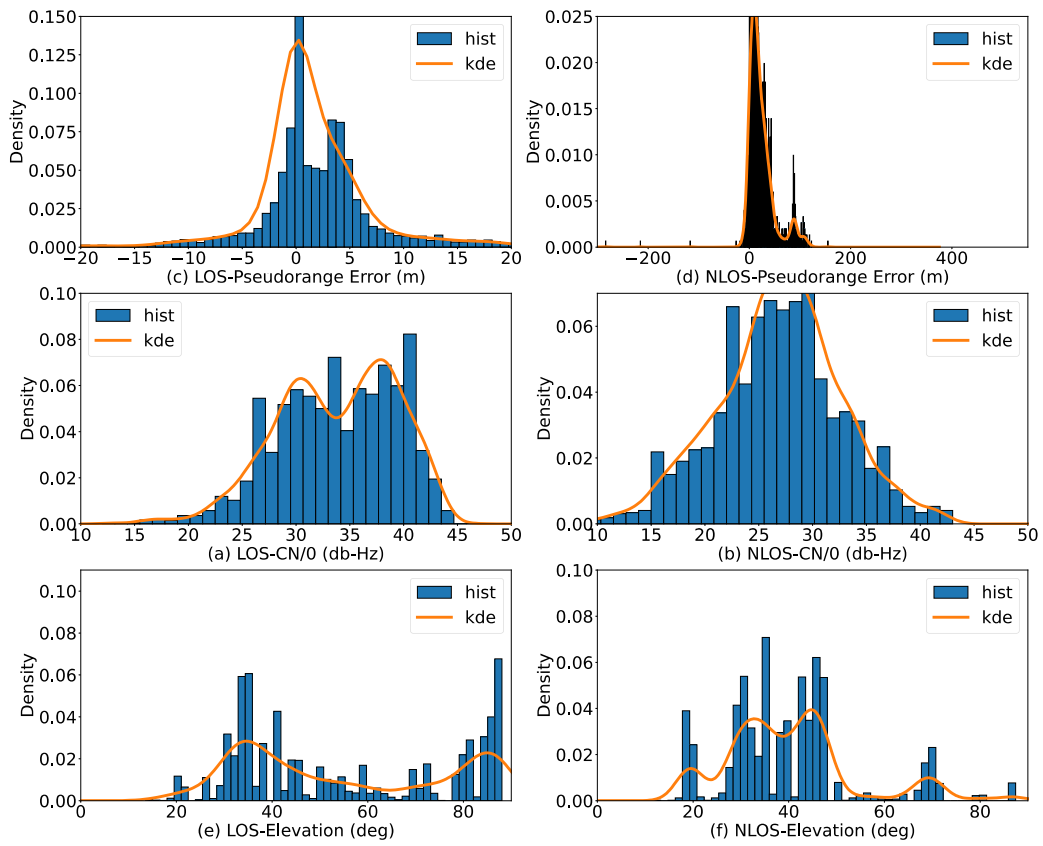


Figure E.6: Data specification of Galileo (Europe) measurements in Hong Kong.

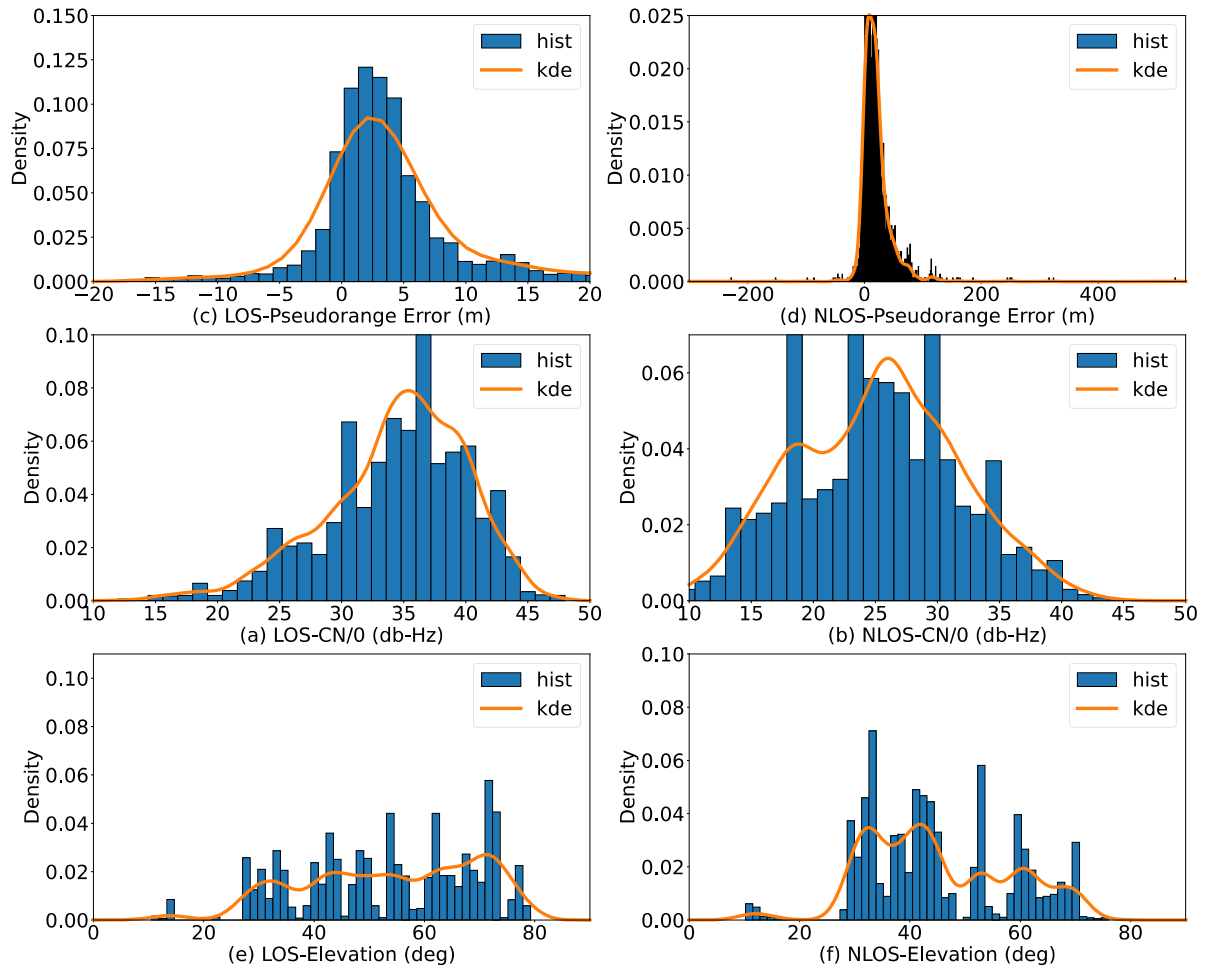


Figure E.7: Data specification of GLONASS (Russia) measurements in Hong Kong.

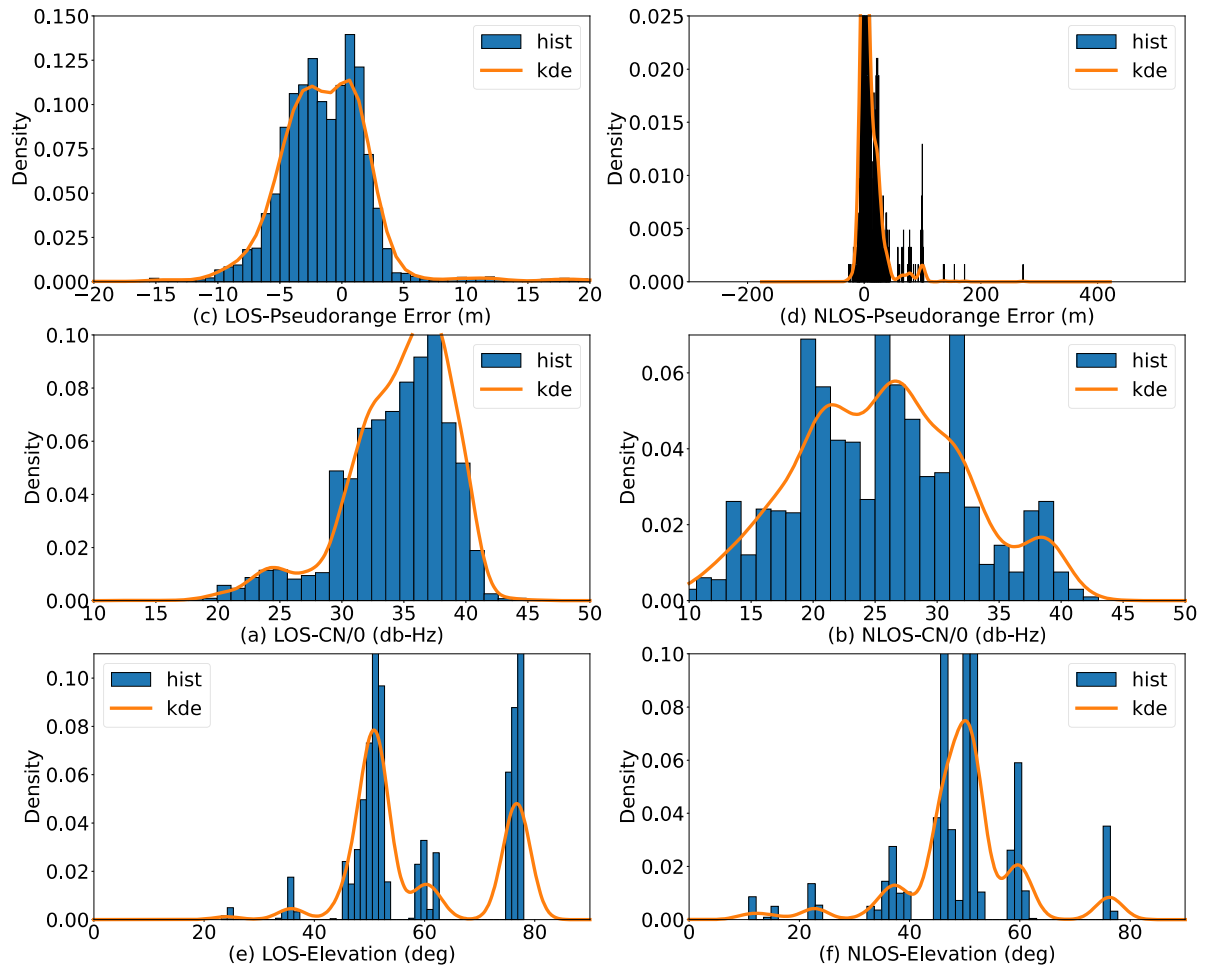


Figure E.8: Data specification of QZSS (Japan) measurements in Hong Kong.

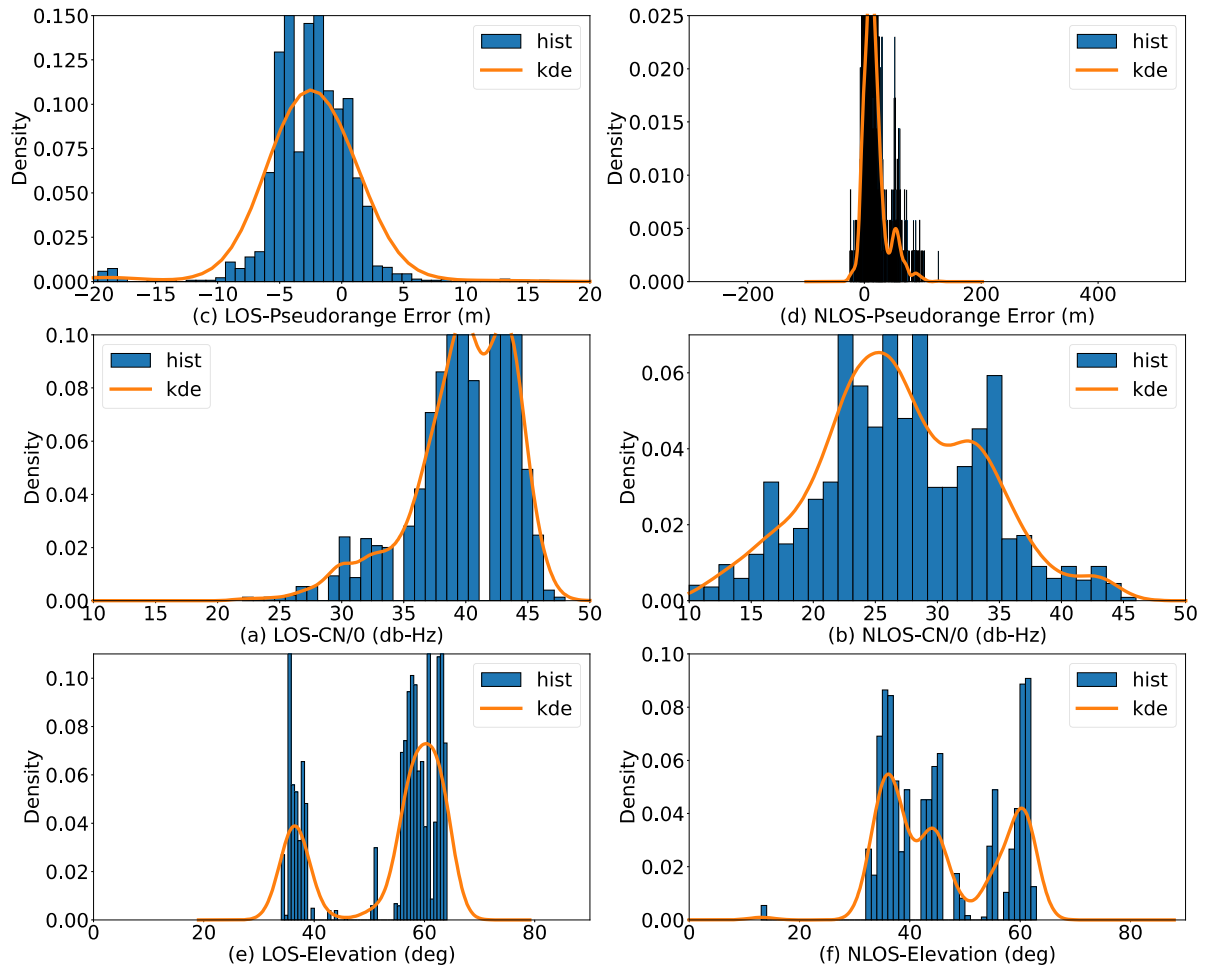


Figure E.9: Data specification of Beidou (China) measurements in Hong Kong.

List of Figures

1.1	Demonstrations of failures in vehicle localization approaches due to measurement outliers.	4
1.2	Scheme of common approaches to handle measurement outliers toward robust vehicle localization.	5
2.1	Test Vehicles.	21
2.2	Application I: test run in the Dortmund-Ems water canal.	23
2.3	Application II: test run in the cities of Aachen, Düsseldorf, Cologne, and on the highways.	24
2.4	Measurement campaigns in Hong Kong.	27
2.5	GPS data specification from Hong Kong.	28
2.6	GPS data specification from Aachen.	29
3.1	Continuous-time state estimation with asynchronous measurements. . . .	44
3.2	A general time-centric factor graph.	44
3.3	Flowchart of the UKF-based GNSS/INS fusion following.	48
3.4	Factor graph proposed in Application I.	48
3.5	Flowchart of the system for tightly coupled GNSS/INS integration with continuous-time FGO in Application I.	50
3.6	Trajectory plot by crossing the bridges 10 and 9a in Dortmund-Ems canal. .	51
3.7	Estimation Results of Application I.	54
3.8	FGO with different motion prior models of Application I.	55
3.9	A general graph of loose coupling in gnssFGO.	57
3.10	A general graph of tight coupling in gnssFGO.	59
3.11	System overview showing all data interfaces and factor types by implementing Algorithm 1.	60
3.12	Demonstration of multisensor fusion for vehicle localization of test sequence C02 in the city of Cologne	64
3.13	Trajectory plot (700 s - 1400 s) in urban areas in Aachen.	67
3.14	Trajectory plot (450 s - 1350 s) in challenging areas in Düsseldorf.	70
3.15	Trajectory plot near the central station of Cologne (Seq. C01).	72
4.1	Plot of loss values using different kernel functions at $E_{ff} = 95\%$	80
4.2	Plot of influence values using different kernel functions at $E_{ff} = 95\%$. . .	80
4.3	Analytical quantities of the Tukey estimator.	81
4.4	Plot of all kernels' influence function and unwhitened residual weights. . .	85
4.5	Plot of 2D error of sequence #1 in open-sky areas using different M-estimators. .	88
4.6	Plot of 2D error of sequence #2 in urban areas using different M-estimators. .	89

4.7	Plot of 2D error of sequence #3 in light-urban areas using different M-estimators.	90
4.8	Plot of 2D error of sequence #4 in light-urban areas using different M-estimators.	91
5.1	Proposed dataset generation process.	96
5.2	Generated lidar Map in Aachen City.	96
5.3	Sky plots from two scenes: a) urban tree-rich area, b) urban. The received GPS satellites are marked in orange.	97
5.4	lidar point clouds collected in different time intervals.	98
5.5	Classical Machine Learning Models.	102
5.6	Referenced LSTM Architecture.	103
5.7	Proposed Neural Network Architecture.	104
5.8	Feature Permutation Importance of different models for NLOS classification.	109
5.9	Predicted and ground-truth pseudorange errors with SVM and XGBoost.	111
5.10	Predicted and ground-truth pseudorange errors with the baseline model and the proposed network.	112
5.11	Feature Permutation Importance of different models for pseudorange error prediction.	114
5.12	Vehicle Localization in Urban Area in Düsseldorf with NLOS Exclusion.	116
6.1	Example of resilient state estimation system.	127
A.1	Example of a 2-dimensional Gaussian distribution.	133
A.2	Projection of 2-D Gaussian into a new coordinate y , where the covariance matrix is decomposed into two independent eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$	134
A.3	Illustration of the iterative state update using gradient descent (SD), Gauss-Newton (GN), and Powell's dogleg (DL) methods.	147
A.4	Schema of the recursive Bayes filter.	148
A.5	Toy examples of three graphical models.	159
A.6	Jacobian and Hessian matrices of the toy example in Figure A.5b.	164
A.7	Comparison of state estimation techniques to a 1-D mass-spring system.	170
A.8	Coordinate frames used in this thesis.	171
A.9	Manifold, tangent space, and their transformations.	175
A.10	GNSS delays and interference.	183
A.11	Lidar map before and after fusing the loop-closure constraints in SLAM.	187
A.12	Venn diagram of learning mechanisms to build data-driven models.	191
A.13	Basic concept of XGBoost following.	192
A.14	Basic concept of support vector machine.	193
A.15	Concept of perceptron.	195
A.16	Basic concept of MLP.	196
A.17	LSTM Architecture.	197

A.18	Multi-head attention architecture.	200
B.1	Example of the transformation from <i>ECEF</i> to <i>WGS84</i>	204
C.1	Trajectories of loosely coupled fusion using different state and optimization frequencies.	222
C.2	Trajectories with different noise modeling while crossing bridge 9a.	225
C.3	Estimated GNSS receiver clock bias of Seq. C01 without near-zero-velocity detection.	227
C.4	Coordinates in the ENU frame of Seq. C02.	228
C.5	Examples of lidar odometry degradation in three scenarios: a) unstructured feature-less area, b) high-speed scenario, and c) long tunnel 400 m.	229
C.6	Computation time with different configurations of Seq. DUS.	230
C.7	Histogram of whitened state errors of GP models in an open-sky area.	232
D.1	Plot of loss values using different kernel functions at $E_{ff} = 90\%$	234
D.2	Plot of loss values using different kernel functions at $E_{ff} = 85\%$	234
D.3	Plot of loss values using different kernel functions at $E_{ff} = 80\%$	235
D.4	Fair Influence at four levels of efficiency through tuning.	236
D.5	Cauchy Influence function at four levels of efficiency through tuning.	236
D.6	Geman-McClure Influence function at four levels of efficiency.	238
D.7	Welsch Influence function at four levels of efficiency through tuning.	238
D.8	Huber influence function curves at four levels of efficiency.	240
D.9	Tukey influence function at four levels.	240
D.10	Plot of 2D error of sequence #1 in open-sky areas in kernel/efficiency view.	241
D.10	(Continued) Plot of 2D error of sequence #1 in open-sky areas in kernel/efficiency view.	242
D.11	Plot of 2D error of sequence #2 in open-sky areas in kernel/efficiency view.	243
D.11	(Continued) Plot of 2D error of sequence #2 in open-sky areas in kernel/efficiency view.	244
D.12	Plot of 2D error of sequence #3 in open-sky areas in kernel/efficiency view.	245
D.12	(Continued) Plot of 2D error of sequence #3 in open-sky areas in kernel/efficiency view.	246
D.13	Plot of 2D error of sequence #4 in open-sky areas in kernel/efficiency view.	247
D.13	(Continued) Plot of 2D error of sequence #4 in open-sky areas in kernel/efficiency view.	248
E.1	Input data format for classical machine-learning models.	250
E.2	Input data for the referenced LSTM model.	251
E.3	Input data for context information (central input).	252
E.4	Error Prediction Results of Classical Machine Learning Regressor in Aachen.	255
E.4	(Continued) Error Prediction Results of Classical Machine Learning Regressor in Aachen.	256

List of Figures

E.5	Error prediction results of the deep learning regressor in Aachen.	257
E.5	(Continued) Error prediction results of deep learning regressor in Aachen.	258
E.6	Data specification of Galileo (Europe) measurements in Hong Kong.	261
E.7	Data specification of GLONASS (Russia) measurements in Hong Kong. . .	262
E.8	Data specification of QZSS (Japan) measurements in Hong Kong.	263
E.9	Data specification of Beidou (China) measurements in Hong Kong.	264

List of Tables

2.1	Specification of measurement campaign in Application II.	24
2.2	Specification of Dataset I.	25
2.3	Test sequences definition.	26
2.4	Details of Dataset 3.	27
3.1	State Estimation Performance of Application I.	52
3.2	State Estimation Performance with Linear Interpolated States on GNSS Delay in Application I.	56
3.3	General Trajectory Estimation Metrics in Application II.	62
4.1	Robust kernels used in this thesis with their loss function $\rho(\cdot)$, influence function $\Psi(\cdot)$, and weighting function $w(\cdot)$	79
4.2	Sequence definition containing four scenarios in the dataset collected in Aachen.	83
4.3	Configured tuning parameter c of six M-estimators at four efficiencies. . .	83
4.4	Mean 2-D error of all sequences is presented. Values marked in red indicate the worst accuracy, while those in green highlight the best results.	84
5.1	Performance Metrics using HK Dataset.	107
5.2	Performance Metrics using AC Dataset.	108
5.3	Cross-Dataset Performance Metrics.	108
5.4	Result of NLOS classification with out-of-distribution data from the measurement campaign in Düsseldorf.	115
C.1	Initial Uncertainty Parameterization in Application I.	220
C.2	Noise Parameters in Application I.	220
C.3	Elapsed Computation Time in Application I.	223
C.4	Evaluation of the Fixed-Lag Size in Application I.	224
C.5	Estimation Performance of Seq. DUS using Different Solver Configurations. .	230
E.1	Hyper Parameters for the SVM.	253
E.2	Hyperparameters for XGBoost.	253
E.3	Hyperparameters for the proposed deep neural network.	254
E.4	Pseudorange error prediction Average L1 loss in Aachen.	256
E.5	Performance metrics of the ablation study on the classification task for the proposed method.	258
E.6	Performance metrics of the ablation study on the regression task for the proposed method.	259

List of Abbreviations

Abbreviation	Description
ATE	Absolute Trajectory Error
BDS	BeiDou Navigation Satellite System
BP	Breakdown Point (in robust statistics)
CMC	Code Minus Carrier Signal (GNSS)
CRLB	Cramér-Rao Lower Bound
CVM	Constant Velocity Model
DCM	Direction Cosine Matrix
DL	Deep Learning
DRC	Doppler Rate Consistency (GNSS)
ECEF	Earth-Centered, Earth-Fixed
EKF	Extended Kalman Filter
ELBO	Evidence Lower Bound
EM	Expectation Maximization
ENU	East-North-Up
FGO	Factor Graph Optimization
FN	False Negative
FP	False Positive
GLONASS	Global Navigation Satellite System in Russian
GMM	Gaussian Mixture Model
GN	Gauss-Newton
GNSS	Global Navigation Satellite System
GP	Gaussian Process
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
iSAM	Incremental Smoothing and Mapping
KF	Kalman Filter
LiDAR (lidar)	Light Detection and Ranging
LLH	Latitude, Longitude, Height
LM	Levenberg-Marquardt
LOS	Line of Sight
LS	Least Square
LSTM	Long Short-Term Memory
LTI-SDE	Linear Time-Invariant Stochastic Differential Equation
LTV-SDE	Linear Time-Varying Stochastic Differential Equation

Abbreviation	Description
M-Estimation	Maximum-Likelihood-Type Estimation
MAE	Mean Absolute Error
MAP	Maximum a Posteriori
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MRF	Markov Random Field
MSE	Mean Squared Error
NED	North-East-Down
NLOS	Non Line of Sight
NLS	Nonlinear Least Square
NN	Neural Network
PPS	Pulse per Second
PVA	Position, Velocity, and Attitude
PVT	Position, Velocity, and Time
RANSAC	Random Sample Consensus
RNN	Recurrent Neural Network
RPE	Relative Pose Error
RSS	Root-Sum-Square
RTCM	Radio Technical Commission for Maritime Services (GNSS)
RTK	Real-Time Kinematic
SD	Stochastic Gradient Descent
SDE	Stochastic Differential Equation
SDP	Semi-Definite Programming
SLAM	Simultaneous Localization and Mapping
std	Standard Deviation
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UKF	Unscented Kalman Filter
UT	Unscented Transformation
WGS84	World Geodetic System 1984
WNOA	White Noise on Acceleration
WNOJ	White Noise on Jerk
XGBoost	Extreme Gradient Boosting

List of Symbols

Symbol	Description	Unit
Greek Variables		
σ	One-dimensional noise.	-
ω	Angular/Rotation rate vector.	rad/s
Σ	Covariance matrix.	-
ϱ	Gain ratio in the trust-region method.	-
Γ	Information matrix.	-
φ	Latitude.	deg
ξ	Local pose variable.	-
γ	Local state variable.	-
λ	Longitude.	deg
μ	Mean vector.	-
ν	Measurement noise vector.	-
η	Noise vector.	-
η	Normalization constant or one-dimensional noise parameter.	-
ρ	GNSS pseudorange.	m
β	Scale parameter in the inverse Wishart distribution.	-
$\Omega(\cdot)$	Scaled transition matrix for the current state while state interpolation using the Gaussian process motion priors.	-
$\Lambda(\cdot)$	Scaled transition matrix for the previous state while state interpolation using the Gaussian process motion priors.	-
α	Step size or Learning rate.	-
$\Phi(\cdot)$	Transition matrix used in Gaussian process motion priors.	-
ϖ	Velocity vector in the local frame.	-
λ_c	GNSS signal wavelength.	m
$\Phi(t)$	Orientation vector in the local frame.	-
$\rho(t)$	Position vector in the local frame.	-
$\nu(t)$	Linear velocity vector in the local frame.	-

$\nabla_{\theta} J(\theta_k)$	Gradient of the objective function with respect to the model parameters at timestamp k .	-
-------------------------------	--	---

Indices and Frames

BI	Bayesian inference.	-
$(\cdot)^b$	Body frame.	-
D	Dimension.	-
Δf	GNSS Doppler shift.	Hz
$(\cdot)^e$	Earth-centered, Earth-fixed frame.	-
ENU	East-North-Up frame.	-
$(\cdot)_n^e$	Vector/Matrix of frame n in frame e .	-
anc	Anchor point.	-
ant	Antenna index.	-
LLH	Index of latitude, longitude, and height.	-
$(\cdot)_D$	Discrete matrix or vector.	-
sat	Index of satellite.	-
$(\cdot)^w$	Local world frame.	-
NED	North-East-Down frame.	-
$(\cdot)^n$	Navigation frame.	-
k	Timestamp.	-
t	Time.	-
est	Index estimate.	-
gt	Index ground truth.	-
τ	Between or temporary timestamp.	-

Latin Variables

$\mathcal{Z}$	A set of measurement vectors.	-
$\mathcal{Y}$	A set of output vectors.	-
$\mathcal{X}$	A set of state vectors.	-
$\mathbf{a}$	Linear acceleration vector.	m/s ²
$\mathbf{b}^a$	Accelerometer bias vector.	m/s ²
V	Asymptotic variance.	-
B	Batch size.	-
$\mathbf{u}$	Control input vector.	-

Q	Covariance matrix of the dynamics or prior model.	-
R	Covariance matrix of measurement.	-
a	Equatorial radius.	m
e	Error vector.	-
c_b	GNSS receiver clock bias.	m
c_d	GNSS receiver clock drift.	m s ⁻¹
c^r	GNSS receiver clock error vector.	-
q_c	One-dimensional hyperparameter in the covariance matrix Q_c of the GP motion model.	-
g	Gravity vector.	-
b^g	Gyroscope bias vector.	rad/s ²
h	Height.	m
B	Input matrix.	-
I	GNSS ionospheric delay.	m
J	Jacobian matrix.	-
H	Jacobian matrix of the linearized measurement function $h(\cdot)$.	-
K	Kalman gain.	-
l	Landmark vector.	-
z	Measurement vector.	-
M	GNSS multipath delay.	m
C	Observation or measurement matrix.	-
y	Output vector.	-
b	Perturbation state vector.	-
p	Position vector.	-
E_{ff}	Relative efficiency.	-
r	Residual vector.	-
R	Rotation matrix.	-
F	State noise shaping matrix.	-
w	State noise vector.	-
x	State vector.	-
A	System matrix.	-
T	Homogeneous transformation matrix.	-

List of Tables

R_e	Transverse radius.	m
T	GNSS tropospheric delay.	m
$\mathbf{v}$	Velocity vector.	m/s
d	Update step in nonlinear optimization methods.	-

Operators and Functions

$bel(\cdot)$	Belief.	-
$J(\cdot)$ or J_{ind}	Cost function.	-
$\text{diag}(\mathbf{x})$	Generate a diagonal matrix using the given vector $\mathbf{x}$.	-
$\text{Exp}(\cdot)$	Exponential map of a variable from Lie group.	-
$f(\cdot)$	General function.	-
$\text{Log}(\cdot)$	Logarithmic map of a variable from Lie group.	-
$p(\cdot)$	Probabilistic density function.	-
$\text{ReLU}(\cdot)$	Rectified linear unit function for the nonlinear activation.	-
$\Psi(\cdot)$	Score function (M-Estimation).	-
$\exp(\cdot)$	Exponential map of a variable from Lie algebra.	-
$(\cdot)$	Transform group elements from $\mathbb{R}^3$ to $\mathfrak{se}(3)$.	-
$\log(\cdot)$	Logarithmic map of a variable from Lie algebra.	-
$\mathbf{h}(\cdot)$	Calculate the pseudo-measurement with the measurement function.	-
$\rho(\cdot)$	Calculate the robust noise model in the M-estimation.	-
$[\cdot]_{\times}$	Calculate the skew-symmetric matrix.	-
$\tanh(\cdot)$	Hyperbolic function for the nonlinear activation.	-
$(\cdot)^{\vee}$	Transform group elements from $\mathfrak{se}(3)$ to $\mathbb{R}^3$.	-
$w(\cdot)$	Weighting function (M-Estimation).	-

Special Variables

$\mathbb{E}$	Expectation.	-
$\mathcal{F}$	Feature set of local sensors.	-
$\mathcal{I}$	Fisher information matrix.	-
$\mathcal{N}$	Gaussian distribution.	-
$\mathcal{GP}$	Gaussian process.	-
$\mathcal{H}$	Hessian matrix.	-
$\mathbf{1}$	Identity matrix.	-

$\mathcal{K}$	Kernel.	-
$\mathfrak{g}$	Lie algebra.	-
G	Lie group.	-
Δ^2	Mahalanobis distance.	-
$\mathcal{Y}$	Measurement set.	-
$\mathbb{R}$	Real number.	-
$SE(3)$	Three-dimensional special euclidean group.	-
$SO(3)$	Three-dimensional special orthogonal group.	-
$\mathcal{X}$	State set.	-
$T_\varepsilon G$	Tangent space of the group G at the identity ε .	-
Δ	Trust region in the trust-region method.	-
$\mathbf{0}$	Zero matrix.	-
$\mathcal{J}$	Left Jacobian matrix of $SE(3)$.	-
$\mathcal{T}$	Adjoint map of $SE(3)$.	-

Bibliography

- [1] BARFOOT, T. D.: *State Estimation for Robotics: Second Edition*. 2nd ed. Cambridge University Press, 2024.
- [2] DURRANT-WHYTE, H. ; HENDERSON, T. C.: *Multisensor Data Fusion*. Ed. by SICILIANO, B. ; KHATIB, O. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 585–610. ISBN: 978-3-540-30301-5. DOI: 10 . 1007/978-3-540-30301-5_26.
- [3] THRUN, S. ; BURGARD, W. ; FOX, D.: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.
- [4] FISCHLER, M. A. ; BOLLES, R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24. no. 6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10 . 1145/358669 . 358692.
- [5] MARTÍNEZ-OTZETA, J. M. ; RODRÍGUEZ-MORENO, I. ; MENDIALDUA, I. ; SIERRA, B.: RANSAC for Robotic Applications: A Survey. In: *Sensors* 23. no. 1 (2023). ISSN: 1424-8220. DOI: 10 . 3390/s23010327.
- [6] YANG, S.-W. ; WANG, C.-C. ; CHANG, C.-H.: RANSAC matching: Simultaneous registration and segmentation. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 1905–1912. DOI: 10 . 1109/ROBOT . 2010 . 5509809.
- [7] LI, J. ; HU, Q. ; AI, M.: Point Cloud Registration Based on One-Point RANSAC and Scale-Annealing Biweight Estimation. In: *IEEE Transactions on Geoscience and Remote Sensing* 59. no. 11 (2021), pp. 9716–9729. DOI: 10 . 1109/TGRS . 2020 . 3045456.
- [8] HARTLEY, R. ; ZISSERMAN, A.: *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004.
- [9] RAGURAM, R. ; FRAHM, J.-M. ; POLLEFEYS, M.: A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In: *Computer Vision – ECCV 2008*. Ed. by FORSYTH, D. ; TORR, P. ; ZISSERMAN, A. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 500–513. ISBN: 978-3-540-88688-4.
- [10] LEPETIT, V. ; MORENO-NOGUER, F. ; FUA, P.: EPnP: An Accurate O(n) Solution to the PnP Problem. In: *Int. J. Comput. Vision* 81. no. 2 (Feb. 2009), pp. 155–166. ISSN: 0920-5691. DOI: 10 . 1007/s11263-008-0152-6.
- [11] SCARAMUZZA, D. ; FRAUNDORFER, F.: Visual Odometry [Tutorial]. In: *IEEE Robotics & Automation Magazine* 18. no. 4 (2011), pp. 80–92. DOI: 10 . 1109/MRA . 2011 . 943233.
- [12] QIN, T. ; LI, P. ; SHEN, S.: VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. In: *IEEE Transactions on Robotics* 34. no. 4 (2018), pp. 1004–1020. DOI: 10 . 1109/TRO . 2018 . 2853729.

- [13] ROSINOL, A. ; ABATE, M. ; CHANG, Y. ; CARLONE, L.: Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1689–1696. DOI: 10 . 1109 / ICRA40945 . 2020 . 9196885.
- [14] CASTALDO, G. ; ANGRISANO, A. ; GAGLIONE, S. ; TROISI, S.: P-RANSAC: An Integrity Monitoring Approach for GNSS Signal Degraded Scenario. In: *International Journal of Navigation and Observation* 2014. no. 1 (2014), p. 173818. DOI: <https://doi.org/10.1155/2014/173818>.
- [15] ZHANG, Q. ; LIN, H. ; DING, L. ; CHEN, Q. ; ZHANG, T. ; NIU, X.: RANSAC-Based Fault Detection and Exclusion Algorithm for Single-Difference Tightly Coupled GNSS/INS Integration. In: *IEEE Transactions on Intelligent Vehicles* 9. no. 2 (2024), pp. 3986–3997. DOI: 10 . 1109 /TIV . 2023 . 3342274.
- [16] CHENG, Q. ; CHEN, W. ; SUN, R. ; WANG, J. ; WENG, D.: RANSAC-based instantaneous real-time kinematic positioning with GNSS triple-frequency signals in urban areas. English. In: *Journal of Geodesy* 98. no. 4 (Apr. 2024). Publisher Copyright: © Springer-Verlag GmbH Germany, part of Springer Nature 2024. ISSN: 0949-7714. DOI: 10 . 1007/s00190-024-01833-6.
- [17] SPECIALE, P. ; PAUDEL, D. P. ; OSWALD, M. R. ; KROEGER, T. ; GOOL, L. V. ; POLLEFEYS, M.: Consensus Maximization with Linear Matrix Inequality Constraints. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5048–5056. DOI: 10 . 1109 /CVPR . 2017 . 536.
- [18] TZOUMAS, V. ; ANTONANTE, P. ; CARLONE, L.: Outlier-Robust Spatial Perception: Hardness, General-Purpose Algorithms, and Guarantees. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 5383–5390. DOI: 10 . 1109 /IROS40897 . 2019 . 8968174.
- [19] TONG, C. H. ; BARFOOT, T. D.: Batch heterogeneous outlier rejection for feature-poor SLAM. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2630–2637. DOI: 10 . 1109 /ICRA . 2011 . 5979612.
- [20] LATIF, Y. ; CADENA, C. ; NEIRA, J.: Robust loop closing over time for pose graph SLAM. In: *The International Journal of Robotics Research* 32. no. 14 (2013), pp. 1611–1626. DOI: 10 . 1177/0278364913498910.
- [21] BAI, F. ; VIDAL-CALLEJA, T. ; HUANG, S. ; XIONG, R.: Predicting Objective Function Change in Pose-Graph Optimization. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 145–152. DOI: 10 . 1109 / IROS . 2018 . 8594248.
- [22] XIE, L. ; WANG, S. ; MARKHAM, A. ; TRIGONI, N.: GraphTinker: Outlier rejection and inlier injection for pose graph SLAM. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6777–6784. DOI: 10 . 1109 / IROS . 2017 . 8206596.

-
- [23] SPECIALE, P. ; PAUDEL, D. P. ; OSWALD, M. R. ; KROEGER, T. ; GOOL, L. V. ; POLLEFEYS, M.: Consensus Maximization with Linear Matrix Inequality Constraints. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5048–5056. DOI: 10.1109/CVPR.2017.536.
- [24] JING, H. ; GAO, Y. ; SHAHBEIGI, S. ; DIANATI, M.: Integrity Monitoring of GNSS/INS Based Positioning Systems for Autonomous Vehicles: State-of-the-Art and Open Challenges. In: *IEEE Transactions on Intelligent Transportation Systems* 23. no. 9 (2022), pp. 14166–14187. DOI: 10.1109/TITS.2022.3149373.
- [25] WALTER, T. ; ENGE, P.: Weighted RAIM for Precision Approach. In: *Proceedings of the 8th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1995)*. 1995, pp. 1995–2004.
- [26] ANGUS, J. E.: RAIM with Multiple Faults. In: *NAVIGATION* 53. no. 4 (2006), pp. 249–257. DOI: <https://doi.org/10.1002/j.2161-4296.2006.tb00387.x>.
- [27] DREVELLE, V. ; BONNIFAIT, P.: High Integrity GNSS Location Zone Characterization using Interval Analysis. In: *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*. 2009, pp. 2178–2187.
- [28] BHAMIDIPATI, S. ; KOUSIK, S. ; GAO, G.: Set-Valued Shadow Matching using Zonotopes. In: *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*. 2021, pp. 2373–2390.
- [29] SU, J. ; SCHÖN, S. ; JOERGER, M.: Towards a set-based detector for GNSS integrity monitoring. In: *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2023, pp. 421–429. DOI: 10.1109/PLANS53410.2023.10139987.
- [30] LI, C. ; WASLANDER, S. L.: Visual Measurement Integrity Monitoring for UAV Localization. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2019, pp. 22–29. DOI: 10.1109/SSRR.2019.8848975.
- [31] ZHU, C. ; MEURER, M. ; GÜNTHER, C.: Integrity of Visual Navigation - Developments, Challenges, and Prospects. In: *NAVIGATION: Journal of the Institute of Navigation* 69. no. 2 (2022). DOI: 10.33012/navi.518.
- [32] NAGAI, K. ; CHEN, Y. ; SPENKO, M. ; HENDERSON, R. ; PERVAN, B.: Integrity with Extraction Faults in LiDAR-Based Urban Navigation for Driverless Vehicles. In: *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2023, pp. 1099–1106. DOI: 10.1109/PLANS53410.2023.10140132.
- [33] ARANA, G. D. ; HAFEZ, O. A. ; JOERGER, M. ; SPENKO, M.: Recursive Integrity Monitoring for Mobile Robot Localization Safety. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 305–311. DOI: 10.1109/ICRA.2019.8794115.

- [34] HAFEZ, O. A. ; ARANA, G. D. ; JOERGER, M. ; SPENKO, M.: Quantifying Robot Localization Safety: A New Integrity Monitoring Method for Fixed-Lag Smoothing. In: *IEEE Robotics and Automation Letters* 5. no. 2 (2020), pp. 3182–3189. DOI: 10.1109/LRA.2020.2975769.
- [35] HAFEZ, O. A. ; JOERGER, M. ; SPENKO, M.: How Safe Is Particle Filtering-Based Localization for Mobile Robots? An Integrity Monitoring Approach. In: *IEEE Transactions on Robotics* 40 (2024), pp. 3372–3387. DOI: 10.1109/TRO.2024.3420798.
- [36] HUBER, P. ; RONCHETTI, E.: *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN: 9781118210338.
- [37] BOSSE, M. ; AGAMENNONI, G. ; GILITSCHENSKI, I.: Robust Estimation and Applications in Robotics. In: *Foundations and Trends® in Robotics* 4. no. 4 (2016), pp. 225–269. ISSN: 1935-8253. DOI: 10.1561/23000000047.
- [38] ZHANG, Z.: Parameter estimation techniques: a tutorial with application to conic fitting. In: *Image and Vision Computing* 15. no. 1 (1997), pp. 59–76. ISSN: 0262-8856. DOI: [https://doi.org/10.1016/S0262-8856\(96\)01112-2](https://doi.org/10.1016/S0262-8856(96)01112-2).
- [39] DE MENEZES, D. ; PRATA, D. M. ; SECCHI, A. R. ; PINTO, J. C.: A review on robust M-estimators for regression analysis. In: *Computers & Chemical Engineering* 147 (2021), p. 107254.
- [40] MAC TAVISH, K. ; BARFOOT, T. D.: At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers. In: *2015 12th Conference on Computer and Robot Vision*. 2015, pp. 62–69. DOI: 10.1109/CRV.2015.52.
- [41] BABIN, P. ; GIGUÈRE, P. ; POMERLEAU, F.: Analysis of Robust Functions for Registration Algorithms. In: *2019 International Conference on Robotics and Automation (ICRA)* (2018), pp. 1451–1457.
- [42] VIZZO, I. ; GUADAGNINO, T. ; MERSCH, B. ; WIESMANN, L. ; BEHLEY, J. ; STACHNISS, C.: KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. In: *IEEE Robotics and Automation Letters* 8. no. 2 (2023), pp. 1029–1036. DOI: 10.1109/LRA.2023.3236571.
- [43] HU, G. ; KHOSOUSI, K. ; HUANG, S.: Towards a reliable SLAM back-end. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 37–43. DOI: 10.1109/IROS.2013.6696329.
- [44] LATIF, Y. ; CADENA, C. ; NEIRA, J.: Robust graph SLAM back-ends: A comparative analysis. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2683–2690. DOI: 10.1109/IROS.2014.6942929.
- [45] CRESPILO, O. G. ; ANDREETTI, A. ; GROSCH, A.: Design and Evaluation of Robust M-estimators for GNSS Positioning in Urban Environments. In: *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*. 2020, pp. 750–762.

- [46] ZHAO, W. ; PANERATI, J. ; SCHOELLIG, A. P.: Learning-Based Bias Correction for Time Difference of Arrival Ultra-Wideband Localization of Resource-Constrained Mobile Robots. In: *IEEE Robotics and Automation Letters* 6. no. 2 (2021), pp. 3639–3646. DOI: 10.1109/LRA.2021.3064199.
- [47] WANG, J. ; CHEN, X. ; SHI, C. ; LIU, J.: Robust M-Estimation-Based ICKF for GNSS Outlier Mitigation in GNSS/SINS Navigation Applications. In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–17. DOI: 10.1109/TIM.2023.3306521.
- [48] AGAMENNONI, G. ; FURGALE, P. ; SIEGWART, R.: Self-tuning M-estimators. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4628–4635. DOI: 10.1109/ICRA.2015.7139840.
- [49] DING, Y. ; CHAUCHAT, P. ; PAGES, G. ; ASSEMAN, P.: Learning-Enhanced Adaptive Robust GNSS Navigation in Challenging Environments. In: *IEEE Robotics and Automation Letters* 7. no. 4 (2022), pp. 9905–9912. DOI: 10.1109/LRA.2022.3192889.
- [50] BARRON, J. T.: A General and Adaptive Robust Loss Function. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4326–4334. DOI: 10.1109/CVPR.2019.00446.
- [51] CHEBROLU, N. ; LÄBE, T. ; VYSOTSKA, O. ; BEHLEY, J. ; STACHNISS, C.: Adaptive Robust Kernels for Non-Linear Least Squares Problems. In: *IEEE Robotics and Automation Letters* 6. no. 2 (2021), pp. 2240–2247. DOI: 10.1109/LRA.2021.3061331.
- [52] DAS, S. ; GROSS, J. N.: Analysis of Scale-Variant Robust Kernel Optimization for Nonlinear Least-Squares Problems. In: *IEEE Transactions on Aerospace and Electronic Systems* 59. no. 6 (2023), pp. 7529–7538. DOI: 10.1109/TAES.2023.3290142.
- [53] YANG, H. ; ANTONANTE, P. ; TZOUMAS, V. ; CARLONE, L.: Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection. In: *IEEE Robotics and Automation Letters* 5. no. 2 (2020), pp. 1127–1134. DOI: 10.1109/LRA.2020.2965893.
- [54] JUNG, K. ; HITCHCOX, T. ; FORBES, J. R.: An Adaptive Graduated Nonconvexity Loss Function for Robust Nonlinear Least-Squares Solutions. In: *IEEE Transactions on Robotics* 40 (2024), pp. 4207–4221. DOI: 10.1109/TRO.2024.3434169.
- [55] STENGEL, R. F.: *Optimal control and estimation / Robert F. Stengel*. eng. Dover books on advanced mathematics. New York: Dover Publications, 1994. ISBN: 0486682005.
- [56] BISHOP, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.
- [57] SHIKHALIEV, A. P. ; POTTER, L. C. ; CHI, Y.: Low-Rank Structured Covariance Matrix Estimation. In: *IEEE Signal Processing Letters* 26. no. 5 (2019), pp. 700–704. DOI: 10.1109/LSP.2019.2906405.

- [58] WONG, J. N. ; YOON, D. J. ; SCHOELLIG, A. P. ; BARFOOT, T. D.: A Data-Driven Motion Prior for Continuous-Time Trajectory Estimation on SE(3). In: *IEEE Robotics and Automation Letters* 5. no. 2 (2020), pp. 1429–1436. DOI: 10 . 1109 / LRA . 2020 . 2969153.
- [59] BARFOOT, T. D. ; FORBES, J. R. ; YOON, D. J.: Exactly sparse Gaussian variational inference with application to derivative-free batch nonlinear state estimation. In: *The International Journal of Robotics Research* 39. no. 13 (2020), pp. 1473–1502. DOI: 10 . 1177 / 0278364920937608.
- [60] WONG, J. N. ; YOON, D. J. ; SCHOELLIG, A. P. ; BARFOOT, T. D.: Variational Inference With Parameter Learning Applied to Vehicle Trajectory Estimation. In: *IEEE Robotics and Automation Letters* 5. no. 4 (2020), pp. 5291–5298. DOI: 10 . 1109 / LRA . 2020 . 3007381.
- [61] COURTS, J. ; WILLS, A. G. ; SCHÖN, T. B.: Gaussian Variational State Estimation for Nonlinear State-Space Models. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 5979–5993. DOI: 10 . 1109 / TSP . 2021 . 3122296.
- [62] GOUDAR, A. ; ZHAO, W. ; BARFOOT, T. D. ; SCHOELLIG, A. P.: Gaussian Variational Inference with Covariance Constraints Applied to Range-only Localization. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 2872–2879. DOI: 10 . 1109 / IROS47612 . 2022 . 9981520.
- [63] PFEIFER, T. ; PROTZEL, P.: Incrementally learned Mixture Models for GNSS Localization. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1131–1138. DOI: 10 . 1109 / IVS . 2019 . 8813847.
- [64] WEN, W. ; BAI, X. ; HSU, L.-T. ; PFEIFER, T.: GNSS/LiDAR Integration Aided by Self-Adaptive Gaussian Mixture Models in Urban Scenarios: An Approach Robust to Non-Gaussian Noise. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2020, pp. 647–654. DOI: 10 . 1109 / PLANS46316 . 2020 . 9110157.
- [65] ZHAO, W. ; GOUDAR, A. ; TANG, M. ; QIAO, X. ; SCHOELLIG, A. P.: Uncertainty-Aware Gaussian Mixture Model for UWB Time Difference of Arrival Localization in Cluttered Environments. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 5266–5273. DOI: 10 . 1109 / IROS55552 . 2023 . 10342365.
- [66] PFEIFER, T.: *Adaptive Estimation using Gaussian Mixtures*. Available at <https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-873459>. PhD thesis. Chemnitz, Germany: Technischen Universität Chemnitz, Oct. 2023.
- [67] SÜNDERHAUF, N. ; PROTZEL, P.: Switchable constraints for robust pose graph SLAM. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1879–1884. DOI: 10 . 1109 / IROS . 2012 . 6385590.

-
- [68] AGARWAL, P. ; TIPALDI, G. D. ; SPINELLO, L. ; STACHNISS, C. ; BURGARD, W.: Robust map optimization using dynamic covariance scaling. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 62–69. DOI: 10.1109/ICRA.2013.6630557.
- [69] PFEIFER, T. ; LANGE, S. ; PROTZEL, P.: Advancing Mixture Models for Least Squares Optimization. In: *IEEE Robotics and Automation Letters* 6. no. 2 (2021), pp. 3941–3948. DOI: 10.1109/LRA.2021.3067307.
- [70] HUANG, G. ; KAESS, M. ; LEONARD, J. J. ; ROUMELIOTIS, S. I.: Analytically-selected multi-hypothesis incremental MAP estimation. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6481–6485. DOI: 10.1109/ICASSP.2013.6638914.
- [71] HSIAO, M. ; KAESS, M.: MH-iSAM2: Multi-hypothesis iSAM using Bayes Tree and Hypo-tree. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 1274–1280. DOI: 10.1109/ICRA.2019.8793854.
- [72] KAESS, M. ; JOHANNSSON, H. ; ROBERTS, R. ; ILA, V. ; LEONARD, J. ; DELLAERT, F.: iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641.
- [73] HSIAO, M. ; MANGELSON, J. G. ; SURESH, S. ; DEBRUNNER, C. ; KAESS, M.: ARAS: Ambiguity-aware Robust Active SLAM based on Multi-hypothesis State and Map Estimations. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5037–5044. DOI: 10.1109/IROS45743.2020.9341384.
- [74] DOHERTY, K. J. ; LU, Z. ; SINGH, K. ; LEONARD, J. J.: Discrete-Continuous Smoothing and Mapping. In: *IEEE Robotics and Automation Letters* 7. no. 4 (2022), pp. 12395–12402. DOI: 10.1109/LRA.2022.3216938.
- [75] ZHANG, Y. ; SEVERINSEN, O. A. ; LEONARD, J. J. ; CARLONE, L. ; KHOSOUSI, K.: Data-Association-Free Landmark-based SLAM. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 8349–8355. DOI: 10.1109/ICRA48891.2023.10160719.
- [76] ROSEN, D. M. ; DOHERTY, K. J. ; TERÁN ESPINOZA, A. ; LEONARD, J. J.: Advances in Inference and Representation for Simultaneous Localization and Mapping. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4. no. Volume 4, 2021 (2021), pp. 215–242. ISSN: 2573-5144. DOI: <https://doi.org/10.1146/annurev-control-072720-082553>.
- [77] GIAMOU, M. P.: *Semidefinite Relaxations for Geometric Problems in Robotics*. Available at <https://tspace.library.utoronto.ca/handle/1807/126896?mode=full>. PhD thesis. Toronto, Canada: University of Toronto, Mar. 2023.

- [78] BOYD, S. ; VANDENBERGHE, L.: *Convex Optimization*. Cambridge University Press, 2004.
- [79] ROSEN, D. M. ; CARLONE, L. ; BANDEIRA, A. S. ; LEONARD, J. J.: SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. In: *The International Journal of Robotics Research* 38. no. 2-3 (2019), pp. 95–125. DOI: 10.1177/0278364918784361.
- [80] YANG, H. ; CARLONE, L.: One ring to rule them all: certifiably robust geometric perception with outliers. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.
- [81] PAPALIA, A. ; FISHBERG, A. ; O'NEILL, B. W. ; HOW, J. P. ; ROSEN, D. M. ; LEONARD, J. J.: Certifiably Correct Range-Aided SLAM. In: *IEEE Transactions on Robotics* 40 (2024), pp. 4265–4283. DOI: 10.1109/TRO.2024.3454430.
- [82] CARLONE, L. ; ROSEN, D. M. ; CALAFIORE, G. ; LEONARD, J. J. ; DELLAERT, F.: Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 125–132. DOI: 10.1109/IROS.2015.7353364.
- [83] DÜMBGEN, F. ; HOLMES, C. ; BARFOOT, T. D.: Safe and Smooth: Certified Continuous-Time Range-Only Localization. In: *IEEE Robotics and Automation Letters* 8. no. 2 (2023), pp. 1117–1124. DOI: 10.1109/LRA.2022.3233232.
- [84] HOLMES, C. ; BARFOOT, T. D.: An Efficient Global Optimality Certificate for Landmark-Based SLAM. In: *IEEE Robotics and Automation Letters* 8. no. 3 (2023), pp. 1539–1546. DOI: 10.1109/LRA.2023.3238173.
- [85] HOLMES, C. ; DÜMBGEN, F. ; BARFOOT, T.: On Semidefinite Relaxations for Matrix-Weighted State-Estimation Problems in Robotics. In: *IEEE Transactions on Robotics* 40 (2024), pp. 4805–4824. DOI: 10.1109/TRO.2024.3475220.
- [86] DÜMBGEN, F. ; HOLMES, C. ; BARFOOT, T. D.: A Program of Research for Globally Optimal State Estimation. In: (2024).
- [87] CURSI, F. ; YANG, G.-Z.: A Novel Approach for Outlier Detection and Robust Sensory Data Model Learning. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4250–4257. DOI: 10.1109/IROS40897.2019.8967653.
- [88] HSU, L.-T.: GNSS multipath detection using a machine learning approach. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317700.
- [89] SUZUKI, T. ; AMANO, Y.: NLOS Multipath Classification of GNSS Signal Correlation Output Using Machine Learning. In: *Sensors* 21. no. 7 (2021). ISSN: 1424-8220. DOI: 10.3390/s21072503.

-
- [90] ZHANG, G. ; XU, P. ; XU, H. ; HSU, L.-T.: Prediction on the Urban GNSS Measurement Uncertainty Based on Deep Learning Networks With Long Short-Term Memory. In: *IEEE Sensors Journal* 21. no. 18 (2021), pp. 20563–20577. doi: 10 . 1109 / JSEN . 2021 . 3098006.
 - [91] MOHANTY, A. ; GAO, G.: Learning GNSS Positioning Corrections for Smartphones Using Graph Convolution Neural Networks. In: *NAVIGATION: Journal of the Institute of Navigation* 70. no. 4 (2023). issn: 0028-1522. doi: 10 . 33012 / navi . 622.
 - [92] NG, A. ; JORDAN, M.: On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In: *Advances in Neural Information Processing Systems*. Ed. by DIETTERICH, T. ; BECKER, S. ; GHAHRAMANI, Z. Vol. 14. MIT Press, 2001.
 - [93] HUANG, Q. ; PU, C. ; FOURIE, D. ; KHOSOUSSI, K. ; HOW, J. P. ; LEONARD, J. J.: NF-iSAM: Incremental Smoothing and Mapping via Normalizing Flows. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1095–1102. doi: 10 . 1109 / ICRA48506 . 2021 . 9561436.
 - [94] HUANG, Q. ; PU, C. ; KHOSOUSSI, K. ; ROSEN, D. M. ; FOURIE, D. ; HOW, J. P. ; LEONARD, J. J.: Incremental Non-Gaussian Inference for SLAM Using Normalizing Flows. In: *IEEE Transactions on Robotics* 39. no. 2 (2023), pp. 1458–1475. doi: 10 . 1109 / TRO . 2022 . 3216498.
 - [95] WEN, W.: *3D LiDAR aided GNSS positioning and its application in sensor fusion for autonomous vehicles in urban canyons*. Available at [https : / / theses . lib . polyu . edu . hk / handle / 200 / 11076](https://theses.lib.polyu.edu.hk/handle/200/11076). PhD thesis. Hong Kong: Hong Kong Polytechnic University, 2020.
 - [96] HU, R. ; WEN, W. ; HSU, L.-T.: Fisheye Camera Aided GNSS NLOS Detection and Learning-Based Pseudorange Bias Correction for Intelligent Vehicles in Urban Canyons. In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 6088–6095. doi: 10 . 1109 / ITSC57777 . 2023 . 10422540.
 - [97] AMINI, A. ; ROSMAN, G. ; KARAMAN, S. ; RUS, D.: Variational End-to-End Navigation and Localization. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8958–8964. doi: 10 . 1109 / ICRA . 2019 . 8793579.
 - [98] LIU, Z. ; AMINI, A. ; ZHU, S. ; KARAMAN, S. ; HAN, S. ; RUS, D. L.: Efficient and Robust LiDAR-Based End-to-End Navigation. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 13247–13254. doi: 10 . 1109 / ICRA48506 . 2021 . 9561299.
 - [99] ALI, S. A. ; AOUADA, D. ; REIS, G. ; STRICKER, D.: DELO: Deep Evidential LiDAR Odometry using Partial Optimal Transport. In: *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2023, pp. 4519–4528. doi: 10 . 1109 / ICCVW60793 . 2023 . 00486.

- [100] HSU, L.-T.: Analysis and modeling GPS NLOS effect in highly urbanized area. In: *GPS solutions* 22. no. 1 (2018), p. 7.
- [101] HSU, L.-T.: Analysis and modeling GPS NLOS effect in highly urbanized area. In: *GPS Solutions* 22 (Nov. 2017). DOI: 10.1007/s10291-017-0667-9.
- [102] DELLAERT, F.: Factor Graphs: Exploiting Structure in Robotics. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4. no. Volume 4, 2021 (2021), pp. 141–166. ISSN: 2573-5144. DOI: <https://doi.org/10.1146/annurev-control-061520-010504>.
- [103] TANG, T. Y. ; YOON, D. J. ; BARFOOT, T. D.: A White-Noise-on-Jerk Motion Prior for Continuous-Time Trajectory Estimation on SE(3). In: *IEEE Robot. Autom. Lett.* 4. no. 2 (2019), pp. 594–601. DOI: 10.1109/LRA.2019.2891492.
- [104] WEN, W. ; PFEIFER, T. ; BAI, X. ; HSU, L.-T.: Factor graph optimization for GNSS/INS integration: A comparison with the extended Kalman filter. In: *NAVIGATION* 68. no. 2 (2021), pp. 315–331.
- [105] SÜNDERHAUF, N. ; PROTZEL, P.: Towards robust graphical models for GNSS-based localization in urban environments. In: *Int. Multi-Conf. on Systems, Signals & Devices*. 2012, pp. 1–6. DOI: 10.1109/SSD.2012.6198059.
- [106] WATSON, R. ; GROSS, J.: Robust Navigation in GNSS Degraded Environment Using Graph Optimization. In: *Proc. of 30th Int. Tech. Meet. Satell. Div. Inst. Navig.* Sept. 2017. DOI: 10.33012/2017.15164.
- [107] GROVES, P.: *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. Artech House, 2013.
- [108] WEN, W. ; KAN, Y. C. ; HSU, L.-T.: Performance Comparison of GNSS/INS Integrations Based on EKF and Factor Graph Optimization. In: *Proc. of 32th Int. Tech. Meet. Satell. Div. Inst. Navig.* Sept. 2019, pp. 3019–3032. DOI: 10.33012/2019.17129.
- [109] WEN, W. ; BAI, X. ; KAN, Y. C. ; HSU, L.-T.: Tightly Coupled GNSS/INS Integration via Factor Graph and Aided by Fish-Eye Camera. In: *IEEE Trans. on Vehicular Technology* 68. no. 11 (2019), pp. 10651–10662. DOI: 10.1109/TVT.2019.2944680.
- [110] WEN, W. ; HSU, L.-T.: Towards Robust GNSS Positioning and Real-time Kinematic Using Factor Graph Optimization. In: *2021 IEEE Int. Conf. on Robot. and Automat. (ICRA)*. 2021, pp. 5884–5890. DOI: 10.1109/ICRA48506.2021.9562037.
- [111] BAI, X. ; WEN, W. ; HSU, L.-T.: Time-Correlated Window-Carrier-Phase-Aided GNSS Positioning Using Factor Graph Optimization for Urban Positioning. In: *IEEE Trans. Aerosp. Electron. Syst.* 58. no. 4 (2022), pp. 3370–3384. DOI: 10.1109/TAES.2022.3149730.
- [112] SUZUKI, T.: GNSS Odometry: Precise Trajectory Estimation Based on Carrier Phase Cycle Slip Estimation. In: *IEEE Robot. Autom. Lett.* 7. no. 3 (2022), pp. 7319–7326. DOI: 10.1109/LRA.2022.3182795.

- [113] CONGRAM, B. ; BARFOOT, T. D.: Field Testing and Evaluation of Single-Receiver GPS Odometry for Use in Robotic Navigation. In: *Field Robotics 2* (2022), pp. 1849–1873. DOI: 10.55417/fr.2022057.
- [114] PFEIFER, T. ; WEISSIG, P. ; LANGE, S. ; PROTZEL, P.: Robust factor graph optimization - a comparison for sensor fusion applications. In: *2016 IEEE 21st Int. Conf. on Emerging Technologies and Factory Automation*. 2016, pp. 1–4. DOI: 10.1109/ETFA.2016.7733598.
- [115] BARRON, J. T.: A General and Adaptive Robust Loss Function. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4326–4334. DOI: 10.1109/CVPR.2019.00446.
- [116] SUZUKI, T.: 1st Place Winner of the Smartphone Decimeter Challenge: Two-Step Optimization of Velocity and Position Using Smartphone’s Carrier Phase Observations. In: *Proc. of the 35th Int. Tech. Meet. Satell. Div. Inst. Navig.* Sept. 2022, pp. 2276–2286. DOI: 10.33012/2022.18377.
- [117] PFEIFER, T. ; PROTZEL, P.: Incrementally learned Mixture Models for GNSS Localization. In: *2019 IEEE Intelli. Vehicles Symposium (IV)*. 2019, pp. 1131–1138. DOI: 10.1109/IVS.2019.8813847.
- [118] WEN, W. ; ZHANG, G. ; HSU, L.-T.: GNSS Outlier Mitigation via Graduated Non-Convexity Factor Graph Optimization. In: *IEEE Trans. on Vehicular Technology* 71. no. 1 (2022), pp. 297–310. DOI: 10.1109/TVT.2021.3130909.
- [119] KOIDE, K. ; MIURA, J. ; MENEGATTI, E.: A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. In: *Int. J. of Advanced Robotic Systems* 16. no. 2 (2019). DOI: 10.1177/1729881419841532.
- [120] SHAN, T. ; ENGLLOT, B. ; MEYERS, D. ; WANG, W. ; RATTI, C. ; RUS, D.: LIO-SAM: Tightly-coupled LiDAR Inertial Odometry via Smoothing and Mapping. In: *2020 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)* (2020), pp. 5135–5142.
- [121] ZHANG, J. ; WEN, W. ; FENG, H. ; CHEN, X. ; HSU, L.-T.: Continuous GNSS-RTK Aided by LiDAR/Inertial Odometry with Intelligent GNSS Selection in Urban Canyons. In: *Proc. of 34th Int. Tech. Meet. Satell. Div. Inst. Navig.* Sept. 2021, pp. 4198–4207. DOI: 10.33012/2021.18089.
- [122] GONG, Z. ; LIU, P. ; WEN, F. ; YING, R. ; JI, X. ; MIAO, R. ; XUE, W.: Graph-Based Adaptive Fusion of GNSS and VIO Under Intermittent GNSS-Degraded Environment. In: *IEEE Trans. Instrum. Meas.* 70 (2021), pp. 1–16. DOI: 10.1109/TIM.2020.3039640.
- [123] CAO, S. ; LU, X. ; SHEN, S.: GVINS: Tightly Coupled GNSS–Visual–Inertial Fusion for Smooth and Consistent State Estimation. In: *IEEE Trans. on Robot.* 38. no. 4 (2022), pp. 2004–2021. DOI: 10.1109/TRO.2021.3133730.

- [124] WEN, W. ; BAI, X. ; HSU, L.-T.: 3D Vision Aided GNSS Real-Time Kinematic Positioning for Autonomous Systems in Urban Canyons. In: *NAVIGATION* 70. no. 3 (2023). ISSN: 0028-1522. DOI: 10.33012/navi.590.
- [125] WANG, Y. ; SONG, W. ; LOU, Y. ; ZHANG, Y. ; HUANG, F. ; TU, Z. ; LIANG, Q.: Rail Vehicle Localization and Mapping With LiDAR-Vision-Inertial-GNSS Fusion. In: *IEEE Robot. Autom. Lett.* 7. no. 4 (2022), pp. 9818–9825. DOI: 10.1109/LRA.2022.3190093.
- [126] WISTH, D. ; CAMURRI, M. ; DAS, S. ; FALLON, M.: Unified Multi-Modal Landmark Tracking for Tightly Coupled LiDAR-Visual-Inertial Odometry. In: *IEEE Robot. Autom. Lett.* 6. no. 2 (2021), pp. 1004–1011. DOI: 10.1109/LRA.2021.3056380.
- [127] SHAO, W. ; VIJAYARANGAN, S. ; LI, C. ; KANTOR, G.: Stereo Visual Inertial LiDAR Simultaneous Localization and Mapping. In: *2019 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2019, pp. 370–377. DOI: 10.1109/IROS40897.2019.8968012.
- [128] NUBERT, J. ; KHATTAK, S. ; HUTTER, M.: Graph-based Multi-sensor Fusion for Consistent Localization of Autonomous Construction Robots. In: *2022 Int. Conf. on Robotics and Automation (ICRA)*. 2022, pp. 10048–10054. DOI: 10.1109/ICRA46639.2022.9812386.
- [129] ZHAO, S. ; ZHANG, H. ; WANG, P. ; NOGUEIRA, L. ; SCHERER, S.: Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments. In: *2021 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2021, pp. 8729–8736. DOI: 10.1109/IROS51168.2021.9635862.
- [130] CIOFFI, G. ; SCARAMUZZA, D.: Tightly-coupled Fusion of Global Positional Measurements in Optimization-based Visual-Inertial Odometry. In: *2020 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2020, pp. 5089–5095. DOI: 10.1109/IROS45743.2020.9341697.
- [131] BOCHE, S. ; ZUO, X. ; SCHAEFER, S. ; LEUTENEGGER, S.: Visual-Inertial SLAM with Tightly-Coupled Dropout-Tolerant GPS Fusion. In: *2022 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2022, pp. 7020–7027. DOI: 10.1109/IROS47612.2022.9981134.
- [132] HU, C. ; ZHU, S. ; LIANG, Y. ; SONG, W.: Tightly-Coupled Visual-Inertial-Pressure Fusion Using Forward and Backward IMU Preintegration. In: *IEEE Robot. Autom. Lett.* 7. no. 3 (2022), pp. 6790–6797. DOI: 10.1109/LRA.2022.3177847.
- [133] FURGALE, P. ; BARFOOT, T. D. ; SIBLEY, G.: Continuous-time batch estimation using temporal basis functions. In: *2012 IEEE Int. Conf. on Robot. and Automat.* 2012, pp. 2088–2095. DOI: 10.1109/ICRA.2012.6225005.
- [134] HUG, D. ; BÄNNINGER, P. ; ALZUGARAY, I. ; CHLI, M.: Continuous-Time Stereo-Inertial Odometry. In: *IEEE Robot. Autom. Lett.* 7. no. 3 (2022), pp. 6455–6462. DOI: 10.1109/LRA.2022.3173705.

-
- [135] BARFOOT, T. D. ; TONG, C. H. ; SÄRKKÄ, S.: Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. In: *Robotics: Science and Systems*. 2014.
 - [136] MUKADAM, M. ; DONG, J. ; DELLAERT, F. ; BOOTS, B.: STEAP: simultaneous trajectory estimation and planning. In: *Auton. Robots* 43 (Feb. 2019). DOI: 10.1007/s10514-018-9770-1.
 - [137] YAN, X. ; INDELMAN, V. ; BOOTS, B.: Incremental sparse GP regression for continuous-time trajectory estimation and mapping. In: *Robotics and Autonomous Systems* 87 (2017), pp. 120–132. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2016.10.004>.
 - [138] DONG, J. ; BURNHAM, J. G. ; BOOTS, B. ; RAINS, G. C. ; DELLAERT, F.: 4D Crop Monitoring: Spatio-temporal reconstruction for agriculture. In: *2017 IEEE Int. Conf. on Robot. and Automat. (ICRA)* (2017), pp. 3878–3885.
 - [139] JOHNSON, J. ; MANGELSON, J. ; BARFOOT, T. D. ; RANDAL, B.: *Continuous-time Trajectory Estimation: A Comparative Study Between Gaussian Process and Spline-based Approaches*. 2024. DOI: 10.48550/arXiv.2402.00399. eprint: arXiv:2402.00399.
 - [140] TALBOT, W. ; NUBERT, J. ; TUNA, T. ; CADENA, C. ; DÜMBGEN, F. ; TORDESILLAS, J. ; BARFOOT, T. D. ; HUTTER, M.: *Continuous-Time State Estimation Methods in Robotics: A Survey*. 2024. arXiv: 2411.03951 [cs.RO].
 - [141] LABBÉ, M. ; MICHAUD, F.: Memory management for real-time appearance-based loop closure detection. In: *2011 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2011, pp. 1271–1276. DOI: 10.1109/IROS.2011.6094602.
 - [142] LABBÉ, M. ; MICHAUD, F.: RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. In: *J. Field Robot.* 36. no. 2 (2019), pp. 416–446. DOI: <https://doi.org/10.1002/rob.21831>.
 - [143] VINCENT, J. ; LABBÉ, M. ; LAUZON, J.-S. ; GRONDIN, F. ; COMTOIS-RIVET, P.-M. ; MICHAUD, F.: Dynamic Object Tracking and Masking for Visual SLAM. In: *2020 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2020, pp. 4974–4979. DOI: 10.1109/IROS45743.2020.9340958.
 - [144] LABBÉ, M. ; MICHAUD, F.: Multi-Session Visual SLAM for Illumination-Invariant Re-Localization in Indoor Environments. In: *Frontiers in Robotics and AI* 9 (2022). ISSN: 2296-9144. DOI: 10.3389/frobt.2022.801886.
 - [145] COLOSI, M. ; ALOISE, I. ; GUADAGNINO, T. ; SCHLEGEL, D. ; CORTE, B. D. ; ARRAS, K. O. ; GRISETTI, G.: Plug-and-Play SLAM: A Unified SLAM Architecture for Modularity and Ease of Use. In: *2020 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2020, pp. 5051–5057. DOI: 10.1109/IROS45743.2020.9341611.

- [146] SOLÀ, J. ; VALLVÉ, J. ; CASALS, J. ; DERAY, J. ; FOURMY, M. ; ATCHUTHAN, D. ; COROMINAS-MURTRA, A. ; ANDRADE-CETTO, J.: WOLF: A Modular Estimation Framework for Robotics Based on Factor Graphs. In: *IEEE Robot. Autom. Lett.* 7. no. 2 (2022), pp. 4710–4717. DOI: 10 . 1109 / LRA . 2022 . 3151404.
- [147] BARFOOT, T. D. ; TONG, C. H. ; SÄRKKÄ, S.: Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression.. In: *Robotics: Science and Systems*. Vol. 10. Citeseer. 2014, pp. 1–10.
- [148] ANDERSON, S. ; BARFOOT, T. D.: Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3). In: *2015 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2015, pp. 157–164. DOI: 10 . 1109 / IROS . 2015 . 7353368.
- [149] BARFOOT, T. D. ; FURGALE, P. T.: Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems. In: *IEEE Trans. on Robot.* 30. no. 3 (2014), pp. 679–693. DOI: 10 . 1109 / TRO . 2014 . 2298059.
- [150] WONG, J. N. ; YOON, D. J. ; SCHOELLIG, A. P. ; BARFOOT, T. D.: A Data-Driven Motion Prior for Continuous-Time Trajectory Estimation on SE(3). In: *IEEE Robot. Autom. Lett.* 5. no. 2 (2020), pp. 1429–1436. DOI: 10 . 1109 / LRA . 2020 . 2969153.
- [151] DONG, J. ; MUKADAM, M. ; BOOTS, B. ; DELLAERT, F.: Sparse Gaussian Processes on Matrix Lie Groups: A Unified Framework for Optimizing Continuous-Time Trajectories. In: *2018 IEEE Int. Conf. on Robot. and Automat. (ICRA)*. 2018, pp. 6497–6504. DOI: 10 . 1109 / ICRA . 2018 . 8461077.
- [152] WARREN, M. ; GREEFF, M. ; PATEL, B. ; COLLIER, J. ; SCHOELLIG, A. P. ; BARFOOT, T. D.: There’s No Place Like Home: Visual Teach and Repeat for Emergency Return of Multirotor UAVs During GPS Failure. In: *IEEE Robot. Autom. Lett.* 4. no. 1 (2019), pp. 161–168. DOI: 10 . 1109 / LRA . 2018 . 2883408.
- [153] DELLAERT, F. ; KAESS, M. 2017. DOI: 10 . 1561 / 23000000043.
- [154] BELL, B. ; CATHEY, F.: The iterated Kalman filter update as a Gauss-Newton method. In: *IEEE Transactions on Automatic Control* 38. no. 2 (1993), pp. 294–297. DOI: 10 . 1109 / 9 . 250476.
- [155] BROSSARD, M. ; BARRAU, A. ; BONNABEL, S.: A Code for Unscented Kalman Filtering on Manifolds (UKF-M). In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5701–5708. DOI: 10 . 1109 / ICRA40945 . 2020 . 9197489.
- [156] FORNASIER, A. ; NG, Y. ; MAHONY, R. ; WEISS, S.: Equivariant Filter Design for Inertial Navigation Systems with Input Measurement Biases. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4333–4339. DOI: 10 . 1109 / ICRA46639 . 2022 . 9811778.
- [157] GEHRT, J.-J. ; ZWEIGEL, R. ; KONRAD, T. ; ABEL, D.: DVL-aided navigation filter for maritime applications. In: *IFAC-PapersOnLine* 51. no. 29 (2018), pp. 418–423.

-
- [158] GEHRT, J.-J. ; NITSCH, M. ; ABEL, D. ; ZWEIGEL, R.: High accuracy navigation filter with dual antenna enabling double-differencing with dual-constellation. In: *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*. 2019, pp. 2186–2197.
- [159] GEHRT, J.-J. ; NITSCH, M. ; ABEL, D. ; ZWEIGEL, R.: High Accuracy Navigation Filter with Dual Antenna enabling Double-Differencing with Dual-Constellation. In: *32nd Int. Tech. Meet. Satell. Div. Inst. Navig.* Oct. 2019, pp. 2186–2197. doi: 10 . 33012 / 2019 . 16917.
- [160] RANGANATHAN, A. ; KAESS, M. ; DELLAERT, F.: Fast 3D pose estimation with out-of-sequence measurements. In: *2007 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2007, pp. 2486–2493. doi: 10 . 1109 / IROS . 2007 . 4399318.
- [161] KAESS, M. ; JOHANSSON, H. ; ROBERTS, R. ; ILA, V. ; LEONARD, J. ; DELLAERT, F.: iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: *2011 IEEE Int. Conf. on Robotics and Automation*. 2011, pp. 3281–3288. doi: 10 . 1109 / ICRA . 2011 . 5979641.
- [162] HUBER, P. J. ; RONCETTI, E. M.: *Robust Statistics*. 2nd. John Wiley & Sons, Inc., 2009. ISBN: 9780470129906.
- [163] PESONEN, H.: Robust estimation techniques for GNSS positioning. In: (Jan. 2007).
- [164] KNIGHT, N. ; WANG, J.: A Comparison of Outlier Detection Procedures and Robust Estimation Methods in GPS Positioning. In: *Journal of Navigation* 62 (Oct. 2009), pp. 699–709. doi: 10 . 1017 / S0373463309990142.
- [165] GAGLIONE, S. ; INNAC, A. ; PASTORE CARBONE, S. ; TROISI, S. ; ANGRISANO, A.: Robust estimation methods applied to GPS in harsh environments. In: *2017 European Navigation Conference (ENC)*. 2017, pp. 14–25. doi: 10 . 1109 / EURONAV . 2017 . 7954169.
- [166] CRESPILO, O. G. ; MEDINA, D. ; SKALoud, J. ; MEURER, M.: Tightly coupled GNSS/INS integration based on robust M-estimators. In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018, pp. 1554–1561. doi: 10 . 1109 / PLANS . 2018 . 8373551.
- [167] GAGLIONE, S. ; ANGRISANO, A. ; CROCETTO, N.: Robust Kalman Filter applied to GNSS positioning in harsh environment. In: *2019 European Navigation Conference (ENC)*. 2019, pp. 1–6. doi: 10 . 1109 / EURONAV . 2019 . 8714132.
- [168] MEDINA, D. ; LI, H. ; VILÀ-VALLS, J. ; CLOSAS, P.: Robust Statistics for GNSS Positioning under Harsh Conditions: A Useful Tool?. In: *Sensors* 19. no. 24 (2019). ISSN: 1424-8220. doi: 10 . 3390 / s19245402.
- [169] MEDINA, D. ; LI, H. ; VILÀ-VALLS, J. ; CLOSAS, P.: On Robust Statistics for GNSS Single Point Positioning. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 3281–3287. doi: 10 . 1109 / ITSC . 2019 . 8917484.

- [170] CHAUCHAT, P. ; BELLÉS, A. ; MEDINA, D. ; VILÀ-VALLS, J.: Insights on Adaptive Robust Filtering for Navigation Under Harsh Time-Varying Environments. In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. 2024, pp. 1227–1231. doi: 10.23919/EUSIPCO63174.2024.10715142.
- [171] CRESPILO, O. G. ; ANDREETTI, A. ; GROSCH, A.: Design and Evaluation of Robust M-estimators for GNSS Positioning in Urban Environments. In: *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*. 2020, pp. 750–762. doi: 10.33012/2020.17211.
- [172] DAS, S. ; GROSS, J. N.: Analysis of Scale-Variant Robust Kernel Optimization for Nonlinear Least-Squares Problems. In: *IEEE Transactions on Aerospace and Electronic Systems* 59. no. 6 (2023), pp. 7529–7538. doi: 10.1109/TAES.2023.3290142.
- [173] ALBUQUERQUE, J. S. ; BIEGLER, L. T.: Data reconciliation and gross-error detection for dynamic systems. In: *AIChE Journal* 42. no. 10 (1996), pp. 2841–2856. doi: <https://doi.org/10.1002/aic.690421014>.
- [174] HAMPEL, F. R. ; RONCHETTI, E. M. ; ROUSSEEUW, P. J. ; STAHEL, W. A.: *Robust statistics: the approach based on influence functions*. John Wiley & Sons, 2011.
- [175] ZHU, N. ; MARAIS, J. ; BÉTAILLE, D. ; BERBINEAU, M.: GNSS Position Integrity in Urban Environments: A Review of Literature. In: *IEEE Transactions on Intelligent Transportation Systems* 19. no. 9 (2018), pp. 2762–2778. doi: 10.1109/TITS.2017.2766768.
- [176] LIM, B. ; ARIK, S. O. ; LOEFF, N. ; PFISTER, T.: Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. In: *International Journal of Forecasting* 37. no. 4 (2021), pp. 1748–1764. ISSN: 0169-2070. doi: <https://doi.org/10.1016/j.ijforecast.2021.03.012>.
- [177] HSU, L.-T.: What are the roles of artificial intelligence and machine learning in GNSS positioning?. In: *Inside GNSS* (2020), pp. 20–27.
- [178] SUN, R. ; HSU, L.-T. ; XUE, D. ; ZHANG, G. ; OCHIENG, W. Y.: GPS Signal Reception Classification Using Adaptive Neuro-Fuzzy Inference System. In: *The Journal of Navigation* 72. no. 3 (2019), pp. 685–701. doi: 10.1017/S0373463318000899.
- [179] HSU, L.-T.: GNSS multipath detection using a machine learning approach. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. doi: 10.1109/ITSC.2017.8317700.
- [180] SUZUKI, T. ; AMANO, Y.: NLOS Multipath Classification of GNSS Signal Correlation Output Using Machine Learning. In: *Sensors* 21. no. 7 (2021). ISSN: 1424-8220. doi: 10.3390/s21072503.
- [181] QUAN, Y. ; LAU, L. ; ROBERTS, G. W. ; MENG, X. ; ZHANG, C.: Convolutional Neural Network Based Multipath Detection Method for Static and Kinematic GPS High Precision Positioning. In: *Remote Sensing* 10. no. 12 (2018). ISSN: 2072-4292.

-
- [182] ZAWISLAK, R. ; GREIFF, M. ; KIM, K. ; BERNTORP, K. ; DI CAIRANO, S. ; KONISHI, M. ; PARSONS, K. ; V. ORLIK, P. ; SATO, Y.: GNSS Multipath Detection Aided by Unsupervised Domain Adaptation. In: *Proc. of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)* (2022), pp. 2127–2137. DOI: 10.33012/2022.18333.
 - [183] SIEMURI, A. ; SELVAN, K. ; KUUSNIEMI, H. ; VALISUO, P. ; ELMUSRATI, M. S.: A Systematic Review of Machine Learning Techniques for GNSS Use Cases. In: *IEEE Transactions on Aerospace and Electronic Systems* 58. no. 6 (2022), pp. 5043–5077. DOI: 10.1109/TAES.2022.3219366.
 - [184] BAI, X. ; WEN, W. ; ZHANG, G. ; NG, H.-F. ; HSU, L.-T.: GNSS Outliers Mitigation in Urban Areas Using Sparse Estimation Based on Factor Graph Optimization. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022, pp. 197–202. DOI: 10.1109/ITSC55140.2022.9921906.
 - [185] LI, Y. ; ZHU, Z. ; KONG, D. ; HAN, H. ; ZHAO, Y.: EA-LSTM: Evolutionary attention-based LSTM for time series prediction. In: *Knowledge-Based Systems* 181 (2019), p. 104785. ISSN: 0950-7051. DOI: <https://doi.org/10.1016/j.knosys.2019.05.028>.
 - [186] VASWANI, A. ; SHAZEER, N. ; PARMAR, N. ; USZKOREIT, J. ; JONES, L. ; GOMEZ, A. N. ; KAISER, Ł. ; POLOSUKHIN, I.: Attention is All you Need. In: *Advances in Neural Information Processing Systems*. Ed. by GUYON, I. ; LUXBURG, U. V. ; BENGIO, S. ; WALLACH, H. ; FERGUS, R. ; VISHWANATHAN, S. ; GARNETT, R. Vol. 30. Curran Associates, Inc., 2017.
 - [187] GOGLIETTINO, G. ; RENNA, M. ; PISONI, F. ; DI GRAZIA, D. ; PAU, D.: A Machine Learning Approach to GNSS Functional Safety. In: *Proc. of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)* (2019), pp. 1738–1752. DOI: 10.33012/2019.17001.
 - [188] GOODE, M. ; EDWARDS, S. ; MOORE, P.: Time correlation in GNSS precise point positioning. In: *Proc. of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)* (Sept. 2013), pp. 1207–1214.
 - [189] GRAVES, A. ; FERNÁNDEZ, S. ; SCHMIDHUBER, J.: Bidirectional LSTM networks for improved phoneme classification and recognition. In: *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005: 15th International Conference, Warsaw, Poland, September 11–15, 2005. Proceedings, Part II* 15. Springer. 2005, pp. 799–804.
 - [190] BATUWITA, R. ; PALADE, V.: “Class Imbalance Learning Methods for Support Vector Machines”. In: *Imbalanced Learning*. John Wiley & Sons, Ltd, 2013. Chap. 5, pp. 83–99. ISBN: 9781118646106. DOI: <https://doi.org/10.1002/9781118646106.ch5>.

- [191] BROSSARD, M. ; BARRAU, A. ; CHAUCHAT, P. ; BONNABEL, S.: Associating Uncertainty to Extended Poses for on Lie Group IMU Preintegration With Rotating Earth. In: *IEEE Transactions on Robotics* 38. no. 2 (2022), pp. 998–1015. DOI: 10 . 1109/TRO . 2021 . 3100156.
- [192] KELLY, J.: *Making Space for Time: The Special Galilean Group and Its Application to Some Robotics Problems*. 2024. arXiv: 2409 . 14276 [cs . RO].
- [193] PFEIFER, T. ; LANGE, S. ; PROTZEL, P.: Advancing Mixture Models for Least Squares Optimization. In: *IEEE Robotics and Automation Letters* 6. no. 2 (2021), pp. 3941–3948. DOI: 10 . 1109/LRA . 2021 . 3067307.
- [194] KOROTKINE, V. ; COHEN, M. ; FORBES, J. R.: A Hessian for Gaussian Mixture Likelihoods in Nonlinear Least Squares. In: *IEEE Robotics and Automation Letters* 9. no. 9 (2024), pp. 7891–7898. DOI: 10 . 1109/LRA . 2024 . 3432350.
- [195] MURPHY, K. P.: *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020.
- [196] LUNZE, J.: *Regelungstechnik 2 : Mehrgrößensysteme, Digitale Regelung*PDF; 7., überarbeitete Aufl. 2013. Springer-Lehrbuch. Berlin, Heidelberg: Springer, 2013, Online–Ressource (XXV, 672 S, digital. ISBN: 978-3-642-29562-1. DOI: 10 . 1007 / 978 - 3 - 642 - 29562 - 1.
- [197] MOORE, R. E.: *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis. Department of Computer Science, Stanford University, 1962.
- [198] VAN DER MERWE, R.: *Sigma-point kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis. 2004. DOI: 10 . 6083 / M4Z60KZ5.
- [199] HAYKIN, S. S.: *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009.
- [200] WALKER, A. M.: On the Asymptotic Behaviour of Posterior Distributions. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 31. no. 1 (1969), pp. 80–88. DOI: <https://doi.org/10.1111/j.2517-6161.1969.tb00767.x>.
- [201] DIACONIS, P. ; YLVISAKER, D.: Conjugate Priors for Exponential Families. In: *The Annals of Statistics* 7. no. 2 (1979), pp. 269–281. DOI: 10 . 1214 / aos / 1176344611.
- [202] HEARD, N.: *An introduction to Bayesian inference, methods and computation*. eng. Cham, Switzerland: Springer, 2022 - 2022. ISBN: 9783030828103.
- [203] NOCEDAL, J. ; WRIGHT, S.: *Numerical optimization*. 2. ed. Springer series in operations research and financial engineering. Springer, 2006. ISBN: 978-0-387-30303-1.
- [204] GOLUB, G. H. ; LOAN, C. F. van: *Matrix Computations*. Fourth. JHU Press, 2013. ISBN: 1421407949 9781421407944.
- [205] MADSEN, K. ; NIELSEN, H. ; TINGLEFF, O.: *Methods for Non-Linear Least Squares Problems (2nd ed.)* English. 2004.

-
- [206] MARQUARDT, D. W.: An algorithm for least-squares estimation of nonlinear parameters. In: *SIAM Journal on Applied Mathematics* 11. no. 2 (1963), pp. 431–441. DOI: 10 . 1137 / 0111030.
- [207] KALMAN, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME–Journal of Basic Engineering* 82. no. Series D (1960), pp. 35–45.
- [208] SIMON, D.: *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. USA: Wiley-Interscience, 2006. ISBN: 0471708585.
- [209] JULIER, S. ; UHLMANN, J.: Unscented filtering and nonlinear estimation. In: *Proceedings of the IEEE* 92. no. 3 (2004), pp. 401–422. DOI: 10 . 1109 / JPROC . 2003 . 823141.
- [210] LABBE, R.: *Kalman and Bayesian Filters in Python*. <https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python>. 2014.
- [211] JULIER, S.: The spherical simplex unscented transformation. In: *Proceedings of the 2003 American Control Conference, 2003*. Vol. 3. 2003, 2430–2434 vol.3. DOI: 10 . 1109 / ACC . 2003 . 1243439.
- [212] RYU, K. ; KANG, J. ; LEE, D.: Performance Comparison between EKF and UKF in GPS/INS Low Observability Conditions. In: *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. 2021, pp. 1911–1916. DOI: 10 . 23919 / ICCAS52745 . 2021 . 9649748.
- [213] GILLIJNS, S. ; MENDOZA, O. ; CHANDRASEKAR, J. ; DE MOOR, B. ; BERNSTEIN, D. ; RIDLEY, A.: What is the ensemble Kalman filter and how well does it work?. In: *2006 American Control Conference*. 2006, 6 pp.-. DOI: 10 . 1109 / ACC . 2006 . 1657419.
- [214] BONNABLE, S. ; MARTIN, P. ; SALAÜN, E.: Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. 2009, pp. 1297–1304. DOI: 10 . 1109 / CDC . 2009 . 5400372.
- [215] SÄRKKÄ, S.: *Bayesian Filtering and Smoothing*. USA: Cambridge University Press, 2013. ISBN: 1107619289.
- [216] KOLLER, D. ; FRIEDMAN, N.: *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 0262013193.
- [217] KAESSE, M. ; RANGANATHAN, A. ; DELLAERT, F.: iSAM: Incremental Smoothing and Mapping. In: *IEEE Trans. on Robotics* 24. no. 6 (2008), pp. 1365–1378. DOI: 10 . 1109 / TRO . 2008 . 2006706.
- [218] KAESSE, M. ; ILA, V. ; ROBERTS, R. ; DELLAERT, F.: The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping. In: *Springer Tracts in Advanced Robotics*. Vol. 68. Springer, 2010, pp. 157–173.

- [219] DAVIS, T. A. ; GILBERT, J. R. ; LARIMORE, S. I. ; NG, E. G.: A column approximate minimum degree ordering algorithm. In: *ACM Transactions on Mathematical Software (TOMS)* 30. no. 3 (2004), pp. 353–376.
- [220] SUBIRANA, J. ; ZORNOZA, J. ; HERNÁNDEZ-PAJARES, M. ; AGENCY, E. S. ; FLETCHER, K.: *GNSS Data Processing*. ESA TM Bd. 1. ESA Communications, 2013. ISBN: 9789292218867.
- [221] HEIKKINEN, M.: Geschlossene Formeln zur Berechnung rauemlicher geodaetischer Koordinaten aus rechtwinkligen Koordinaten. In: *Zeitschrift fuer Vermessungswessen* 107 (1982), pp. 207–211.
- [222] ZHU, J.: Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates. In: *IEEE Trans. Aerosp. Electron. Syst.* 30. no. 3 (1994), pp. 957–961. DOI: 10 . 1109 / 7 . 303772.
- [223] IMAGERY, N. ; AGENCY, M.: United States Department of Defense, World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. In: *Tech. Rep., TR8350.2* (Jan. 2000).
- [224] LONG, A. W. ; WOLFE, K. C. ; MASHNER, M. ; CHIRIKJIAN, G. S.: The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates.. In: *Robotics: Science and Systems*. 2012. ISBN: 978-0-262-51968-7.
- [225] HOWE, R.: Very basic Lie theory. In: *The American Mathematical Monthly* 90. no. 9 (1983), pp. 600–623.
- [226] SOLÀ, J. ; DERAY, J. ; ATCHUTHAN, D.: *A micro Lie theory for state estimation in robotics*. Dec. 2018. DOI: 10 . 48550 / arXiv . 1812 . 01537.
- [227] MURRAY, R. M. ; SASTRY, S. S. ; ZEXIANG, L.: *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc., 1994. ISBN: 0849379814.
- [228] BUCHANAN, R.: *Allan Variance ROS*. Version 1.2. Nov. 2021.
- [229] REHDER, J. ; NIKOLIC, J. ; SCHNEIDER, T. ; HINZMANN, T. ; SIEGWART, R.: Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4304–4311. DOI: 10 . 1109 / ICRA . 2016 . 7487628.
- [230] WENDEL, J.: *Integrierte Navigationssysteme Sensordatenfusion, GPS und Inertiale Navigation*. München: Oldenbourg Wissenschaftsverlag, 2011. ISBN: 9783486705720. DOI: doi : 10 . 1524 / 9783486705720.
- [231] FARRELL, J.: *Aided Navigation: GPS with High Rate Sensors*. 1st ed. USA: McGraw-Hill, Inc., 2008. ISBN: 0071493298.
- [232] LUPTON, T. ; SUKKARIEH, S.: Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions. In: *IEEE Trans. on Robot.* 28. no. 1 (2012), pp. 61–76. DOI: 10 . 1109 / TRO . 2011 . 2170332.

-
- [233] FORSTER, C. ; CARLONE, L. ; DELLAERT, F. ; SCARAMUZZA, D.: On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. In: *IEEE Trans. on Robot.* PP (Aug. 2016). DOI: 10.1109/TRO.2016.2597321.
 - [234] TEUNISSEN, P. ; MONTENBRUCK, O.: *Springer Handbook of Global Navigation Satellite Systems*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 3319429264.
 - [235] ZHU, N. ; MARAIS, J. ; BÉTAILLE, D. ; BERBINEAU, M.: GNSS Position Integrity in Urban Environments: A Review of Literature. In: *IEEE Trans. Intell. Transp. Syst.* 19. no. 9 (2018), pp. 2762–2778. DOI: 10.1109/TITS.2017.2766768.
 - [236] LEE, D. ; JUNG, M. ; YANG, W. ; KIM, A.: LiDAR odometry survey: recent advancements and remaining challenges. In: *Intelligent Service Robotics* 17. no. 2 (Mar. 2024), pp. 95–118. ISSN: 1861-2784. DOI: 10.1007/s11370-024-00515-8.
 - [237] YIN, H. ; XU, X. ; LU, S. ; CHEN, X. ; XIONG, R. ; SHEN, S. ; STACHNISS, C. ; WANG, Y.: A Survey on Global LiDAR Localization: Challenges, Advances and Open Problems. In: *Int. J. Comput. Vision* 132. no. 8 (Mar. 2024), pp. 3139–3171. ISSN: 0920-5691. DOI: 10.1007/s11263-024-02019-5.
 - [238] SEGAL, A. ; HÄHNEL, D. ; THRUN, S.: Generalized-ICP.. In: *Robotics: Science and Systems*. The MIT Press, 2009. ISBN: 978-0-262-51463-7.
 - [239] BIBER, P. ; STRASSER, W.: The normal distributions transform: a new approach to laser scan matching. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 3. 2003, 2743–2748 vol.3. DOI: 10.1109/IROS.2003.1249285.
 - [240] ZHANG, J. ; SINGH, S.: Low-drift and Real-time LiDAR Odometry and Mapping. In: *Auton. Robots* 41 (Feb. 2017), pp. 401–416. DOI: 10.1007/s10514-016-9548-2.
 - [241] SHAN, T. ; ENGLLOT, B.: LeGO-LOAM: Lightweight and Ground-Optimized LiDAR Odometry and Mapping on Variable Terrain. In: *2018 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2018, pp. 4758–4765. DOI: 10.1109/IROS.2018.8594299.
 - [242] SCHÜTZ, A. ; SÁNCHEZ-MORALES, D. E. ; PANY, T.: Precise positioning through a loosely-coupled sensor fusion of GNSS-RTK, INS and LiDAR for autonomous driving. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2020, pp. 219–225. DOI: 10.1109/PLANS46316.2020.9109934.
 - [243] LEE, S. H. ; CIVERA, J.: Loosely-Coupled Semi-Direct Monocular SLAM. In: *IEEE Robotics and Automation Letters* 4. no. 2 (2019), pp. 399–406. DOI: 10.1109/LRA.2018.2889156.
 - [244] WU, Y. ; GUADAGNINO, T. ; WIESMANN, L. ; KLINGBEIL, L. ; STACHNISS, C. ; KUHLMANN, H.: LIO-EKF: High Frequency LiDAR-Inertial Odometry using Extended Kalman Filters. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 13741–13747. DOI: 10.1109/ICRA57147.2024.10610667.

- [245] XU, W. ; ZHANG, F.: FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. In: *IEEE Robotics and Automation Letters* 6. no. 2 (2021), pp. 3317–3324. doi: 10.1109/LRA.2021.3064227.
- [246] SMITH, R. C. ; CHEESEMAN, P.: On the Representation and Estimation of Spatial Uncertainty. In: *The International Journal of Robotics Research* 5. no. 4 (1986), pp. 56–68. doi: 10.1177/027836498600500404.
- [247] CADENA, C. ; CARLONE, L. ; CARRILLO, H. ; LATIF, Y. ; SCARAMUZZA, D. ; NEIRA, J. ; REID, I. ; LEONARD, J. J.: Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. In: *IEEE Transactions on Robotics* 32. no. 6 (2016), pp. 1309–1332. doi: 10.1109/TRO.2016.2624754.
- [248] SAHILI, A. R. ; HASSAN, S. ; SAKHRIEH, S. M. ; MOUNSEF, J. ; MAALOUF, N. ; ARAIN, B. ; TAHA, T.: A Survey of Visual SLAM Methods. In: *IEEE Access* 11 (2023), pp. 139643–139677. doi: 10.1109/ACCESS.2023.3341489.
- [249] EBADI, K. ; BERNREITER, L. ; BIGGIE, H. ; CATT, G. ; CHANG, Y. ; CHATTERJEE, A. ; DENNISTON, C. E. ; DESCHÊNES, S.-P. ; HARLOW, K. ; KHATTAK, S. ; NOGUEIRA, L. ; PALIERI, M. ; PETRÁČEK, P. ; PETRLÍK, M. ; REINKE, A. ; KRÁTKÝ, V. ; ZHAO, S. ; AGHAMOHAMMADI, A.-a. ; ALEXIS, K. ; HECKMAN, C. ; KHOSOUSSI, K. ; KOTTEGE, N. ; MORRELL, B. ; HUTTER, M. ; PAULING, F. ; POMERLEAU, F. ; SASKA, M. ; SCHERER, S. ; SIEGWART, R. ; WILLIAMS, J. L. ; CARLONE, L.: Present and Future of SLAM in Extreme Environments: The DARPA SubT Challenge. In: *IEEE Transactions on Robotics* 40 (2024), pp. 936–959. doi: 10.1109/TRO.2023.3323938.
- [250] STURM, J. ; ENGELHARD, N. ; ENDRES, F. ; BURGARD, W. ; CREMERS, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. doi: 10.1109/IROS.2012.6385773.
- [251] ABU-MOSTAFA, Y. S. ; MAGDON-ISMAIL, M. ; LIN, H.-T.: *Learning From Data*. AMLBook, 2012.
- [252] BISHOP, C. M. ; BISHOP, H.: *Deep Learning - Foundations and Concepts*. Ed. by CHAM, S. 1st ed. 2023. ISBN: 978-3-031-45468-4. doi: <https://doi.org/10.1007/978-3-031-45468-4>.
- [253] CHEN, T. ; GUESTRIN, C.: Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.
- [254] GUO, R. ; ZHAO, Z. ; WANG, T. ; LIU, G. ; ZHAO, J. ; GAO, D.: Degradation state recognition of piston pump based on ICEEMDAN and XGBoost. In: *Applied Sciences* 10. no. 18 (2020), p. 6593.
- [255] NOBLE, W. S.: What is a support vector machine?. In: *Nature biotechnology* 24. no. 12 (2006), pp. 1565–1567.

- [256] BISHOP, C. M. ; NASRABADI, N. M.: *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.
- [257] GOODFELLOW, I. ; BENGIO, Y. ; COURVILLE, A.: *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016.
- [258] ROSENBLATT, F.: *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
- [259] RUDER, S.: An overview of gradient descent optimization algorithms. In: *arXiv preprint arXiv:1609.04747* (2016).
- [260] KINGMA, D. P. ; BA, J.: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014).
- [261] RUMELHART, D. E. ; HINTON, G. E. ; WILLIAMS, R. J.: Learning representations by back-propagating errors. In: *nature* 323. no. 6088 (1986), pp. 533–536.
- [262] HOCHREITER, S. ; SCHMIDHUBER, J.: Long short-term memory. In: *Neural computation* 9. no. 8 (1997), pp. 1735–1780.
- [263] HAHN, M.: Theoretical Limitations of Self-Attention in Neural Sequence Models. In: *Transactions of the Association for Computational Linguistics* 8 (Jan. 2020), pp. 156–171. ISSN: 2307-387X. DOI: 10.1162/tac1_a_00306.
- [264] ANDERSON, S. W.: *Batch Continuous-Time Trajectory Estimation*. Available at <https://hdl.handle.net/1807/77974>. PhD thesis. University of Toronto, June 2017.
- [265] AGRAWAL, M.: A Lie Algebraic Approach for Consistent Pose Registration for General Euclidean Motion. In: *2006 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2006, pp. 1891–1897. DOI: 10.1109/IROS.2006.282313.
- [266] WISTH, D. ; CAMURRI, M. ; FALLON, M.: VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots. In: *IEEE Trans. on Robot.* 39. no. 1 (2023), pp. 309–326. DOI: 10.1109/TRO.2022.3193788.
- [267] YOON, D. J. ; BURNETT, K. ; LACONTE, J. ; CHEN, Y. ; VHAVLE, H. ; KAMMEL, S. ; REUTHER, J. ; BARFOOT, T. D.: Need for Speed: Fast Correspondence-Free Lidar Odometry Using Doppler Velocity. In: *2023 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. 2023.
- [268] WONG, J. N. ; YOON, D. J. ; SCHOELLIG, A. P. ; BARFOOT, T. D.: Variational Inference With Parameter Learning Applied to Vehicle Trajectory Estimation. In: *IEEE Robot. Autom. Lett.* 5. no. 4 (2020), pp. 5291–5298. DOI: 10.1109/lra.2020.3007381.
- [269] MAC TAVISH, K. ; BARFOOT, T. D.: At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers. In: *2015 12th Conference on Computer and Robot Vision*. IEEE, 2015, pp. 62–69. ISBN: 978-1-4799-1986-4. DOI: 10.1109/CRV.2015.52.

- [270] REY, W. J. J.: *Introduction to robust and quasi-robust statistical methods*. Springer Science & Business Media, 2012.
- [271] DAS, S. ; WATSON, R. ; GROSS, J.: Review of factor graphs for robust GNSS applications. In: *arXiv preprint arXiv:2112.07794* (2021).
- [272] GOODFELLOW, I. ; BENGIO, Y. ; COURVILLE, A.: *Deep learning*. MIT press, 2016.
- [273] CHENG, Q. ; CHEN, Y. ; XIAO, Y. ; YIN, H. ; LIU, W.: A dual-stage attention-based Bi-LSTM network for multivariate time series prediction. In: *The Journal of Supercomputing* 78. no. 14 (2022), pp. 16214–16235.