


# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :**

Institut National Polytechnique de Toulouse (Toulouse INP)

**Discipline ou spécialité :**

Informatique et Télécommunication

---

**Présentée et soutenue par :**

M. ANTHONY GUILLARD

le vendredi 8 décembre 2023

**Titre :**

Détection et la correction d'erreurs de positionnement sur un récepteur GNSS de positionnement haute précision par hybridation avec d'autres capteurs et apprentissage automatique

---

**Ecole doctorale :**

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

**Unité de recherche :**

Laboratoire de Télécommunications (TELECOM-ENAC)

**Directeur(s) de Thèse :**

M. CARL MILNER

M. PAUL THEVENON

**Rapporteurs :**

M. LIANG CHEN, Université de Wuhan

**Membre(s) du jury :**

MME JULIETTE MARAIS, UNIVERSITE GUSTAVE EIFFEL, Président

M. CARL MILNER, ECOLE NATIONALE DE L'AVIATION CIVILE, Membre

M. PAUL THEVENON, ECOLE NATIONALE DE L'AVIATION CIVILE, Membre

# Abstract

Global Navigation Satellite Systems (GNSS) have undergone a boom in recent years in a wide range of applications. These applications often include use in urban environments where most of the consumers are. However, urban canyons are the most challenging conditions for a GNSS receiver to operate in without large position errors. Thanks to the increased availability of GNSS signals via new constellations(Galileo and Beidou) and new frequencies (e.g. E6), this has led to a better positioning accuracy in challenging conditions. But, the sheer nature of urban environments still leads to positioning errors despite the increased number of signals.

The presence of high obstacles such as buildings and trees can create an environment where the incoming GNSS signal is reflected, diffracted, or even blocked before reaching the antenna. This phenomenon is called multipath. The blockage of direct signals means that only reflected or diffracted signals can reach the antenna called non-line of sight signals (NLOS). On the other hand, when the original signal reaches the antenna on top of reflections and/or diffractions, this is referred to as multipath reflections which will be studied in this thesis. Multipath depends on the satellite positions and the surrounding environment; therefore, predicting its impact is near impossible for real-time systems. Multipath impacts the three GNSS observables that can be generated to estimate the receiver's position: code pseudorange, carrier phase and Doppler measurement.

In GNSS positioning, carrier phase measurements are widely used as they are much more precise than the code pseudoranges. However, these observables are ambiguous as the total number of cycles is not exactly known. To reap the accuracy benefit of these measurements, the ambiguity has to be estimated and then fixed in some cases. This mandatory process is often complicated and complexifies the positioning algorithms by including additional processing steps. Furthermore, if the lock on the phase of the signal is temporarily lost - called a cycle slip, the ambiguity must be re-estimated/fixed. In urban conditions, cycle slips frequently occur due to lower carrier to noise density ratios (CN0) or temporary signal blockage.

This thesis was part of the collaboration between the research group SIGNAV at ENAC and 3D Aerospace. The goal was to develop a light-weight GNSS receiver that functions in real-time and can operate in urban conditions. Thus, there was an emphasis on making algorithms lightweight so that they could be implemented in the real-time receiver down the line.

To alleviate some of the computational burden brought by using raw carrier phase measurements, this work presents an extended Kalman filter that uses time differenced carrier phase measurements (TDCP) tightly coupled with an inertial navigation system (INS). TDCPs are the difference of two consecutive carrier phase measurements. In the absence of

cycle slips, the ambiguity term remains constant so the difference effectively removes it. The proposed filter is 57% faster than a filter that estimates the ambiguities due to its much smaller state vector size. Furthermore, as there is no convergence period on the ambiguities, the TDCP filter performs 5% and 17% better for the 68% and 95% horizontal errors than the ambiguity estimation filter in urban conditions.

It was observed on the TDCP filter that its performance is still heavily reliant on the accuracy of code pseudoranges. Yet, this measurement is the most impacted by multipath out of the three. Therefore, another goal of this thesis was to detect when multipath reflections affected the code pseudoranges. As convolutional neural networks (CNN) can be very efficient in finding features that are hard to model, they were used in this work as a binary classifier to detect multipath. The CNN inputs are obtained thanks to correlator output values as a function of their delay (with respect to the prompt correlator) and as a function of time yielding a 2D image. The presented model has a F score of 94.7% on Galileo E1-B and 91.6% on GPS L1 C/A. To reduce the computational load, the number of correlator outputs and correlator sampling frequency was then decreased by a factor of 4, and the CNN still had a F score of 91.8% on Galileo E1-B and 90.5% on GPS L1 C/A.

To take advantage of the detected multipath rays from the CNN, this work proposes an algorithm that fuses a weighted least squares estimator (WLSE) with a robust MM estimator. WLSE estimators can be severely impacted by outliers but perform optimally in ideal conditions. On the other hand, robust MM estimators can mitigate the impact of outlier but suffer from a loss of efficiency when no outliers are present. Hence, a hybrid positioning solution is investigated: when multipath is detected by the CNN, the robust MM estimator is used and when no multipath is detected the WLSE algorithm is used. For the presented data, it was shown to improve the horizontal positioning accuracy by 20% and 11% for 20% and 60% of the time.

# Acknowledgments

I believe the acknowlegdments are the only place in my thesis where I do not have to be technical and where I can let my writing be less homogeneous. This thesis has been a three year battle with myself to keep pushing despite all the setbacks and obstacles.

The truth is I would not have made it without the help of Paul Thevenon and Carl Milner, my thesis directors. Thank you for your pedagogy, sound advice, and support. A PhD is never an easy task, but it can be much easier with great PhD supervisors.

I would also like to thank Benjamin Kawak who hired me at 3D Aerospace and gave me the opportunity to pursue a PhD in his company. From being the first intern, to first employee to first PhD student, we have come a long way. I wish all the best to him and 3D Aerospace for the future.

I express my gratitude and thanks to the jury of this thesis for reviewing the work and their valuable feedback for the improvement of this thesis. Many thanks to everyone who attended the PhD defense.

Thank you to all the 3D Aerospace employees/interns that have worked with me and made everyday life at the office enjoyable. A special mention to Estelle who endured my passion for debating about everything and anything on a daily basis. There are some that have only passed by the office for a short 6 months, but in that short time have made working a blast: Théo, Lucie, and Alexis.

I would also like to thank all of my ENAC colleagues that have welcomed me with open arms despite my few appearances at ENAC. The Telecom drinking group has become a monthly habit that will be tough to beat in future working groups! Many thanks to all of you. This unhealthy habit of bar hopping was counterbalanced with the Telecom Running group. Even though our group was much smaller, it was very pleasant to race alongside you! Many thanks to Jan for his knowledge about EKFs but more importantly the good times we had collecting data and outside the office.

Last but definitely not least, my uttermost gratitude to my family for their unlimited support and reading my articles despite not being familiar with my topic. I had the chance of having great friends to rely on in the hardest times of my PhD, so know that your support means the world to me even when I was despicable. Thank you to my girlfriend for bearing with me when I was at my grumpiest or spent nights working on my research.

I could not include everyone's name that impacted my life these past three years in my

acknowledgements but know that any quality time spent together means you contributed to helping me to finish this thesis.

# Abbreviations

| Notation | Description |
|---|---|
| ACF | Autocorrelation Function |
| ADC | Analog to Digital Converter |
| ANN | Artificial Neural Network |
| AWGN | Additive White Gaussian Noise |
| CDF | Cumulative Distributive Function |
| CN0 | Carrier to Noise density ratio |
| CNN | Convolutional Neural Network |
| CPU | Computational Power Unit |
| DLL | Delay Lock Loop |
| ECEF | Earth Centered Earth Fixed |
| EKF | Extended Kalman Filter |
| EL | Early Late |
| EMLP | Early Minus Late Power |
| FLL | Frequency Lock Loop |
| GAN | Generative Adversarial Network |
| GNSS | Global Navigation Satellite Systems |
| GPST | Global Positioning System Time |
| IMU | Inertial Measurement Unit |
| INS | Inertial Navigation Solution |
| KF | Kalman Filter |
| LOS | Line of Sight |
| LSE | Least Squares Estimation |
| MAD | Median Absolute Deviation |
| MEE | Maximum Error Envelope |

| Notation | Description |
|---|---|
| MEMS | Microelectromechanical Systems |
| ML | Machine Learning |
| MLE | Maximum Likelihood Estimators |
| NCO | Numerically Controlled Oscillator |
| NED | North East Down |
| NLOS | Non-Line of Sight |
| PLL | Phase Lock Loop |
| PPP | Precise Point Positioning |
| PRN | Pseudo Random Noise |
| PVT | Position Velocity Time |
| RAIM | Receiver Autonomous Integrity Monitoring |
| ReLU | Rectified Linear Unit |
| RNN | Reccurent Neural Network |
| SDR | Software Defined Radio |
| SGD | Stochastic Gradient Descent |
| SVM | Support Vector Machine |
| TDCP | Time Differenced Carrier Phase |
| WLSE | Weighted Least Squares Estimation |

# Contents

**Abstract** i

**Acknowledgments** iii

**Abbreviations** v

**Contents** vii

**List of Figures** xi

**List of Tables** xiv

**1 Introduction** 1
- 1.1 Thesis background and motivation . . . 1
- 1.2 Thesis objectives . . . 4
- 1.3 Thesis contributions . . . 5
- 1.4 Thesis outline . . . 6

**2 GNSS Receiver Processing and Observables** 7
- 2.1 Signal Structure . . . 8
  - 2.1.1 Data Bits . . . 8
  - 2.1.2 Pseudo-Random Noise Code . . . 8
  - 2.1.3 Carrier Wave . . . 9
  - 2.1.4 IF Signal Model . . . 10
- 2.2 Signal Conditioning and Digitization by the RF front-end . . . 10
  - 2.2.1 Filtering and Amplification . . . 11
  - 2.2.2 Down Conversion . . . 12
  - 2.2.3 Automatic Gain Control and Analog to Digital Converter . . . 12
- 2.3 Acquisition . . . 13
  - 2.3.1 Generating in-phase and quadrature phase correlator outputs . . . 13
  - 2.3.2 Decision . . . 15
  - 2.3.3 Alternate Methods . . . 16
- 2.4 Tracking . . . 17
  - 2.4.1 Carrier frequency offset (Doppler shift) and phase . . . 18
  - 2.4.2 Code Delay . . . 21
  - 2.4.3 Effect of Multipath . . . 24

2.4.4 Alternate Methods . . . 26
2.5 Observables Generation . . . 27
2.5.1 Code Pseudorange . . . 28
2.5.2 Carrier Phase . . . 30
2.5.3 Doppler . . . 31
2.5.4 Effect of Multipath . . . 31
2.6 Conclusions . . . 34

**3 GNSS Positioning Algorithms** **36**
3.1 Positioning Principles . . . 36
3.2 Least Squares Estimator . . . 38
3.2.1 Classical Estimator . . . 38
3.2.2 Weighted Least Squares Estimator . . . 41
3.2.3 Dilution of Precision . . . 42
3.3 Kalman Filter . . . 43
3.3.1 GNSS Only . . . 46
3.3.2 GNSS/INS Hybridization . . . 49
3.4 Robust Estimator . . . 54
3.4.1 M Estimator . . . 55
3.4.2 S Estimator . . . 56
3.4.3 MM Estimator . . . 57
3.5 Conclusions . . . 57

**4 Convolutional Neural Networks** **58**
4.1 Differences with Artificial Neural Networks . . . 59
4.2 Layers . . . 60
4.2.1 Convolutional Layer . . . 60
4.2.2 Pooling Layer . . . 62
4.2.3 Output Layer . . . 62
4.3 Predictions and Training . . . 63
4.4 Weight Update and Gradient Descent . . . 64
4.5 Overfitting . . . 67
4.5.1 Dropout . . . 67
4.5.2 Regularization . . . 68
4.6 Conclusions . . . 69

**5 Proposed TDCP based EKF** **70**
5.1 Formation of TDCP Measurements . . . 71
5.2 State of the Art EKF/PPP Algorithms . . . 72
5.3 Adaptation of the EKF . . . 73
5.3.1 Measurement correlation . . . 74
5.3.2 Adapted EKF Equations . . . 74
5.3.3 Error State vector . . . 75
5.3.4 State Transition Model and State Process Noise Matrix . . . 76
5.3.5 Observation Model . . . 76
5.4 Cycle Slip Detection . . . 77
5.4.1 EKF based Cycle Slip Rejection . . . 78
5.4.2 Wavelength based Cycle Slip Rejection . . . 78

5.5 Method and Scenarios . . . 79
5.5.1 Synchronization of sensors . . . 80
5.5.2 Reference Trajectory . . . 81
5.5.3 Differences between Datasets . . . 82
5.6 Results . . . 83
5.6.1 IGS Dataset . . . 84
5.6.2 SPAN IMU Datasets . . . 86
5.6.3 MEMS IMU Datasets . . . 92
5.7 Conclusions . . . 99

**6 CNN Based Multipath Detector** **100**
6.1 State of the Art Multipath Detection . . . 101
6.2 Data Labeling Classification . . . 102
6.3 Method and Scenarios . . . 107
6.3.1 Static Dataset . . . 107
6.3.2 Dynamic Dataset . . . 108
6.4 Signal Deformation . . . 109
6.5 Inputs . . . 109
6.5.1 Cross Correlation . . . 110
6.5.2 Correlator Configurations . . . 111
6.6 CNN Architecture . . . 112
6.7 Results . . . 113
6.7.1 Galileo E1-B . . . 115
6.7.2 GPS L1 C/A . . . 117
6.8 Conclusions . . . 120

**7 Positioning Performance of Adapted Robust Estimator** **122**
7.1 State of the Art Robust Estimation . . . 123
7.2 Algorithm Implementation . . . 124
7.3 Variation of Robust Estimator Parameters . . . 125
7.3.1 Variable efficiency . . . 125
7.3.2 Variable breakdown point . . . 127
7.4 Method and Scenarios . . . 128
7.5 Results . . . 129
7.5.1 No Parameter Variation . . . 130
7.5.2 Robust estimator parameter variation . . . 133
7.6 Conclusions . . . 134

**8 Conclusions and Perspectives** **136**
8.1 Conclusions . . . 136
8.2 Perspectives . . . 139

**A EKF State and Observation Models** **141**
A.1 State Transition Model . . . 141
A.2 Observation Model . . . 142
A.2.1 Code Pseudorange and Carrier Phase . . . 142
A.2.2 Doppler . . . 143

**B Median Absolute Deviation Relationship with Standard Deviation for Gaussian Laws** **145**

**C Different CNN Architecture Results** **147**
C.1 Overly Complex Model . . . . . 147
C.2 Overfitting model . . . . . 148

**Bibliography** **149**

# List of Figures


2.1 GNSS Signal Elements (not to scale). . . . . 8
2.2 Illustration of a GNSS RF front-end. . . . . 11
2.3 Illustration of an ideal bandpass filter. . . . . 11
2.4 Generation of I and Q correlator outputs during the acquisition process. . . . 14
2.5 Acquisition Grid. . . . . 15
2.6 Parallel Doppler Search Method. . . . . 17
2.7 Parallel Code Search Method. . . . . 17
2.8 Phase Lock Loop. . . . . 18
2.9 Arctangent discriminator linearity region for several CN0s [Foucras et al., 2014]. 20
2.10 Delay Lock Loop. . . . . 21
2.11 EMLP Discriminator with no error. . . . . 23
2.12 Decomposition of a single multipath ray on top of the direct signal. . . . . 25
2.13 Early late power discriminator function affected by some multipath. . . . . 26
2.14 Common Reception Technique illustration [Marco Rao, 2012]. . . . . 29
2.15 MEE of code pseudorange for two different EL spacing with an $\alpha = 0.5$. . . . 33
2.16 Simplified MEE of carrier phase for an $\alpha = 0.5$. . . . . 34

3.1 Tri-lateration for GNSS purposes. . . . . 37
3.2 Kalman Filter steps during one iteration. . . . . 44
3.3 Different GNSS/INS hybridization techniques. . . . . 50
3.4 M Estimator Algorithm for GNSS positioning. . . . . 56

4.1 Artificial neural network architecture example. . . . . 59
4.2 CNN Architecture Example. . . . . 60
4.3 Convolution illustration. . . . . 61
4.4 Max Pooling and Average Pooling example. . . . . 62
4.5 Gradient Descent technique for low and high learning rates. . . . . 65
4.6 Dropout example for a flattened CNN. . . . . 67

5.1 Formation of TDCP Measurements. . . . . 71
5.2 EKF Filter Architecture for one epoch. . . . . 73
5.3 Hypothesis Test for cycle slips. . . . . 77
5.4 GNSS and IMU Synchronization Set-up. . . . . 80
5.5 Map reference trajectory for the static dataset of the 4th of October 2021. . 82
5.6 Map reference trajectory for the open-sky dataset of the 7th of October 2020. 82

5.7 Map reference trajectory for the sub urban dataset of the 16th of December 2020. . . . 82
5.8 Map reference trajectory for the deep urban dataset of the 27th of June 2023. 82
5.9 Number of satellites as a function of time. . . . 84
5.10 IGS time series v3. . . . 84
5.11 IGS time series v3TDCP. . . . 84
5.12 IGS CDFs. . . . 85
5.13 Number of satellites as a function of time. . . . 86
5.14 Highway Time series v3. . . . 86
5.15 Highway Time series v3TDCP. . . . 86
5.16 Highway CDFs (with an uncombined TDCP filter with EKF based rejection). 87
5.17 Number of satellites as a function of time. . . . 88
5.18 Sub-Urban Time series v3. . . . 88
5.19 Sub-Urban Time series v3TDCP. . . . 88
5.20 Sub-Urban CDFs (with an uncombined TDCP filter with EKF based rejection). 89
5.21 Number of satellites as a function of time. . . . 90
5.22 Urban Time series v3. . . . 90
5.23 Urban Time series v3TDCP. . . . 90
5.24 Urban CDFs (with an uncombined TDCP filter with EKF based rejection). . 91
5.25 Number of satellites as a function of time. . . . 93
5.26 Highway Time series v3. . . . 93
5.27 Highway Time series v3TDCP. . . . 93
5.28 Highway CDFs. . . . 94
5.29 Number of satellites as a function of time. . . . 95
5.30 Sub-Urban Time series v3. . . . 95
5.31 Sub-Urban Time series v3TDCP. . . . 95
5.32 Sub-urban CDFs. . . . 96
5.33 Number of satellites as a function of time. . . . 97
5.34 Urban Time series v3. . . . 97
5.35 Urban Time series v3TDCP. . . . 97
5.36 Urban CDFs. . . . 98

6.1 Multipath error on the pseudorange vs tracking bias error as a function of time on synthetic data. . . . 103
6.2 Multipath Error Enveloppe for a brick induced multipath for an E-L spacing of 0.04 chips on Galileo E1-B (left) and for an E-L spacing of 0.125 chips on GPS L1 C/A (right). . . . 105
6.3 EMLP discriminator affected by some multipath fully sampled (blue) and under sampled (red). . . . 106
6.4 Static Distribution of Multipath Decision for Galileo E1-B. . . . 108
6.5 Static Distribution of Multipath Decision for GPS L1 C/A. . . . 108
6.6 PRN Deformation Correction Algorithm. . . . 109
6.7 Filtered (black) and non filtered (blue) tracking bias with its multipath bounds (red) for Galileo E1-B on PRN 3 (left) and their corresponding CNN inputs (right). . . . 110
6.8 Cross correlation factor of a sample of the filtered tracking bias over 5 seconds. 111
6.9 Illustration of some correlator variations for GPS L1 C/A. . . . 112

6.10 CNN Architecture for Multipath Detection. . . . . 113
6.11 Accuracy and loss values over training iterations for training and validation datasets. . . . . 114
6.12 E1 Metrics of the CNN. . . . . 116
6.13 L1 Metrics of the CNN. . . . . 118

7.1 Robust Estimator Fusion Algorithm Diagram for one epoch. . . . . 124
7.2 Tukey and LSE bi-weight loss functions (left) and weight functions (right) for different values of $c$. . . . . 126
7.3 IFEN Data set-up to record data. . . . . 129
7.4 Number of satellites used over the concatenated data collects. . . . . 130
7.5 Number of detected multipath affected satellites over the concatenated data collects. . . . . 130
7.6 CDF of the WLSE, MM estimator, and proposed MM + WLSE mix. . . . . 131
7.7 Normalized horizontal error as a function of the percentage of urban data used with respect to open-sky data for the WLSE, MM estimator, and MM+WLSE algorithms for the 20-th, 50-th, 68-th, and 95-th quantile. . . . . 132

C.1 Complex CNN Architecture for Multipath Detection. . . . . 147
C.2 Overfitting CNN Architecture for Multipath Detection. . . . . 148

# List of Tables


2.1 Neyman-Pearson Test Probabilities . . . . . 16

5.1 Hypothesis Test Probabilities . . . . . 77
5.2 IGS CDF results for all configurations. . . . . 86
5.3 Highway CDF results for all configurations. . . . . 87
5.4 Sub-urban CDF results for all configurations. . . . . 90
5.5 Urban CDF results for all configurations. . . . . 92
5.6 Highway CDF results for all configurations. . . . . 94
5.7 Sub-urban CDF results for all configurations. . . . . 96
5.8 Urban CDF results for all configurations. . . . . 98

6.1 Reflection and attenuation factors for common urban surfaces at normal incidence on L1 frequency . . . . . 104
6.2 IFEN SX3 Settings for GPS L1 C/A and Galileo E1-B. . . . . 109
6.3 Correlator configurations for GPS L1 C/A and Galileo E1-B. . . . . 112
6.4 Confusion Matrix . . . . . 114
6.5 Normalized Mean Execution time for all configurations. . . . . 116
6.6 Mean Execution time for all configurations. . . . . 118

7.1 IFEN SX3 Settings for GPS L1 C/A and Galileo E1-B. . . . . 128
7.2 CDF results for all algorithm parameter variations. . . . . 134

C.1 CNN Performance Metric on Galileo E1-B for 101@200 with a complex CNN model. . . . . 147
C.2 CNN Performance Metric on Galileo E1-B for 101@200 with an overfitting CNN model. . . . . 148

CHAPTER 1

# Introduction

## 1.1 Thesis background and motivation

This PhD was funded by 3D Aerospace; a start-up currently developing a multi-frequency, multi-constellation (GPS + Galileo) receiver. This receiver will serve as one of the main bricks of a positioning engine called the eHermes. This system has two targeted applications. One is to detect various anomalies in the vineyard. These include detecting missing or sick plants. The system also aims at estimating the wine yield from the detected grapes using several onboard cameras. The other targeted application by the eHermes is to detect potential potholes and other road anomalies in urban environments. This PhD was conducted with a focus on the latter. The whole global navigation satellite system (GNSS) signal processing chain was developed by myself for 3D Aerospace, so there was a large flexibility on the data being used for research purposes. On the other hand, as the eHermes needs to operate in real time, there was an emphasis on having light-weight algorithms.

The receiver is required to operate in cities; therefore, a heavy focus of this PhD was placed on designing and testing algorithms in urban conditions. GNSS in urban conditions is one of the most researched topics due to the difficulty of modeling all errors affecting the received signals. One of the most difficult errors to model is multipath due its dependency on the surrounding environment.

The baseline receiver was a tight coupling between an inertial measurement unit and GNSS observables using an extended Kalman filter and using precise point positioning (PPP) corrections. Improving the position solution of this baseline receiver in urban conditions despite the presence of multipath was targeted while keeping the computational complexity of algorithms to a minimum. To do so, two main research axes were identified:

- Using a combination of consecutive carrier phase measurements in a PPP algorithm.
- Mitigation of the multipath effect in the positioning algorithm.

For low cost systems that are not connected to any real time kinematics correction provider, PPP based solutions are often the preferred choice. Thanks to the use of multi constellation and frequency observations, carrier phase measurements, and satellite corrections, the accuracy of position velocity time (PVT) solutions is increased with respect to standard precision positioning solutions. A common approach to PPP solutions is to use an extended Kalman filter (EKF). A Kalman filter (KF) fuses a statistical model of the state evolution

with observations and the corresponding model to estimate the chosen states.

In GNSS positioning, carrier phase measurements are widely used as they are much more precise than the code pseudoranges. Thus, the EKF can be configured to estimate the ambiguity of each carrier phase measurement as done in [Li, 2019] [Duong, 2019]. However, by estimating the ambiguity for each measurement in an Extended Kalman Filter, this augments the size of the state vector. The benefit of using carrier phase measurements, while keeping a low state vector size, can be obtained by using Time Differenced Carrier Phase (TDCP) measurements [Kim et al., 2020], [Soon et al., 2008], and [Kai et al., 2021]. TDCPs are formed by differencing two consecutive carrier phase observables which effectively removes the ambiguous term if no cycle slips are present. This yields a more lightweight algorithm but at the expense of absolute positioning.

To increase the robustness of the navigation solution and mitigate the downsides of GNSS signals, inertial measurement units (IMU) are often combined with GNSS observations. IMUs measure the acceleration and angular rate. Their performance does not depend on the surrounding environment and have great short-term accuracy making them especially useful in challenging GNSS conditions. This comes at the expense of additional states to account for - attitude angles and IMU biases - and initialization processes. In this work, the IMU was tightly coupled to the GNSS solution in the EKF.

Even though carrier phase measurements are more accurate, code pseudoranges are also used in PPP solutions as they are unambiguous and do not suffer from cycle slips. Hence, the performance of the solution still relies on the code pseudoranges. Code pseudoranges have bigger multipath errors than carrier phase measurements [Jong-Hoon Won, 2017]. It is then important to determine when a code pseudorange is affected by multipath in order to either exclude the measurement or mitigate its effect. This thesis had as an objective to improve the positioning accuracy by detecting multipath affected measurements.

Since the GNSS receiver of 3D Aerospace employs a JETSON nano -a small computer on a printed circuit board with a graphical power unit - machine learning (ML) is possible onboard. Therefore, the second part of this thesis focused on detecting multipath with the help of deep learning. GNSS multipath can be divided into two categories [Lau, 2021]:

- A line of sight (LOS) signal with one or multiple reflections of the signal reaching the antenna.
- Only non-line of sight (NLOS) signals arriving to the antenna.

Multipath reflections occur when the direct signal is reflected or diffracted by surrounding materials before reaching the antenna. These reflections are contained in the original signal to produce a distorted version of the signal. After being processed by the receiver, this distorted signal degrades the accuracy of the GNSS observables (code pseudoranges, carrier phase, and Doppler measurements). These observables are used to compute a position solution ultimately resulting in a degraded solution.

NLOS signals happen when the direct signal is masked, yet one or more reflections reach the antenna. If the power of the NLOS signal is large enough, the GNSS tracking loops can

process it without detecting its nature. This type of multipath also induces a bias in the position solution.

In this thesis, detection of NLOS multipath is not the primary focus as the methods proposed herein do not allow for precise NLOS detection. Yet, their effects will be mitigated by the robust estimation technique presented. Detection of NLOS measurements can be performed by other methods such as the ones presented in [Sanromà Sánchez et al., 2016], [Suzuki and Amano, 2021], or [Matera et al., 2019].

In the existing literature, most algorithms focus on the detection of one category of multipath at a time with a focus on NLOS detection with a wide array of techniques. [Hsu, 2017] tried to detect NLOS satellites with a support vector machine SVM. [Jiang C., 2021] aimed at modeling the code discriminator with Gaussian fitting to detect NLOS. [Sanromà Sánchez et al., 2016] and [Marais et al., 2014] used fisheye images and the carrier to noise density ratio. This is due to the fact that NLOS signals and multipath reflections often affect the positioning solution very differently [Groves et al., 2013]. Some account for LOS reflections and NLOS in the same algorithm such as [Matera et al., 2019] who isolates the multipath from reference base stations data fused with fisheye images, or [Suzuki et al., 2020] who uses a rotating antenna to mitigate both categories of multipath errors.

Lately, machine learning has been applied to GNSS data processing especially for multipath detection/mitigation with promising results. For example, [Munin et al., 2020] and [Blais et al., 2022] fed synthetic correlator outputs to convolutional neural network (CNN) to detect multipath. Since they generated controlled multipath and knew when multipath affected the correlators, their method lacked a classification process that would make the method applicable to real signals. Furthermore, the images containing the correlator outputs fed to the CNN were generated from every integration epoch of the tracking process. However, in real-life scenarios, due to the tracking filters and multipath, correlator outputs are correlated in time. [Suzuki and Amano, 2021] also used correlator outputs but to detect NLOS via neural networks and SVMs by classifying NLOS measurements using a fisheye camera based algorithm. They, too, used correlator outputs at every integration epoch leading to correlation between the machine learning inputs. [Quan et al., 2018] also used convolutional neural networks for multipath detection/exclusion but uses the signal to noise ratio and pseudoranges as inputs to the network. [Xu et al., 2020] and [Hsu, 2017] used SVMs to detect NLOS using shadow matching and GNSS measurements (pseudoranges and Doppler shifts) respectively.

Multipath rays depend on the environment; they cannot be predicted unless 3D maps are built and/or ray-tracing is used [Bétaille et al., 2013][Peyret et al., 2014]. Current computational power units (CPU) are not able to process such amounts of data in real-time. To avoid extra processing power on the CPU , no external information regarding the environment was used. The definition of multipath depends on the targeted positioning accuracy.

As using CNNs can be very efficient in finding features that are hard to model [Liu, 2018], they were used in this work as a binary classifier to detect multipath. The CNN was used to detect multipath, but this alone does not improve the positioning accuracy. The CNN classification needs to be fed to a positioning algorithm to take into account this information.

Due to the presence of multipath, classical positioning algorithms such as the weighted least squares estimation (WLSE) tend to perform poorly. Indeed, the WLSE algorithm is optimal for measurements that follow a centered Gaussian distribution. Yet, when multipath is present in the GNSS observables, especially the pseudoranges, their distribution cannot be assumed as centered and/or Gaussian. In this case, robust statistics can be very efficient as shown in [Medina et al., 2019b] as they are able to deal with multiple erroneous observations. The downside of these robust estimators is their loss of efficiency when the data follows a centered Gaussian distribution making them sub-optimal when the pseudoranges are unaffected by multipath. There is a trade-off between using the regular WLSE algorithm with a risk of large positioning errors in heavy multipath conditions and using robust estimators at the expense of slightly degraded position in ideal conditions. Thus, by knowing when the observables are affected by multipath, the positioning solution could switch between the WLSE algorithm and robust estimator according to the multipath conditions.

This kind of algorithm was investigated in [Ding et al., 2023] by using pseudorange residuals to decide which environment the receiver is in. The solution proposed in this thesis is based on the CNN multipath decision to decide on the multipath conditions. It will also investigate adapting the robust estimator parameters according to the estimated multipath environment.

## 1.2 Thesis objectives

The focal points of this research were to design a PPP algorithm for a real-time GNSS receiver and to mitigate the effect of multipath from the positioning solution to improve the positioning accuracy. To do so, the thesis was divided into three phases.

1. Develop a PPP algorithm with a limited computation power requirement
    - One goal of this work is to present an alternative to carrier phase ambiguity estimation in PPP algorithms. Indeed, as these algorithms have become more complex in an effort to maximize positioning accuracy, these approaches present a challenge for systems that require real-time capabilities with limited computational power or energy consumption. The presented TDCP based EKF aims at matching the accuracy of a conventional PPP algorithm, which estimates the carrier phase ambiguities.
    - Another goal of the presented algorithm is to be much faster than the carrier phase ambiguity estimation one so that it can be implemented on a real-time receiver.
2. Detect multipath from correlator outputs using GNSS data only
    - The goal is to detect line of sight reflections and then exclude them on real signals (Galileo E1-B and GPS L1 C/A) using GNSS data only. This is accomplished thanks to CNNs fed by multiple correlator outputs. The labeling of data (multipath present or not) is obtained by computing the tracking bias of the discriminator function.

- The number of correlators used to detect multipath will also be varied to reduce the computational power required to have multiple correlators. The CNN performances will be investigated accordingly.
- The benefit of using CNNs over the raw tracking bias will be demonstrated both in terms of metric performance and computational speed.

3. Mitigate multipath in positioning solutions

   - The goal was to improve the accuracy of WLSE or robust estimators based solutions by proposing an algorithm that fuses both solutions depending on the multipath conditions.
   - Another goal was to adapt the parameters of the robust estimators based on the multipath conditions to further improve the quality of the robust estimator solution.

## 1.3 Thesis contributions

The three objectives of the thesis presented have been reached. The main contributions are the following:

1. Proposal and design of a dual frequency, dual constellation PPP engine using an EKF and TDCP measurements that performs better than an EKF estimating the carrier phase ambiguity without cycle slip repair and is much faster.

2. Proposal and implementation of an algorithm to classify multipath based on the magnitude of the tracking bias using the pseudorange multipath error envelope (MEE) of frequently encountered materials in urban environments.

3. Presentation of a deep learning based method to detect multipath on the fly by using multiple correlator outputs coming from a modified delay lock loop (DLL).

4. Validation of proposed algorithms on real data collects

5. Fusion of the WLSE algorithm and robust estimators in an algorithm that estimates the multipath environment to decide which solution to use.

6. Performance assessment on the adaptation of parameters of robust estimators based on the estimated multipath environment of the receiver.

On top of this thesis, these contributions can be found in the articles listed below:

1. Guillard, Anthony, Thevenon, Paul, Milner, Carl, "Using TDCP Measurements in a Low-Cost PPP-IMU Hybridized Filter for Real-Time Applications," Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, Colorado, September 2022, pp. 2090-2103. https://doi.org/10.33012/2022.18331

2. Guillard A, Thevenon P and Milner C (2023) Using convolutional neural networks to detect GNSS multipath. Front. Robot. AI 10:1106439. doi: 10.3389/frobt.2023.1106439

3. Guillard, Anthony, Thevenon, Paul, Milner, Carl, Macabiau, Christophe "Benefits of CNN-Based Multipath Detection for Robust GNSS Positioning," Proceedings of the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023), Denver, Colorado, September 2023.

## 1.4 Thesis outline

The manuscript is structured as follows.

**Chapter 2** describes the necessary algorithms to implement in order to successfully process an incoming GNSS signal. It starts with the reception of the signal at the antenna which digitalizes the signal so that it can be acquired and tracked. The generation of GNSS observables is also explained as well as the impact that multipath has on both tracking and measurements.

**Chapter 3** presents the different positioning algorithms that can be implemented from the obtained GNSS observables. The traditional least squares algorithm and Kalman filter are presented first. The Kalman filter will also be presented when an inertial measurement unit is fused to the GNSS measurements for added robustness. Then, the three most common robust estimators are described - M, S, and MM.

**Chapter 4** deals with the theory behind convolutional neural networks. This chapter highlights the difference with traditional neural networks and also presents the elements making up a CNN. It also gives insight on how to reduce the risk of overfitting.

**Chapter 5** presents the PPP algorithm implemented with TDCP measurements. It describes how to generate the TDCP measurements and the necessary algorithms to implement this solution. The test set-up and conditions under which this filter was evaluated on are also mentioned. Lastly, the results are discussed from both an accuracy and computational performance point of view.

**Chapter 6** details the implementation of the CNN model to detect multipath and the data labeling used to train the CNN. It also deals with the signal deformation caused by hardware. The test set-up and data used for the results are also presented before the results which detail the metrics and computational performance.

**Chapter 7** defines the algorithm used to mitigate the effect of multipath in the positioning solution based on the CNN algorithm of chapter 6. It also details the different parameters of robust estimators that can be changed. Then, the data and test set-up are expressed before presenting the results and their conclusions.

**Chapter 8** reflects on the work presented and their conclusions as well as depicting research leads that can be pursued in the future.

CHAPTER 2

# GNSS Receiver Processing and Observables

GNSS signals are transmitted from satellites in the form of electromagnetic waves. Upon reaching a GNSS receiver, several steps must be performed before being able to retrieve the underlying information in the signal. One such receiver solution, is the software defined radio (SDR) which is divided into hardware and software blocks. The hardware is made of the antenna and the front-end. On the other hand, the software blocks are the algorithms required to demodulate the signal bits and compute the GNSS measurements. Indeed, the signal contains the Keplerian parameters to compute the positions of the satellites in the form of data bits. Furthermore, by tracking the incoming signal, measurements of the pseudo-distance to the satellite, phase cycles, and the Doppler offset can be obtained. During the course of this PhD, the development of the software blocks was also done. This chapter will detail the traditional algorithms to be implemented to retrieve these GNSS observables. It will also describe alternative acquisition and tracking algorithms used in modern day receivers. This chapter will also look into the effect that multipath has on the tracking process and the generated observables.

## Chapter Contents


2.1 Signal Structure . . . . . 8
2.1.1 Data Bits . . . . . 8
2.1.2 Pseudo-Random Noise Code . . . . . 8
2.1.3 Carrier Wave . . . . . 9
2.1.4 IF Signal Model . . . . . 10
2.2 Signal Conditioning and Digitization by the RF front-end . . . . . 10
2.2.1 Filtering and Amplification . . . . . 11
2.2.2 Down Conversion . . . . . 12
2.2.3 Automatic Gain Control and Analog to Digital Converter . . . . . 12
2.3 Acquisition . . . . . 13
2.3.1 Generating in-phase and quadrature phase correlator outputs . . . . . 13
2.3.2 Decision . . . . . 15
2.3.3 Alternate Methods . . . . . 16
2.4 Tracking . . . . . 17
2.4.1 Carrier frequency offset (Doppler shift) and phase . . . . . 18
2.4.2 Code Delay . . . . . 21
2.4.3 Effect of Multipath . . . . . 24

2.4.4 Alternate Methods . . . . . 26
2.5 Observables Generation . . . . . 27
2.5.1 Code Pseudorange . . . . . 28
2.5.2 Carrier Phase . . . . . 30
2.5.3 Doppler . . . . . 31
2.5.4 Effect of Multipath . . . . . 31
2.6 Conclusions . . . . . 34

## 2.1 Signal Structure

A GNSS signal is composed of a carrier wave whose purpose is to transmit the information as an electromagnetic wave. This carrier wave has a ultra high frequency (in the L-band) and a large bandwidth of several Mega Hertz. A GNSS signal is also composed of a Pseudo Random Noise (PRN) code which is a series of $X$ bits – also called chips – where $X$ depends on the signal. Navigation bits also make up the signal.

These elements are presented in the sections below and illustrated in Figure 2.1 but are not to scale.

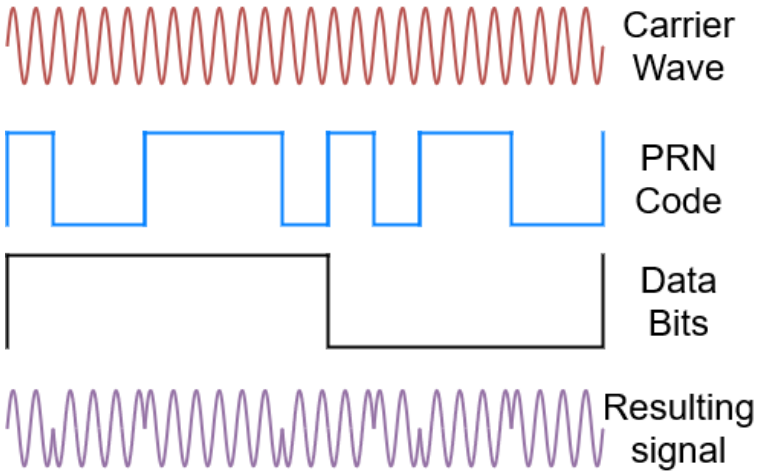


Figure 2.1: GNSS Signal Elements (not to scale).

### 2.1.1 Data Bits

Data bits in GPS and Galileo signals are sent with a very low data rate. They contain the Keplerian parameters, also called ephemeris, which are the information required to compute the position of the satellites. They also contain the values of the satellite clock correction parameters with respect to their true constellation time. Other parameters are transmitted as the almanac, the satellite health status, etc..

### 2.1.2 Pseudo-Random Noise Code

A PRN code is a periodic sequence of bits that is pseudo-random. Galileo and GPS systems are based on code division multiple access, which allows several transmitters to send a

message on the same frequency band but with different codes, in this case the PRN. This means that every GPS and Galileo satellite transmit a unique PRN code. The PRN codes generators are the linear feedback shift registers. This is due to the fact that they are very efficient to generate the codes and provide a very high performance with respect to correlation purposes. This can be broken down into two properties:

1. **Cross Correlation (correlation on two different PRNs)**: if the cross correlation function of a PRN code is computed, the result will be close to 0 for all delays.

$$\forall k \in \mathbb{Z} \text{ s.t } (c_x * c_y)(k) \approx 0 \tag{2.1}$$

   As the discrete form of the correlation operation is presented, equation 2.1 can be expressed as:

$$R_{c_x c_y}(k) = \frac{1}{N} \sum_{n=0}^{N_c-1} c_x(n) c_y(n - kT_c) \approx 0 \tag{2.2}$$

2. **Autocorrelation (PRN correlated with itself)**: if the autocorrelation function of a PRN code is computed, the maximum value will be for a delay of 0 (or equal to the period of a PRN), but for a delay different than 0 then the autocorrelation value will be close to 0.

$$\begin{aligned} R_{c_x c_x}(k_0) &= 1 \\ \forall k \in \mathbb{Z}^* \text{ s.t } R_{c_x c_x}(k) &\approx 0 \end{aligned} \tag{2.3}$$

Where:

- $R_{c_i c_j}$ is the correlation result of PRNs from satellite $i$ and $j$
- $N_c$ is the number of chips in a PRN period
- $c_i$ denotes the PRN code for satellite $i$
- $T_c$ is the duration of a PRN chip in seconds

The PRN code has a period of $T_P$ which depends on the type of signal. When the signal reaches the antenna, the start of the PRN code is offset by an unknown value called $\tau$ in (2.5). To demodulate the data bits and to estimate the code pseudoranges, the PRN code must be removed in the signal processing blocks. Hence, the code delay must be estimated. During acquisition, one goal is to roughly estimate this code delay, and during the tracking process, this code delay estimation is refined.

### 2.1.3 Carrier Wave

The carrier wave is a waveform that is modulated with an input signal for the purpose of conveying information. In GNSS, this carrier wave is of the form:

$$\phi(k) = \cos\Big(2\pi(f_{IF} + f_D(k))kT_s + \theta(k)\Big) + j \sin\Big(2\pi(f_{IF} + f_D(k))kT_s + \theta(k)\Big) \tag{2.4}$$

When the satellite and receiver are moving, it induces a frequency shift of the signal. This is called the Doppler effect and the offset is denoted $f_D$ in (2.4). To access the data bits

contained in the signal and estimate the Doppler shift and carrier phase measurements, the carrier wave must be removed from the signal. To do so, the Doppler frequency shift must be estimated. One of the purposes of the acquisition process is to give a first estimate of the Doppler shift. Once the signal is acquired, the tracking process will finely estimate the Doppler shift.

### 2.1.4 IF Signal Model

After going through the RF front-end, a GNSS signal can be modeled as follows.

$$s(k) = \frac{A}{2}\exp\Big(2j\pi(f_{IF} + f_D(k))kT_s + \theta(k)\Big)c\Big(kT_s - \tau(k)\Big)d\Big(kT_s - \tau(k)\Big) + n(kT_s) \quad (2.5)$$

Where:

- $k$ is the discrete index at which the signal was sampled
- $A$ is the complex amplitude of the signal
- $c(t)$ represents the PRN code
- $d(t)$ represents the data bits
- $\tau$ is the receiver PRN code delay in seconds
- $n(t)$ is the incoming noise assumed as Additive White Gaussian Noise (AWGN)
- $f_D(t)$ is the induced Doppler frequency of the GNSS signal in Hz
- $\theta(t)$ is the phase of the signal in radians

## 2.2 Signal Conditioning and Digitization by the RF front-end

The front-end is the hardware block that transforms an analog signal to digital values so that the characteristics of the signal can be obtained by the signal processing algorithms. Figure 2.2 illustrates a typical GNSS radio frequency (RF) front-end.

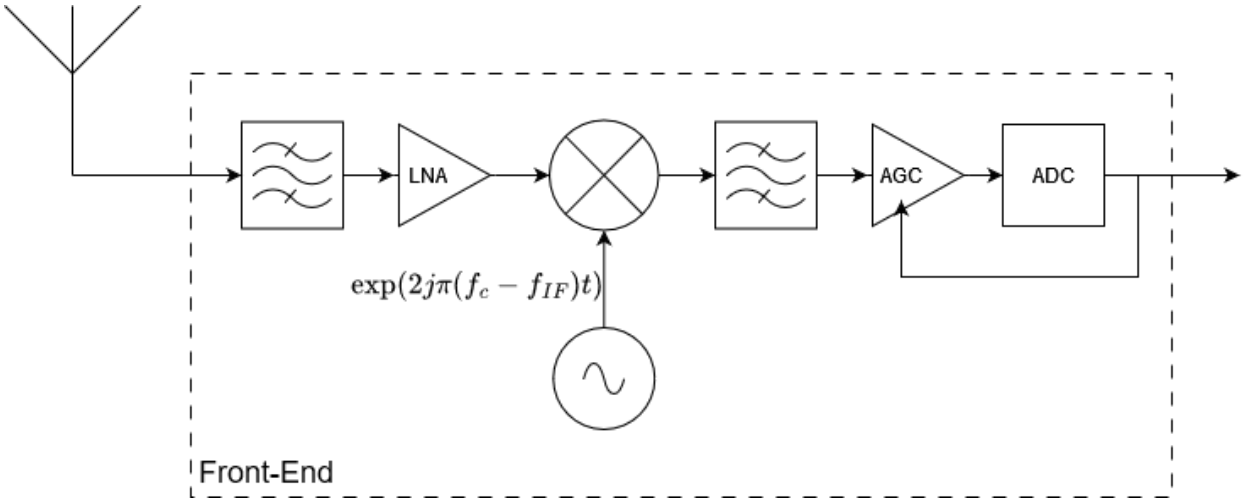


Figure 2.2: Illustration of a GNSS RF front-end.

### 2.2.1 Filtering and Amplification

The transmitted signal has a carrier wave with a frequency of 1575.42 MHz for GPS L1 C/A and a wideband spectrum that depends on the type of signal. A bandpass filter is applied around the carrier frequency to only select the desired frequency range. An example of an ideal bandpass filter applied around the main lobe of the GPS L1 C/A power spectral density is depicted in Figure 2.3.

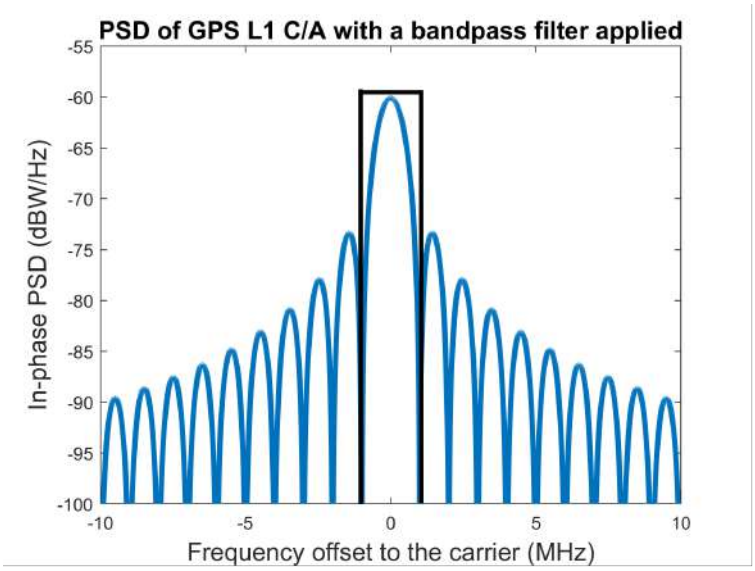


Figure 2.3: Illustration of an ideal bandpass filter.

When reaching the antenna, the GNSS signal has very low power and is mostly composed of noise. In order to increase the power of the signal, an amplifier is used. However, when raising the power of a signal, the amplifier also increases the noise of the signal. In most cases, a low noise amplifier is used in GNSS front-ends as they have good gains while having a low noise figure.

### 2.2.2 Down Conversion

The analog signal has an ultra high frequency. Digital processors that replicate such a carrier wave is financially costly. Therefore, the signal must be down converted before digitalizing it. To do so, a mixer and a local oscillator are used to bring the signal to an intermediate frequency while conserving its structure. The local oscillator will generate a sine wave with a chosen frequency $f_{LO}$. The signal is then multiplied with this sine wave, and the mixer will bring the signal down to the chosen intermediate frequency. This process is illustrated mathematically below. For now, the GNSS signal model is simplified and its detailed definition will be given in section 2.1. The GNSS signal has an in-phase component (I - $\cos(\omega t)$) and an quadrature phase component (Q - $\sin(\omega t)$).

$$s(t) = s_{GNSS}(t)\Bigg( \cos\Big(2j\pi t(f_c + f_D(t) + \theta(t)\Big) - (\sin\Big(2j\pi t(f_c + f_D(t) + \theta(t)\Big)\Bigg) \tag{2.6}$$

Supposing we have only one signal (no Q component), the signal can be brought down to:

$$s(t)\cos(2j\pi f_{LO}t) = \frac{1}{2}s_{GNSS}(t)\Bigg(\cos\Big(2j\pi t(f_c+f_D(t)+f_{LO})+\theta(t)\Big)+\cos\Big(2j\pi t(f_{IF}+f_D(t))+\theta(t)\Big)\Bigg) \tag{2.7}$$

Where:

- $s(t)$ is the analog GNSS signal at time $t$
- $s_{GNSS}(t)$ is the analog GNSS signal without the carrier wave at time $t$
- $f_c$ is the carrier frequency of the GNSS signal in Hz
- $f_D(t)$ is the induced Doppler frequency of the GNSS signal at time $t$ in Hz
- $f_{LO}$ is the frequency of the local oscillator in Hz
- $f_{IF}$ is the intermediate frequency in Hz which is equal to $f_{IF} = f_c - f_{LO}$
- $\theta(t)$ is the phase of the signal at time $t$ in radians

By then using another bandpass filter around the intermediate frequency, it is possible to remove the high frequency component. Then, only the low frequency component is left, leaving the signal to be passed through the analog to digital converter (ADC).

### 2.2.3 Automatic Gain Control and Analog to Digital Converter

The signal is then passed through an automatic gain control which essentially amplifies the signal while controlling its increase in amplitude automatically [Hui, 2020]. This adapts the signal amplitude to benefit from the full quantification range of the ADC, independently from the receiving conditions (antenna gain, cable length, noise or interference levels).

The amplified signal is then passed through an ADC, which is responsible for transforming the analog signal to digital samples. Essentially, it takes snapshots of the analog signal

at every sampling period denoted by $T_s$. The sampling frequency chosen for the ADC should satisfy the Nyquist criterion, meaning that:

$$f_s > 2B_w \tag{2.8}$$

Where:

- $f_s = 1/T_s$ is the sampling frequency in Hz
- $B_w$ is the bandwidth of the bandpass filter defined in 2.2.1

An important parameter of the ADC is the number of bits used to quantize the signal. Every analog value of the signal must be mapped to a chosen but fixed number of bits. Increasing the number of quantization bits reduces the power loss due to the quantization operation, but increases the power consumed and the number of operations in the digital signal processing. Once, the signal is digitized through the ADC, it can be fed to the signal processing blocks of the receiver.

## 2.3 Acquisition

Once the signal has passed through the front-end, it is now a digital signal and can be processed by the digital signal processing blocks. The first step is to check which satellites are in view and roughly estimate the parameters of the corresponding signal. This is the role of the acquisition process.

### 2.3.1 Generating in-phase and quadrature phase correlator outputs

As mentioned earlier, to estimate the GNSS raw measurements, the carrier wave and the PRN code must be wiped off the signal in separate but linked processes. The PRN code is made of a sequence of -1s and 1s; so multiplying it with a synchronized replica would remove it. However the receiver code delay, defined in (2.5) is unknown. Hence, the correlation operation is used between the incoming signal and a locally generated code replica. Since every satellite of a given signal type has a different PRN code, the local code replica must correspond to the targeted satellite to acquire. To wipe-off the carrier, a local carrier replica of the carrier must also be generated. The local replica must be multiplied with the incoming signal and a low-pass filter applied to remove the high frequency component. However, the Doppler frequency shift is unknown; therefore, it must be estimated to remove the carrier wave. The correlation operation is also used for the carrier and code wipe-off. This is illustrated in Figure 2.4.

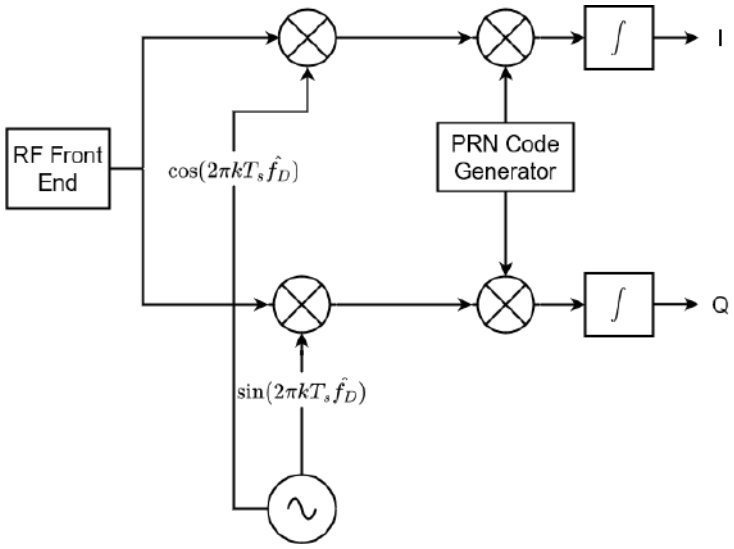


Figure 2.4: Generation of I and Q correlator outputs during the acquisition process.

During the coherent integration interval, three hypotheses are made [Julien, 2006]. The code delay, Doppler shift, and phase are assumed constant. The coherent integration is performed within the duration of one data bit, so its value is assumed constant ($d(kT_s - \tau(k)) = d(n)$). With these assumptions and filtering the high frequency term, Figure 2.4 equates to:

$$I(n) = \frac{A(n)}{2}\frac{d}{NT_s}\sum_{k=0}^{N} c(kT_s - \tau)c_L(kT_s - \hat{\tau})\cos\Big(2\pi\varepsilon_{f_D}kT_s + \theta\Big) + n_I(k) \tag{2.9}$$

$$Q(n) = \frac{A(n)}{2}\frac{d}{NT_s}\sum_{k=0}^{N} c(kT_s - \tau)c_L(kT_s - \hat{\tau})\sin\Big(2\pi\varepsilon_{f_D}kT_s + \theta\Big) + n_Q(k) \tag{2.10}$$

Where:

- $c_L$ is the locally generated code replica
- $\hat{\tau}$ is the estimated code delay in seconds
- $\varepsilon_{f_D}$ is the Doppler shift error in Hz
- $N$ is the number of samples during the integration interval

Equations (2.9) and (2.10) can be simplified to:

$$I(n) = \frac{A(n)}{2}dR_c(\varepsilon_\tau)\cos\Big(\pi\varepsilon_{f_D}NT_s + \theta\Big)sinc\Big(\pi\varepsilon_{f_D}NT_s\Big) + n_I(n) \tag{2.11}$$

$$Q(n) = \frac{A(n)}{2}dR_c(\varepsilon_\tau)\sin\Big(\pi\varepsilon_{f_D}NT_s + \theta\Big)sinc\Big(\pi\varepsilon_{f_D}NT_s\Big) + n_Q(n) \tag{2.12}$$

Where:

- $\varepsilon_\tau$ is the code delay error in seconds
- $n_I$ and $n_Q$ are the noise of the correlators

### 2.3.2 Decision

As there are no previous information regarding the code delay or Doppler shift, all potential values are evaluated. This means that there are $N_\tau \times N_D$ I and Q correlator outputs generated per satellite. The resolution of the code delay is set to half a chip or less with the range going from zero to the number of PRN chips. On the other hand, the Doppler resolution depends on the integration time $NT_s$. For a maximum 3.5 dB loss, the resolution is of $1/2(NT_s)$ [Foucras et al., 2016]. The range of values is given by the maximum and minimum Doppler offset that can be induced by a satellite as it is the predominant Doppler term. It does not exceed $[-5\text{kHz}, 5\text{kHz}]$ [Borre et al., 2007]. This yields an acquisition grid where a pair of Doppler shift and code delay are used to compute the I and Q correlator outputs.

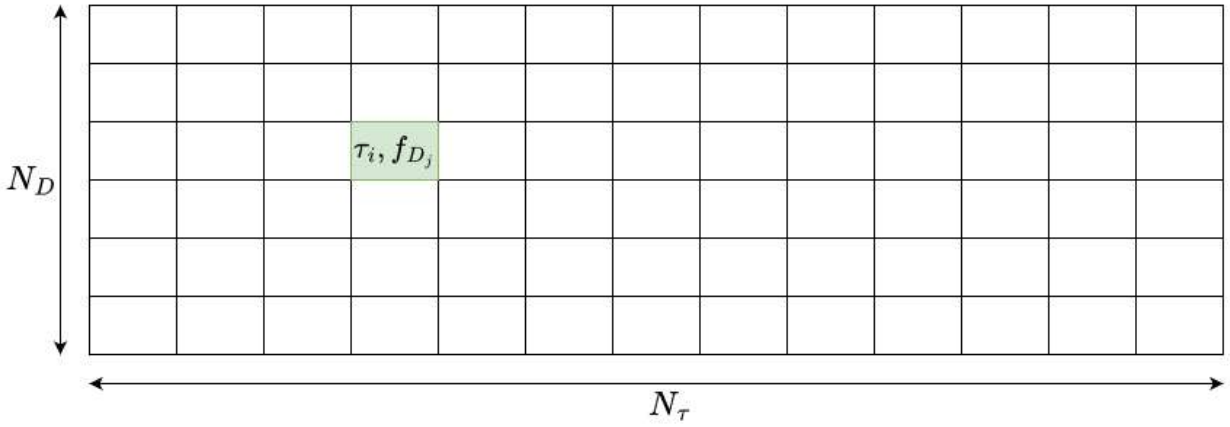


Figure 2.5: Acquisition Grid.

To decide whether a satellite is present, a decision test must be performed. The test statistic is given by:

$$T = I^2(n) + Q^2(n) \tag{2.13}$$

$$T = \frac{A(n)^2}{2} R_c(\varepsilon_\tau) sinc^2\Big(\pi \varepsilon_{f_D} NT_s\Big) + n_T(n) \tag{2.14}$$

Where:

- $T$ is the test statistic
- $n_T(n)$ is the noise of the statistic which is equal to:

$$n_T(n) = n_I(n)^2 + n_Q(n)^2 + 2n_I(n)\frac{A(n)}{2} dR_c(\varepsilon_\tau) \cos\Big(\pi \varepsilon_{f_D} NT_s + \theta\Big) sinc\Big(\pi \varepsilon_{f_D} NT_s\Big) + 2n_Q(n)\frac{A(n)}{2} dR_c(\varepsilon_\tau) \sin\Big(\pi \varepsilon_{f_D} NT_s + \theta\Big) sinc\Big(\pi \varepsilon_{f_D} NT_s\Big)$$

A Neyman Pearson test is frequently used [Neyman and Pearson, 1933] where two hypotheses are tested:

$$\begin{cases} H_0, \text{ Correlation Peak Absent } : T < \gamma \\ H_1, \text{ Correlation Peak Present } : T \geq \gamma \end{cases} \tag{2.15}$$

Where:

- $\gamma$ is the threshold value

In a NP test, there are four distinguishable cases shown in Table 2.1.

| Expected / Truth | $H_0$ | $H_1$ |
|---|---|---|
| $H_0$ | $P_N$ | $P_{Fa}$ |
| $H_1$ | $P_{Md}$ | $P_D$ |

Table 2.1: Neyman-Pearson Test Probabilities

Where:

- $P_N$ is the probability of not acquiring a satellite when it is absent
- $P_{Fa}$ is the probability of false alarm meaning the probability of acquiring a satellite when it is absent
- $P_{Md}$ is the probability of missed detection meaning not acquiring a satellite when it is present
- $P_D$ is the probability of acquiring a satellite when it is present

The threshold value is determined by the desired theoretical probability of false alarm. As the noise terms of the I and Q components are assumed uncorrelated [Van Dierendonck et al., 1992], the test statistic is the sum of squared independent Gaussian variables. Therefore, $T$ follows a chi-square distribution with 2 degrees of freedom.

To increase the detection performance, two methods are widely used:

1. **Increasing the coherent integration time** when computing the I and Q correlators. However, increasing the coherent integration can also be detrimental to the test statistic. Indeed, one assumption made for (2.9) and (2.10) was that the bit value was constant during the integration. Increasing the coherent integration increases the chance of invalidating this hypothesis.
2. **Using non-coherent summations.** This means that the test statistic becomes:

$$T = \sum_{n=0}^{M} I^2(n) + Q^2(n) \tag{2.16}$$

Where:

- $M$ is the number of non-coherent summations

$T$ now has $2M$ degrees of freedom. The downside of using non-coherent summations is that the noise cross-terms are increased due to the squaring of the I and Q correlators.

### 2.3.3 Alternate Methods

The algorithms presented in 2.3 are the traditional ones; however, more efficient algorithms exist. Their principle is presented in the following section.

#### 2.3.3.1 Parallel Searching

Instead of testing a pair of values – code delay and Doppler offset – one by one, this method either computes all possible Doppler values for one code delay or all code delay values for one Doppler offset. These two methods are illustrated in Figures 2.6 and 2.7. These methods use the Fast Fourier Transform to compute all code delay or all frequency bins at once, as described in [Leclère et al., 2014].

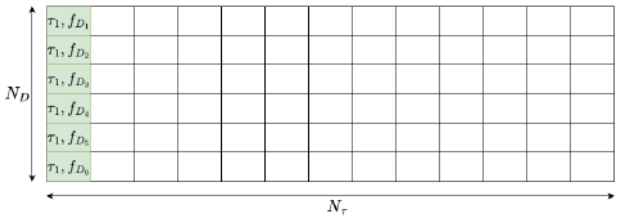


Figure 2.6: Parallel Doppler Search Method.

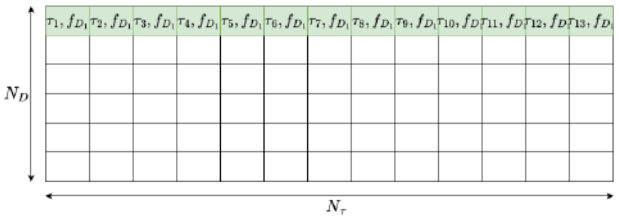


Figure 2.7: Parallel Code Search Method.

#### 2.3.3.2 Double Block Zero Padding

Double block zero padding has a reduced number of operations and can acquire weaker GNSS signals than the conventional algorithm [Prasad, 2019].This algorithm relies on splitting the incoming signal into $N$ blocks. The PRN code is also split into $N$ blocks with zero padding on each one. Circular correlation is performed between the incoming signal and PRN blocks where the FFT is then applied on each resulting correlation block. Finally, the PRN blocks are permuted and the circular correlation is repeated for all permutations. The test statistic is unchanged. This method is further detailed in [Van Neem D.J.R, 1991].

#### 2.3.3.3 Pilot Acquisition

Some signals, such as Galileo E1, are made up of a data and pilot signal. The pilot signal does not contain any bit so is immune to data bit transitions. Hence, the coherent integration time can be increased for a more reliable acquisition decision. The acquisition using the pilot signal can be done by computing the I and Q correlators of both data and pilot signals independently and fuse the results into one test statistic as presented in [Borio and Lo Presti, 2008]:

$$T = \sum_{n=0}^{M} I_D^2(n) + Q_D^2(n) + I_P^2(n) + Q_P^2(n) \tag{2.17}$$

Where:

- subscript $D$ stands for the data component
- subscript $P$ stands for the pilot component

## 2.4 Tracking

Once rough estimates of the Doppler and code delay are obtained, they need to be refined to fully wipe off the PRN code and the carrier wave and estimate the GNSS measurements. In most receivers, this is achieved thanks to the delay lock loop and phase lock loop (PLL)

which estimate the code delay, phase, and Doppler shift of the incoming signal. This section will describe their functioning as well as how noise and multipath affect these loops.

### 2.4.1 Carrier frequency offset (Doppler shift) and phase

From acquisition, two errors (remaining Doppler shift and phase) remain from the carrier wipe-off that make the bit demodulation not possible yet. The goal of carrier tracking is to minimize these errors and update the estimates over time. Indeed, due to the relative motion between satellite and receiver, the Doppler shift of the signal is not constant in time. The PLL as its name suggests estimates the phase of the signal and also the Doppler shift.

To do so, signal samples are first accumulated and multiplied with a locally generated carrier replica with the Doppler estimate from the acquisition process for the first iteration and then from the PLL for the other iterations. This generates I and Q correlators. A discriminator is then computed with a combination of I and Q measurements. Afterwards, the discriminator is filtered by the PLL filter. A numerically controlled oscillator (NCO) uses the filtered discriminator to generate new carrier replicas with updated parameters. The process is then repeated as is depicted in Figure 2.8.

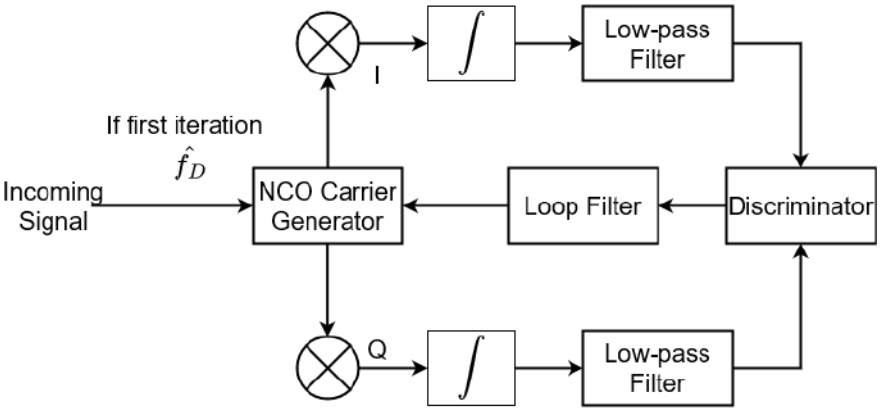


Figure 2.8: Phase Lock Loop.

#### 2.4.1.1 Generation of I and Q components

Similarly to the acquisition process, I and Q correlators are computed by the integration of the multiplication between the incoming signal and a locally generated replica. During tracking, only one set of values is used for the generation I and Q correlators, which are the ones estimated from the tracking loops. The I and Q correlators outputs are equal to:

$$I(n) = \frac{A(n)}{2} dR_c(\varepsilon_\tau) \cos\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) sinc\Big(\pi \varepsilon_{f_D} N T_s\Big) + n_I(n) \tag{2.18}$$

$$Q(n) = \frac{A(n)}{2} dR_c(\varepsilon_\tau) \sin\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) sinc\Big(\pi \varepsilon_{f_D} N T_s\Big) + n_Q(n) \tag{2.19}$$

Where:

- $\varepsilon_\theta$ is the phase error in radians

The goal of the PLL is to drive the Doppler and phase errors to zero meaning $\varepsilon_{f_D} \rightarrow 0$ and $\varepsilon_\theta \rightarrow 0$. Assuming this, equations (2.18) and (2.19) become:

$$I(n) = \frac{A(n)}{2} dR_c(\varepsilon_\tau) + n_I(n) \tag{2.20}$$

$$Q(n) = n_Q(n) \tag{2.21}$$

It can be observed, in this case, that the I arm only contains the code delay error. If the code delay error is correctly estimated, then the data bits can be retrieved from the original signal from the I arm. On the other hand, the Q arm is only composed of noise.

#### 2.4.1.2 Discriminator

The goal of the PLL discriminator is to combine the I and Q correlators to isolate and thus estimate the phase and Doppler errors. The error will then be used to generate a new carrier replica with updated phase and Doppler estimates.

An example of a discriminator, the arctangent discriminator, is presented in (2.22):

$$D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = \tan^{-1}\left(\frac{Q}{I}\right) \tag{2.22}$$

By replacing (2.18) and (2.20) in (2.22) and assuming no noise, this yields:

$$D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = \tan^{-1}\left(\frac{\cancel{\frac{A(n)}{2} dR_c(\varepsilon_\tau)} \sin\left(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\right) \cancel{sinc\left(\pi \varepsilon_{f_D} N T_s\right)}}{\cancel{\frac{A(n)}{2} dR_c(\varepsilon_\tau)} \cos\left(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\right) \cancel{sinc\left(\pi \varepsilon_{f_D} N T_s\right)}}\right) \tag{2.23}$$

$$D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = \tan^{-1}\Big(\tan(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta)\Big) \tag{2.24}$$

$$D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = \pi \varepsilon_{f_D} N T_s + \varepsilon_\theta \tag{2.25}$$

Every discriminator has a region of quasi-linearity where the discriminator output is assumed linearly proportional to the phase error. The arctangent discriminator is widely used due to its large linear region of $[(2k-1)\pi/2; (2k+1)\pi/2]\ \forall k \in \mathbb{Z}$ [Julien, 2005]. The discriminator value is used to generate a new carrier replica. The value of $(\varepsilon_{f_D}; \varepsilon_\theta)$ for $D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = 0$ yields the extra phase error caused by the errors affecting the PLL. To find this phase error, the zero crossing of the discriminator must be found. When no errors affect the signal, this is straightforward, however, when adding noise to the ATAN discriminator becomes:

$$D_{ATAN}(\varepsilon_{f_D}; \varepsilon_\theta) = \tan^{-1}\left(\frac{\Big(\tan(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta)\Big) + n_Q}{1 + \frac{n_I}{\frac{A(n)}{2} dR_c(\varepsilon_\tau) \cos\left(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\right) sinc\left(\pi \varepsilon_{f_D} N T_s\right)}}\right) \tag{2.26}$$

Increasing the noise also shortens the linear region of the discriminator. Since the slope of the discriminator must be of the same sign as its linear region for it to be a stable lock point [Julien, 2005], this means that the number of potential stable lock points decreases with the carrier to noise ratio. Figure 2.9 depicts the ATAN discriminator for different carrier to noise density ratios.

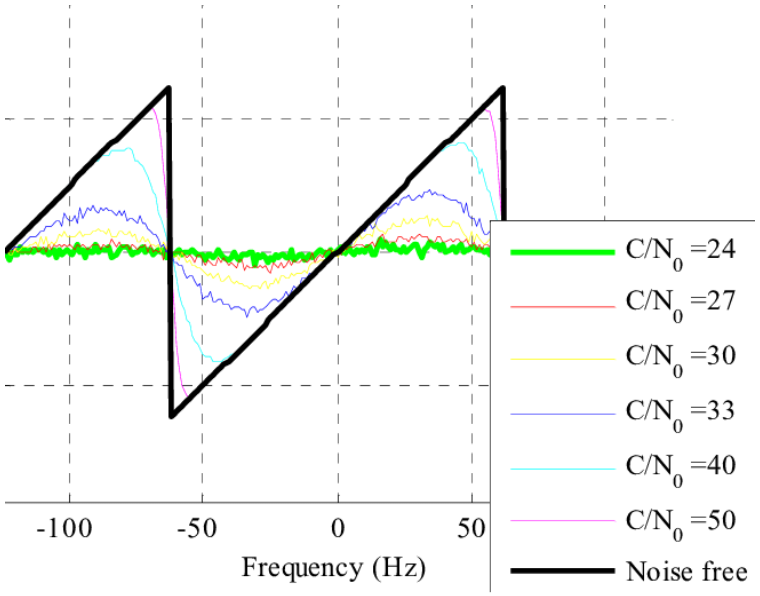


Figure 2.9: Arctangent discriminator linearity region for several CN0s [Foucras et al., 2014].

#### 2.4.1.3 Loop Filter

As highlighted by (2.26), the discriminator output is noisy. To attenuate the effect of noise, a filter is used on the discriminator. The design of these filters greatly affects the tracking performance of the PLL. The three parameters to choose in a loop filter are:

1. **Filter order** A low filter order is better for accuracy but will most likely lose lock under high dynamics of the signal.

2. **Integration time** A long integration time will increase the accuracy of the tracking loop if and only if there are no data sign transitions during the integration interval and if the Doppler shift and phase of the incoming signal have not changed.

3. **Loop bandwidth** The loop bandwidth is how much noise is filtered out by the filter. A low equivalent loop filter bandwidth will filter out most of the noise but will respond poorly to high signal dynamics.

The design of a PLL filter is based on trade-offs. The chosen filter characteristics depend on the targeted application.

The noise, even after being filtered by the loop is still present in the discriminator output. In a PLL where only Additive White Gaussian Noise affects the signal, the propagated noise can be estimated as Zero Mean Gaussian Noise [Legrand et al., 2000]. For the arctangent discriminator, the variance of the noise can be expressed as:

$$\sigma^2_{Noise} = \frac{\lambda^2}{4\pi^2}\frac{B_{DLL}}{C/N_0}\left[1 + \frac{1}{2NT_sC/N_0}\right] \tag{2.27}$$

#### 2.4.1.4 NCO Carrier Generator

The NCO is a digital signal generator which adapts its frequency based on the input. In a PLL, the input to the NCO is the filtered discriminator. The NCO either increases or

decreases the frequency of the carrier wave generated based on the filtered discriminator. By changing the frequency of the carrier wave every tracking interval, the goal is to remove the effect of the carrier wave of the signal.

### 2.4.2 Code Delay

Similarly to the Doppler offset, the estimated code delay value must be refined to demodulate the data bits. To retrieve the delay error, the DLL uses at least 3 correlators (early, prompt, and late). The delay error estimate is used to refine the frequency of the locally generated code replica with the incoming signal. If the code replica is aligned with the incoming PRN sequence, then the early and late correlator values are equal to each other in an error-free scenario. The difference in chips between the early and late correlator position is called the Early-Late (EL) spacing. This parameter has a quintessential role for the DLL as a large EL spacing will lead to a better tracking of the dynamics while a smaller one will reduce the effect of multipath [Van Dierendonck et al., 1992].

The DLL architecture is similar to the one of the PLL. First, the signal samples are accumulated and multiplied with the early, late, and prompt code replicas with the code delay estimate computed in the acquisition process for the first iteration and then coming from the DLL. This generates three I and three Q correlators which are then passed through a chosen discriminator. Afterwards, the discriminator is filtered by the DLL filter. A numerical controlled oscillator uses the filtered discriminator to generate new code replicas with updated parameters. The process is then repeated and is depicted in Figure 2.10.

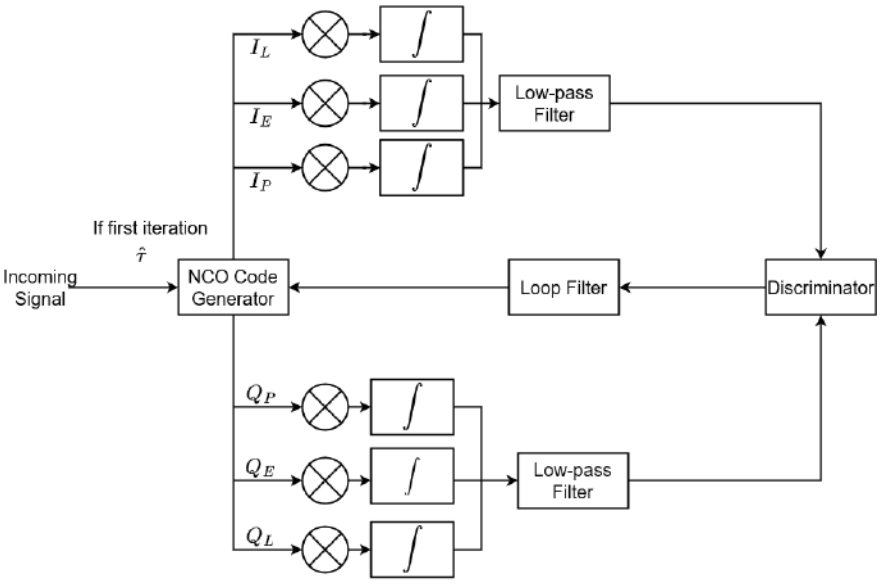


Figure 2.10: Delay Lock Loop.

#### 2.4.2.1 Generation of I and Q components

As mentioned earlier, three correlators per component are generated. The prompt correlators are obtained by multiplying a code replica with the incoming signal. The early correlators are computed by multiplying the signal with a code replica advanced by $E_L/2$ with respect

to the prompt correlator, hence the name early. On the other hand, late correlators are computed by multiplying the signal with a code replica delayed by $E_L/2$.

These correlators can be expressed as:

$$\begin{aligned} I_P(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau) sinc(\pi \varepsilon_{f_D} N T_s) \cos\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{I_P}(n) \\ I_E(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau + \frac{E_L}{2}) sinc(\pi \varepsilon_{f_D} N T_s) \cos\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{I_E}(n) \\ I_L(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau - \frac{E_L}{2}) sinc(\pi \varepsilon_{f_D} N T_s) \cos\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{I_L}(n) \end{aligned} \tag{2.28}$$

$$\begin{aligned} Q_P(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau) sinc(\pi \varepsilon_{f_D} N T_s) \sin\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{Q_P}(n) \\ Q_E(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau + \frac{E_L}{2}) sinc(\pi \varepsilon_{f_D} N T_s) \sin\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{Q_E}(n) \\ Q_L(n) &= \frac{A(n)}{2} d(n) R_c(\varepsilon_\tau - \frac{E_L}{2}) sinc(\pi \varepsilon_{f_D} N T_s) \sin\Big(\pi \varepsilon_{f_D} N T_s + \varepsilon_\theta\Big) + n_{Q_L}(n) \end{aligned} \tag{2.29}$$

Where:

- $E_L$ is the early-late spacing in chips

It can be seen that the prompt correlators (I and Q) are equal to the correlators from the PLL (2.18) and (2.19).

#### 2.4.2.2 Discriminator

There are two types of discriminators:

- Coherent Discriminators
- Non-Coherent Discriminators

Coherent Discriminators assume that the PLL is locked; hence, the phase error must be small for this discriminator to be used. On the other hand, non-coherent discriminators do not make any assumption on the state of the PLL. The discriminator functions are given for a coherent discriminator (early minus late) and for the early minus late power (EMLP), a non-coherent discriminator [Braasch, 2017]. In the absence of noise, the two discriminator functions are given in (2.30) and (2.32):

$$D_{EL}(\tau) = I_E - I_L \tag{2.30}$$

Since a coherent discriminator assumes that the PLL is locked this means that $\varepsilon_\theta \approx 0$ and $\varepsilon_{f_D} \approx 0$. Hence, (2.30) can be written as:

$$D_{EL}(\tau) = \frac{A(n)}{2}\Big(R_c(\tau + \frac{E_L}{2}) - R_c(\tau - \frac{E_L}{2})\Big) \tag{2.31}$$

On the other hand, the non-coherent discriminator function removes the effect of the phase, the early minus late power is shown:

$$D_{EMLP}(\tau) = \frac{I_E^2 + Q_E^2 - I_L^2 - Q_L^2}{2} \tag{2.32}$$

$$D_{EMLP}(\tau) = \frac{A(n)^2}{4} sinc(\pi \varepsilon_{f_D} N T_s)^2 \left( |R_c(\tau + \frac{E_L}{2})|^2 - |R_c(\tau - \frac{E_L}{2})|^2 \right) \tag{2.33}$$

In this thesis, only non-coherent discriminators will be studied, with a heavy focus on the EMLP. The goal of the discriminator is to isolate the code delay error to generate a new code replica with updated frequency. When normalized by early and late I and Q correlators, it can be shown that the code delay is equal to [Julien, 2006]:

$$\varepsilon_\tau = \frac{(2 - \alpha E_L N T_s) D_{EMLP}}{2\alpha (I_E^2 + Q_E^2 + I_L^2 + Q_L^2)} \tag{2.34}$$

Where:

- $\alpha$ is the absolute value of the slope of the autocorrelation function main peak ($\alpha = 1$ for GPS L1 C/A and $\alpha = 3$ for Galileo E1-B)

The value of $\tau$ for $D_{EMLP}(\tau) = 0$ yields the extra delay caused by the errors affecting the DLL. To find this delay, the zero crossing of the discriminator must be found. The slope of the EMLP discriminator must be negative for it to be a stable lock point. As the autocorrelation function of the PRN code is an even function ($\forall x \in \mathbb{R}$ s.t $f(x) = y$ & $f(-x) = y$) and considering that the signal is unaffected by errors, the only possible value of $\tau$ is zero. Therefore, the discriminator will have a zero crossing where the tracking delay is equal to 0 as shown by Figure 2.11.

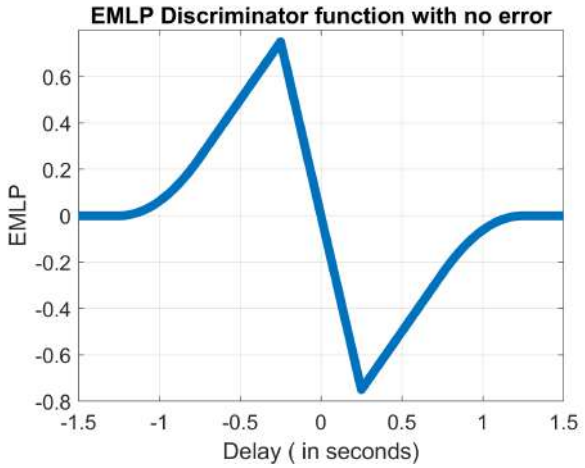


Figure 2.11: EMLP Discriminator with no error.

#### 2.4.2.3 Loop Filter

The design of DLL loop filter is made of the similar trade-offs as the ones presented for the PLL loop filter in 2.4.1.3, so they will not be presented again.

Before reaching the NCO, the discriminator and its noise are filtered. Each discriminator has a different variance. In an ideal case for a DLL where it is assumed that only additive white Gaussian noise affects the signal, the propagated noise can be estimated as zero mean Gaussian noise [Legrand et al., 2000]. For the early-late power discriminator, the variance of the noise can be expressed as:

$$\sigma_{Noise}^2 = L_c^2 \frac{B_{DLL} E_L}{2C/N_0} \left[ 1 + \frac{2}{(2 - E_L) N T_s C/N_0} \right] \tag{2.35}$$

Where:

- $L_c$ is the chip length in meters
- $B_{DLL}$ is the equivalent loop filter bandwidth of the DLL in Hz
- $C/N_0$ is the carrier to noise density ratio in Hz
- $\sigma^2_{Noise}$ is the variance of the DLL noise in meters$^2$

#### 2.4.2.4 NCO Code Generator

The NCO of the DLL functions like the one of the PLL. The difference is that the filtered discriminator input to the NCO is used to refine the frequency at which the PRN code is generated to account for the remaining code delay error.

### 2.4.3 Effect of Multipath

In a realistic scenario, tracking errors will always be present in GNSS receivers. Noise is not the sole cause for errors being propagated from the tracking stage to the code pseudorange but also multipath. As conveyed by [Bellad and Petovello, 2013] multipath can distort the autocorrelation function (ACF). This distortion depends on the number of multipaths, their phase, their amplitude, and their delay. When the ACF is distorted, this shifts the zero crossing of the discriminator function of the DLL. Multipath also shifts the total phase of the signal, changing the zero crossing of the PLL discriminator function.

When adding multipath, (2.28) and (2.29) become:

$$\begin{aligned}
I_P(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i}) sinc(\pi \varepsilon_{f_{D_i}} N T_s) \cos(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{I_P}(n) \\
I_E(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i} + \frac{E_L}{2}) sinc(\pi \varepsilon_{f_{i(k)}} N T_s) \cos(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{I_E}(n) \\
I_L(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i} - \frac{E_L}{2}) sinc(\pi \varepsilon_{f_{D_i}} N T_s) \cos(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{I_L}(n)
\end{aligned} \tag{2.36}$$

$$\begin{aligned}
Q_P(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i}) sinc(\pi \varepsilon_{f_{D_i}} N T_s) \sin(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{Q_P}(n) \\
Q_E(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i} + \frac{E_L}{2}) sinc(\pi \varepsilon_{f_{D_i}} N T_s) \sin(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{Q_E}(n) \\
Q_L(n) &= \sum_{i=1}^{N_{Mp}} \frac{A_i(n)}{2} d(n-\tau_i) R_c(\varepsilon_{\tau_i} - \frac{E_L}{2}) sinc(\pi \varepsilon_{f_{D_i}} N T_s) \sin(\pi \varepsilon_{f_{D_i}} N T_s + \varepsilon_{\theta_i}) + n_{Q_L}(n)
\end{aligned} \tag{2.37}$$

Where:

- $i$ is the index of the signal received (direct or reflected)
- $N_{Mp}$ is the number of multipath reflections

#### 2.4.3.1 PLL

In the case of a single multipath ray, the direct and reflected signal form a composite signal that can be seen as the sum of two vectors. Therefore, this vector can be broken down into two orthogonal components equivalently to the I and Q correlators. This is illustrated in Figure 2.12. The goal of the PLL is to estimate the resulting composite phase of the new signal. By using the arctangent of the Q and I components, the composite phase can be estimated.

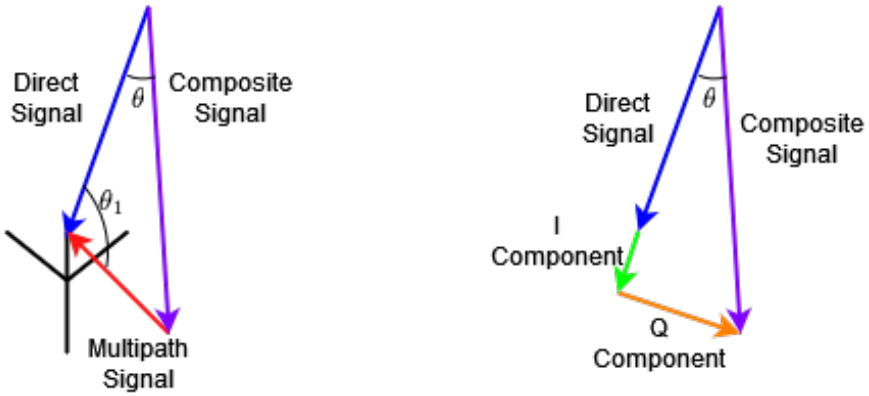


Figure 2.12: Decomposition of a single multipath ray on top of the direct signal.

Figure 2.12 can be generalized for $N_{Mp}$ multipath signals and results in [Irsigler et al., 2005]:

$$\theta = arctan\left(\frac{\sum_{i=1}^{N_{Mp}} \alpha_i R_c(\tau - \tau_i)\sin(\theta_i)}{R_c(\tau) + \sum_{i=1}^{N_{Mp}} \alpha_i R_c(\tau - \tau_i)\cos(\theta_i)}\right) \tag{2.38}$$

Where:

- $\alpha_i$ is the relative amplitude of the multipath signal ($\alpha_i = A_i/A_0$)

#### 2.4.3.2 DLL

Due to the several multipath delays, the zero crossing for the discriminator function is not at $\tau = 0$ anymore. This is illustrated by Figure 2.13 and (2.39).

$$D_{EMLP}(\tau) = \sum_{i=1}^{N_{Mp}} \frac{A_i(n)^2}{4} sinc(\pi\varepsilon_{f_D} NT_s)$$
$$\left(|\exp(j\pi\varepsilon_{f_{D_i}} NT_s + \varepsilon_{\theta_i}) R_c(\tau - \tau_i + \frac{E_L}{2})|^2 - |\exp(j\pi\varepsilon_{f_{D_i}} NT_s + \varepsilon_{\theta_i}) R_c(\tau - \tau_i - \frac{E_L}{2})|^2\right) \tag{2.39}$$

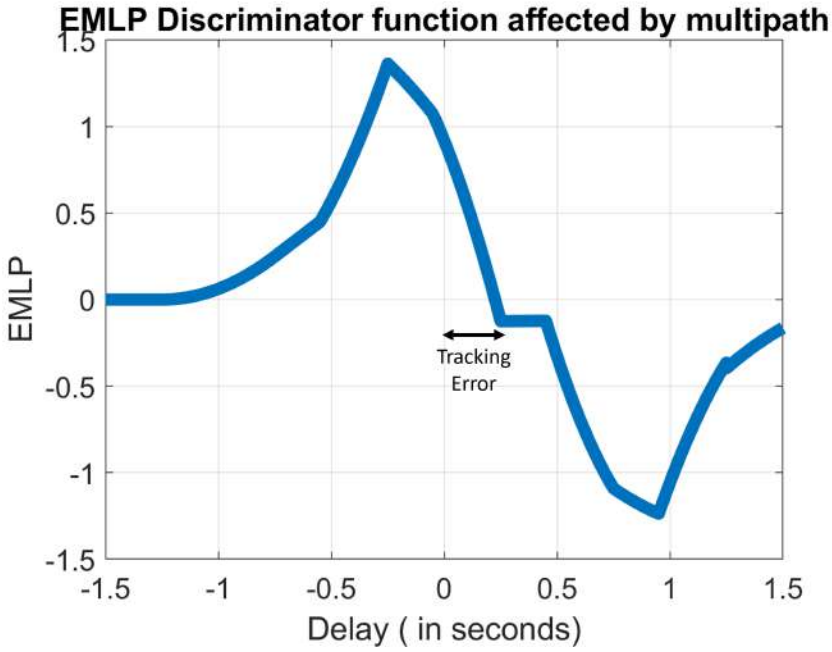


Figure 2.13: Early late power discriminator function affected by some multipath.

Assuming the DLL is at a stable lock point, the tracking bias can be obtained by multiplying the delay value at the discriminator zero crossing by the speed of light. This is shown in (2.40).

$$\exists\ t_0 \in [-T_{obs}, T_{obs}] \text{ such that } D(t_0) = 0$$
$$Tr_b = ct_0 \tag{2.40}$$

Where:

- $T_{obs}$ is the maximum delay for which the discriminator is observed
- $c$ is the speed of light in m/s
- $Tr_b$ is the tracking bias in meters

### 2.4.4 Alternate Methods

To increase the robustness of the generic tracking process, many algorithms have been developed and researched. A few examples are given in this section.

#### 2.4.4.1 FLL Aided PLL

Some loop architectures increase the robustness of the PLL by adding a Frequency Lock Loop (FLL). The FLL solely estimates the frequency component of the signal . A typical discriminator for a FLL is the cross dot product given in (2.41).

$$D_{CP}(\varepsilon_{f_D}) = \frac{I(n-1)Q(n) - I(n)Q(n-1)}{I(n-1)Q(n) + I(n)Q(n-1)} \tag{2.41}$$

The FLL is more robust to dynamics than the PLL and is often used as a pull-in loop. Meaning that after acquisition, the FLL is used without the PLL. Once a lock criterion has been met by the FLL, the loop switches to a PLL architecture for better accuracy.

#### 2.4.4.2 Pilot aided Tracking

The method presented herein is only possible is the signal being tracked is also composed of a pilot – data less – signal. It is based on the adaptive weighting of data and pilot discriminators according to their carrier to noise density ratio.The method is fully presented in [Muthuraman et al., 2008].

First the carrier to noise density ratio of the data and pilot signal are computed separately. Then, the noise variance of the discriminator outputs are estimated. Afterwards, the weights of each signal are calculated:

$$\begin{aligned} w_D &= \frac{\sigma_D^2}{\sigma_D^2 + \sigma_P^2} \\ w_P &= \frac{\sigma_P^2}{\sigma_D^2 + \sigma_P^2} \end{aligned} \tag{2.42}$$

Where:

- $\sigma_D^2$ is the variance of the tracking noise error for the data signal
- $\sigma_P^2$ is the variance of the tracking noise error for the pilot signal

Finally, both weights are combined with the discriminator outputs to yield a combined discriminator value:

$$D_c = w_P D_P + w_D D_D \tag{2.43}$$

Where:

- $D_P$ is the discriminator corresponding to the pilot signal
- $D_D$ is the discriminator corresponding to the data signal
- $D_C$ is the discriminator obtained when fusing both pilot and signal discriminators

#### 2.4.4.3 Kalman Filter Tracking

Another approach proposed in [Macabiau et al., 2012] and [Won et al., 2012] is to replace the generic PLL presented in this thesis by a Kalman filter to estimate the phase of the signal. [Macabiau et al., 2012] proposes to use the following states:

$$X_k = [\varepsilon_\theta; \varepsilon_{f_d}; \alpha] \tag{2.44}$$

Where:

- $\alpha$ is the difference between the jerk of the incoming and estimated carrier

The measurement update uses the discriminator output.

## 2.5 Observables Generation

The observables used for to compute a position are not contained in the data bits but must be estimated by the receiver. These are estimated thanks to the receiver's tracking loops. The generic algorithms to generate them will be presented in this section.

### 2.5.1 Code Pseudorange

A code pseudorange is the difference between the time of reception of the signal in the receiver's time frame with the time of transmission from the satellite in the satellite time frame multiplied by the speed of light, given in (2.45). This results in an estimated distance from the receiver to the corresponding satellite, offset by their clock. Code pseudoranges also contain statistical noise and other errors. These errors are due to the signal passing through media having different refraction indices, the receiver and satellite clocks being asynchronous, and possibly multipath.

$$\rho_{sv} = c(t_{rx}^{rx} - t_{sv}^{sv}) \tag{2.45}$$

Where:

- $\rho_{sv}$ is the code pseudorange for a given satellite in meters
- $t_{rx}^{rx}$ is the reception time in the receiver time frame in seconds
- $t_{sv}^{sv}$ is the transmission time in the satellite time frame in seconds
- $c$ is the speed of light in m/s

In most receivers, the code pseudoranges are generated from the DLL information and the preamble start. The preamble is a known sequence of bits of the navigation message with a given period for each signal. By finding this sequence, the receiver can compare the arrival times of all satellites with respect to each other. In the common reception method, the first arriving one is the reference. This reference is assigned the minimal propagation delay for that constellation; therefore, all the other satellites will have longer propagation delays. The expression of the first arriving code pseudorange is given as:

$$\rho_{1_{st}} = \tau_{ref} c \tag{2.46}$$

Where:

- $\rho_{1_{st}}$ is the code pseudorange for the first arriving satellite in meters
- $\tau_{ref}$ is the minimum time a signal can take to reach a user on Earth ($\sim$68.802 ms for GPS satellites and $\sim$77.46 ms for Galileo satellites) in seconds

The other code pseudoranges are then computed by adding the difference in time between the reference channel and the other ones:

$$\rho_{j_{th}} = \rho_{1_{st}} + c(t_{j_{th}} - t_{1_{st}}) \tag{2.47}$$

Where:

- $\rho_{j_{th}}$ is the code pseudorange for the satellite arriving in $j_{th}$ position in meters
- $t_{j_{th}}$ is the receiver time at which the $j_{th}$ satellite arrives
- $t_{1_{st}}$ is the receiver time at which the first satellite arrives

This method is highlighted in Figure 2.14 provided by [Marco Rao, 2012].

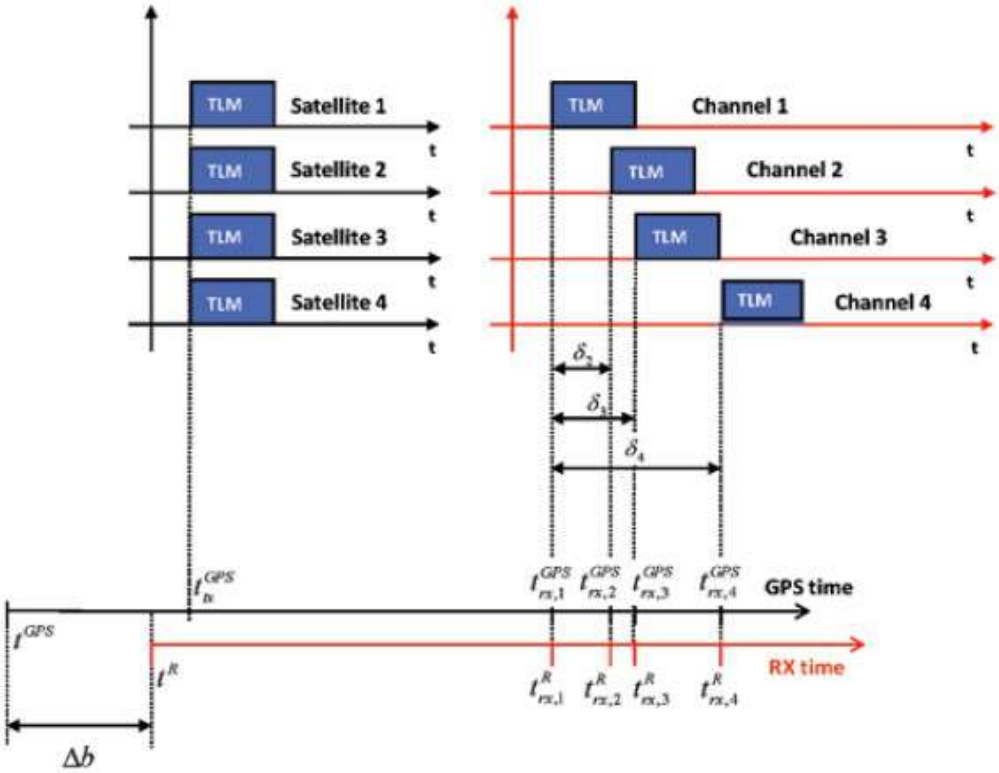


Figure 2.14: Common Reception Technique illustration [Marco Rao, 2012].

Once code pseudoranges are generated, they can be used for position calculation. However, as mentioned earlier, these measurements are still affected by numerous errors. The code pseudorange model is given in (2.48).

$$\rho^j_{Rx} = (\vec{r}^j - \vec{r}_{Rx}) \cdot \vec{e_{Rx}}^j + c(dt_{Rx} - dt^j) + T^j_{Rx} + I^j_{Rx} + \delta^j_{\rho,Rx} + \varepsilon^j_{\rho,Rx} \tag{2.48}$$

Where:

- $\vec{r}^j$ is the satellite's position in meters
- $\vec{r}_{Rx}$ is the receiver's position in meters
- $\vec{e_{Rx}}^j$ is the line of sight vector
- $dt_{Rx}$ is the receiver clock error in seconds
- $dt^j$ is the satellite clock error in seconds
- $T^j_{Rx}$ is the troposphere error in meters
- $I^j_{Rx}$ is the ionosphere error in meters
- $\delta^j_{\rho,Rx}$ is the multipath of the given observable in meters
- $\varepsilon^j_{\rho,Rx}$ is the noise of the given observable in meters
- $\rho^j_{Rx}$ is the code pseudorange in meters

### 2.5.2 Carrier Phase

The carrier phase is the difference between the received phase and the transmitted phase of the signal. This equates to:

$$\phi^j_{Rx} = [\phi^j_{NCO} - \phi^j] \mod N \tag{2.49}$$

Where:

- $\phi^j_{Rx}$ is the carrier phase in cycles
- $\phi^j_{NCO}$ is the phase estimated by the NCO of the PLL
- $\phi^j$ is the phase of the signal at transmission
- $N$ is the number of periods of the carrier wave from the satellite to the receiver

The total number of cycles occurring between the satellite and receiver is unknown. When taking into account the error sources, (2.49) can be rewritten as:

$$\phi^j_{Rx} = f_c(t_{Rx} + dt_{Rx} + \phi^j_{Rx,0}) - f_c(t^j + dt^j + \phi^j_0) + \varepsilon^j_{\phi,Rx} \tag{2.50}$$

Where:

- $t_{Rx}$ is the receiver time
- $\phi^j_{Rx,0}$ is the initial phase of the receiver
- $t^j$ is the satellite time
- $\phi^j_0$ is the initial phase of the satellite

When multiplied by the wavelength of the signal $\lambda$, the carrier phase measurement can be expressed as range measurement in meters, which can be used for positioning if the ambiguity is dealt with.

$$\Phi^j_{Rx} = (\vec{r}^j - \vec{r}_{Rx}) \cdot \vec{e_{Rx}}^j + c(dt_{Rx} - dt^j) + T^j_{Rx} - I^j_{Rx} + \lambda N^j_{Rx} + \delta^j_{\phi,Rx} + \varepsilon^j_{\phi,Rx} \tag{2.51}$$

Where:

- $\lambda N^j_{Rx}$ is the ambiguity term
- $\delta^j_{\phi,Rx}$ is the multipath of the given observable in meters
- $\varepsilon^j_{\phi,Rx}$ is the noise of the given observable in meters
- $\Phi^j_{Rx}$ is the carrier phase in meters

### 2.5.3 Doppler

The Doppler measurement is given by the estimated Doppler shift of the receiver and can be expressed as:

$$D^j = \hat{f}_{Rx_j} - f_{c_j} \tag{2.52}$$

$$D^j = f_{c_j} \frac{(\vec{v_{Rx}} - \vec{v_j}) \cdot \vec{e_{Rx}}^j}{c + \vec{v_j} \cdot \vec{e^j}} \tag{2.53}$$

Where:

- $D^j$ is the Doppler measurement in Hz
- $\hat{f}_{Rx_j}$ is the estimated frequency of the signal when received in Hz
- $f_{c_j}$ is the central frequency of the signal when emitted by the satellite in Hz
- $\vec{e^j}$ is the velocity unitary vector of the satellite

Since the speed of the signal is much higher than the velocity of the receiver or satellite, (2.53) can be simplified as:

$$D^j \approx f_{c_j} \frac{(v_{Rx} - v_j) \cdot \vec{e_{Rx}}^j}{c} \tag{2.54}$$

It is shown in [Gaglione, 2015] that the full Doppler measurement can be expressed as:

$$D^j = \frac{f_{c_j}}{c} \left( (\vec{v^j} - \vec{v}_{Rx}) \cdot \vec{e_{Rx}}^j + c(\dot{dt}_{Rx} - \dot{dt}^j) \right) + \varepsilon^j_{D,Rx} \tag{2.55}$$

Where:

- $\vec{v^j}$ is the satellite's velocity in m/s
- $\vec{v}_{Rx}$ is the receiver's velocity in m/s
- $dt_{Rx}$ is the receiver clock drift in seconds$^2$
- $dt^j$ is the satellite clock drift in seconds$^2$
- $\varepsilon^j_{D,Rx}$ is the noise of the given observable in meters

The Doppler measurement can also be expressed as the derivative of the phase measurement for a differentiating interval equal to the PLL refresh rate. Yet, the Doppler measurement and carrier phase measurement can be assumed as uncorrelated. Indeed, the Doppler measurement is affected by noise over the correlation period whereas the carrier phase measurement is generated over time intervals (refresh rate of 1-10 Hz).

### 2.5.4 Effect of Multipath

Since multipath affects the tracking loops, shown in 2.4.3, and the GNSS observables are generated from the tracking loops, it is evident that they are also affected by multipath. The effect multipath has on each measurement will be detailed in this section.

#### 2.5.4.1 Code Pseudorange

The code pseudorange is the noisier of the three GNSS observables and is most affected by multipath. For multipath weaker than the direct signal and a correlator spacing of one chip, the maximum error on GPS L1 C/A is one-half chip, approximately 147 meters [Braasch and Dierendonck, 1999]. In 2.4.3.2, the method to compute the tracking bias was detailed. Once this bias is passed through the low-pass filter of the tracking process, this filtered bias is what is propagated onto the code pseudorange measurement as follows.

$$\rho_b = LPF(Tr_b) \tag{2.56}$$

Where:

- $LPF$ is the low-pass filter of the DLL
- $\rho_b$ is the code pseudorange bias induced from the tracking bias in meters

This method requires an estimation of the discriminator function with multiple correlators and is used in 6.2 for the proposed multipath detection.

Most commonly, the error added to the code pseudorange is characterized by the multipath error envelope (MEE). The MEE provides the maximum error a single multipath ray with a given relative amplitude, $\alpha$, can add to the code pseudorange. It is called an envelope because the maximum error is given for an in-phase and an out-of-phase multipath ray corresponding to the two worst case scenarios. All potential errors from this multipath ray will land within these two bounds.

The maximum in-phase error can be obtained by adding the autocorrelation function of the PRN multiplied by the chosen relative amplitude to the nominal autocorrelation function of the PRN. The maximum out-of-phase error can be obtained in the same way but by subtracting the autocorrelation of the PRN affected by multipath. Then the induced tracking bias for each delay is computed. This is mathematically expressed in (2.57) - (2.61).

$$R_{c_{in-phase}}(\tau \mid \tau_i) = R_c(\tau) + \alpha * R_c(\tau - \tau_i) \tag{2.57}$$

$$R_{c_{out-of-phase}}(\tau \mid \tau_i) = R_c(\tau) - \alpha * R_c(\tau - \tau_i) \tag{2.58}$$

$$D_{in-phase}(\tau \mid \tau_i) = R_{c_{E,in-phase}}(\tau \mid \tau_i) - R_{c_{L,in-phase}}(\tau \mid \tau_i) \tag{2.59}$$

$$D_{out-of-phase}(\tau \mid \tau_i) = R_{c_{E,out-of-phase}}(\tau \mid \tau_i) - R_{c_{Late,out-of-phase}}(\tau \mid \tau_i) \tag{2.60}$$

$$\exists\ t_0 \in [-T_{obs}, T_{obs}] \text{ such that } D(t_0 \mid \tau_i) = 0$$

$$Tr_b = ct_0 \tag{2.61}$$

Where:

- $R_c$ is the autocorrelation function of the PRN
- $\alpha$ is the amplitude of the multipath with respect to the LOS signal
- $E$ represents the early version of the ACF of the PRN
- $L$ represents the late version of the ACF of the PRN

- $in - phase$ represents the ACF of the PRN for the in-phase multipath
- $out - phase$ represents the ACF of the PRN for the out-phase multipath
- $T_{obs}$ is the observation interval on which the ACF is studied in seconds
- $Tr_b$ is the tracking bias in meters

The tracking bias (2.61) gives the code pseudorange error induced by the in-phase or out-of phase multipath with the input delay. This process can be repeated for all desired multipath delays to the MEE. Figure 2.15 illustrates the MEE on GPS L1 C/A for two different EL spacing for $\alpha = 0.5$.

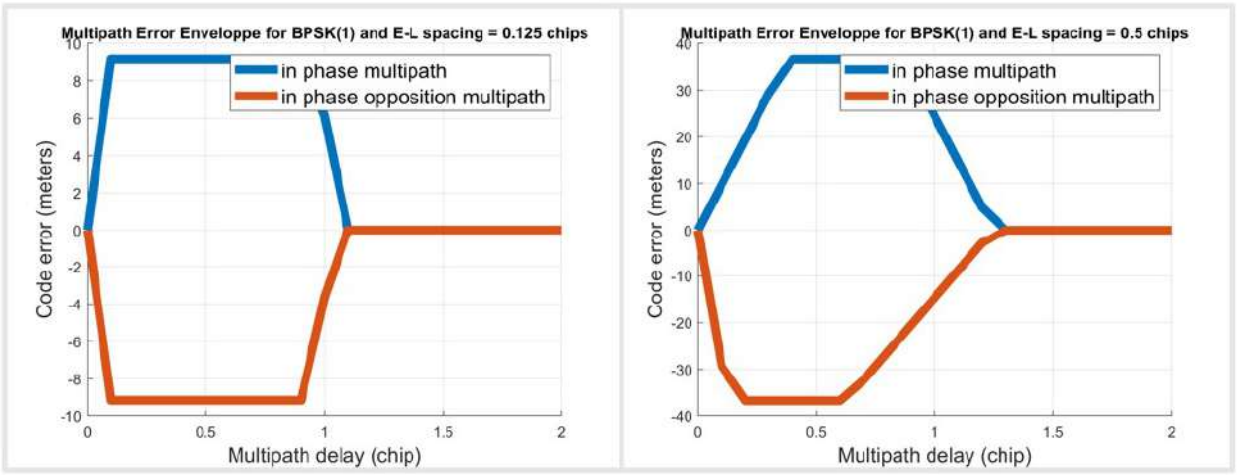


Figure 2.15: MEE of code pseudorange for two different EL spacing with an $\alpha = 0.5$.

As shown in [Van Dierendonck et al., 1992], the multipath error decreases with a lower EL spacing. However, this reduces the stability of the loop as the linearity region of the discriminator is reduced. Furthermore, increase the relative amplitude of the multipath ray increases the MEE.

#### 2.5.4.2 Carrier Phase

To evaluate the multipath error on the carrier phase measurement, the MEE can also be computed. By using (2.38) for one reflection, it is equal to:

$$\theta = arctan\left(\frac{\alpha_1 R_c(\tau - \tau_1)\sin(\theta_1)}{R_c(\tau) + \alpha_1 R_c(\tau - \tau_1)\cos(\theta_1)}\right) \tag{2.62}$$

Determining the MEE for a carrier phase measurement is more complex as the maximum phase error is not given for an obvious $\theta_1$. Hence, the maximum value of $\theta$ must be selected for a given delay and all possible values of $\theta_1$. Figure 2.15 illustrates the MEE of the carrier phase measurement on GPS L1 C/A for $\alpha = 0.5$.

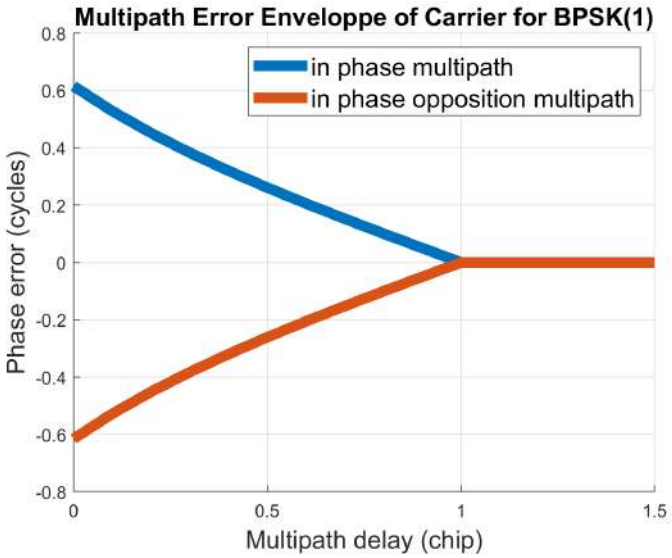


Figure 2.16: Simplified MEE of carrier phase for an $\alpha = 0.5$.

Figure 2.16 is a simplified model of the MEE for the carrier phase measurement. As shown by [Braasch, 2017], this simplification is valid especially for narrow correlators. This model assumes that $\tau = 0$ whereas the exact model computes the maximum and minimum angle error for each potential multipath delay and total delay.

#### 2.5.4.3 Doppler

By multiplying the Doppler measurement with the wavelength of the signal, the pseudo-rate measurement is obtained [Gaglione, 2015]. The pseudo-rate can be seen as the derivative of the code pseudorange measurement. Hence, the multipath affecting the Doppler measurement can be written as:

$$\delta^j_{D,Rx} = \frac{\dot{\delta}^j_{\rho,Rx}}{\lambda} \tag{2.63}$$

Where:

- $\delta^j_{D,Rx}$ is the multipath affecting the Doppler measurement

The effect of multipath on the Doppler measurement is hard to quantify theoretically as its MEE cannot be computed. The presented term in (2.63) was shown to cause poor velocity estimations in dense multipath environments in [Sadrieh et al., 2012] and [Matera et al., 2018].

## 2.6 Conclusions

In this chapter, the generic algorithms required to process a GNSS signal were presented. Some more complex ones were also discussed to improve the robustness of the GNSS signal processing chain. Furthermore, the method to compute GNSS observables was also detailed. These observables will be used in chapter 3 to explain how a position solution is computed. Lastly, the effect that multipath on both the signal processing algorithms and measurements was examined. This chapter highlighted the effect that multipath has on the signal processing

blocks and showed why this error can severely impact the positioning solution. This impact is the motivation to detect and mitigate its effect for positioning solutions as shown in Chapters 6 and 7.

CHAPTER 3

# GNSS Positioning Algorithms

Once the GNSS observables are generated, they may be used in order to compute a position, and potentially velocity and time (PVT), solution. The following chapter will present the two most implemented estimators when using only GNSS observables: the LSE estimator and the Kalman filter. The Kalman filter will also be presented when the GNSS observables are tightly coupled to an IMU. Another category of estimator will also be presented: robust estimators, which are useful when a portion of the input data is affected by outliers.

## Chapter Contents


3.1 Positioning Principles . . . 36
3.2 Least Squares Estimator . . . 38
3.2.1 Classical Estimator . . . 38
3.2.2 Weighted Least Squares Estimator . . . 41
3.2.3 Dilution of Precision . . . 42
3.3 Kalman Filter . . . 43
3.3.1 GNSS Only . . . 46
3.3.2 GNSS/INS Hybridization . . . 49
3.4 Robust Estimator . . . 54
3.4.1 M Estimator . . . 55
3.4.2 S Estimator . . . 56
3.4.3 MM Estimator . . . 57
3.5 Conclusions . . . 57


## 3.1 Positioning Principles

GNSS positioning is based on the trilateration principle. Indeed, by finding the point of intersection of the code pseudoranges or carrier phase measurements, if the ambiguity is accounted for, the position solution can be found. As the name indicates, trilateration uses three measurements to find the intersection point. However, in GNSS, the receiver clock bias from the true constellation time also has to be estimated. Therefore, when employing measurements from a single constellation, a minimum of four code pseudoranges or carrier

phase measurements must be used to estimate the 3D position and the receiver clock bias. This is illustrated on Figure 3.1.

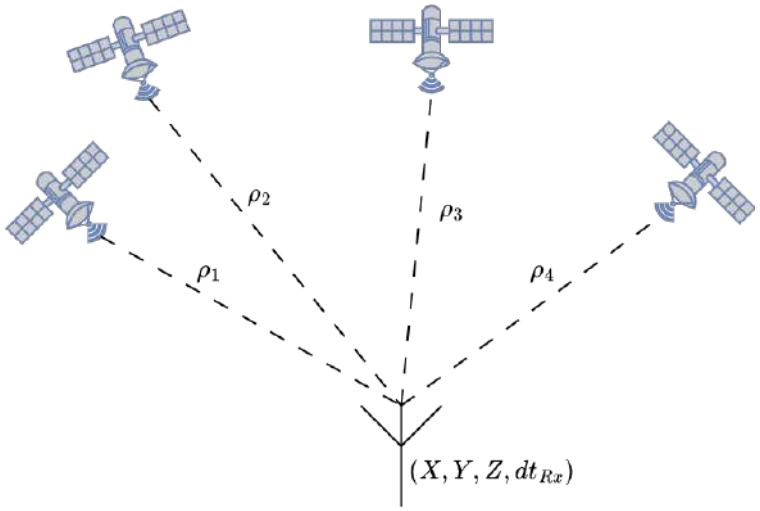


Figure 3.1: Tri-lateration for GNSS purposes.

GNSS signals are now transmitted by several constellations. Adding more measurements is beneficial as it increases the geometric diversity of the satellite to receiver links. However, constellations are not perfectly synchronized in time. Thus, with every extra constellation used, the time bias between the reference and the added one must be estimated. This means that the minimum number of satellites required to compute a position must follow the two criteria:

$$N_{SVs} \geq 4 + N_{const} - 1 \tag{3.1}$$

$$N_{SVs} = \sum_{i=1}^{N_{const}} N_{SVs,i} \text{ s.t } N_{SVs,i} \in [1, N_{SV/Const}] \tag{3.2}$$

Where:

- $N_{SVs}$ is the number of satellites used for a position computation
- $N_{const}$ is the number of constellations used for a position computation
- $N_{SV/Const}$ is the number of satellites per constellation

From the code pseudorange and carrier phase measurement equations given in (2.48) and (2.50), on top of the receiver clock bias, multipath, and noise, other errors still affect the measurement. Indeed, the tropospheric, ionospheric errors, and satellite clock bias are the largest remaining errors. These errors must be accounted for to increase the accuracy of the positioning solution.

The satellite clock bias can be estimated thanks to the transmitted satellite clock parameters. It can then be removed from the observable.

The ionospheric error occurs when the signal passes through the ionosphere and is then

diffracted. When using single frequency measurements, the ionospheric error can be corrected by 64.8% by using the Klobuchar model [Wang et al., 2018] on GPS L1 C/A. This error is frequency-dependent and its first order component [El-naggar, 2011], making up more than 99% of the error [Datta-Barua et al., 2008], can be removed from the code pseudorange or carrier phase by combining dual frequency measurements as such:

$$Obs_{IF} = \frac{Obs_1 f_1^2 - Obs_2 f_2^2}{f_1^2 - f_2^2} \tag{3.3}$$

Where:

- $Obs_x$ are either the code pseudorange or carrier phase measurement for a given frequency $x$ in meters
- $f_x$ the frequency of the signal processed in Hz
- $Obs_{IF}$ the observable with the first order ionospheric delay removed in meters

The tropospheric error happens when the signal is diffracted by the troposphere before reaching the antenna of the receiver. The tropospheric error is composed of a dry and wet component.

$$T = T_d + T_w \tag{3.4}$$

Where:

- $T_d$ is the dry component of the tropospheric error in meters
- $T_w$ is the wet component of the tropospheric error in meters

The tropospheric error depends on the elevation of the satellite. A widely used model to correct the tropospheric error is the mapping of Niell detailed in [Boehm et al., 2006].

Once these errors are corrected, the measurements can be used with greater accuracy to compute a position solution. The most common algorithms to do so are presented in the following sections.

## 3.2 Least Squares Estimator

In the following section, the position computation will be demonstrated when using code pseudoranges. The ambiguities of the carrier phase measurements are assumed to still affect the measurement and accounting for them is outside the scope of this section.

### 3.2.1 Classical Estimator

The least squares estimator (LSE) minimizes the sum of squared residuals. The residual is defined as the difference between the observation and a linear function predicting the observation. This can be denoted as:

$$J(\vec{x}) = \left( \sum_{i=1}^{N} \left( y_i - h(\vec{x}) \right)^2 \right) \tag{3.5}$$

$$J(\vec{x}) = \left( \sum_{i=1}^{N} r_i^2 \right) \tag{3.6}$$

Where:

- $J(\vec{x})$ is the loss function
- $y_i$ is the observation
- $h(\vec{x})$ is the linear function of the form $H\vec{x}$
- $r_i$ is the residual
- $N$ is the number of observations

To apply the LSE to GNSS, the code pseudoranges are used as the observables $y_i$ or the carrier phase measurements if the ambiguity are estimated/fixed prior to using the LSE, which is seldom done. Recalling the code pseudorange model:

$$\rho_{Rx}^j = (\vec{r}^j - \vec{r}_{Rx}) \cdot \vec{e_{Rx}}^j + c(dt_{Rx} - dt^j) + T_{Rx}^j + I_{Rx}^j + \delta_{\rho,Rx}^j + \varepsilon_{\rho,Rx}^j \tag{3.7}$$

From (3.7), it is easily seen that the code pseudorange is not linear so the LSE cannot be used as is. The first step is to linearize the measurement around a point and apply Taylor's expansion theorem. This yields:

$$\rho_{Rx}^j = \rho_0^j + \frac{\partial \rho^j}{\partial x}\Delta x + \frac{\partial \rho^j}{\partial y}\Delta y + \frac{\partial \rho^j}{\partial z}\Delta z + \frac{\partial \rho^j}{\partial cdt_{Rx}}\Delta cdt_{Rx} \tag{3.8}$$

Since the satellite positions and the linearization point coordinates are known, the distance $\rho_0^j$ can be computed. Assuming this as the linear function of (3.5), the residual can be written as:

$$\Delta\rho_{Rx}^j = \frac{\partial \rho^j}{\partial x}\Delta x + \frac{\partial \rho^j}{\partial y}\Delta y + \frac{\partial \rho^j}{\partial z}\Delta z + \frac{\partial \rho^j}{\partial cdt_{Rx}}\Delta cdt_{Rx} + \epsilon_{\rho,Rx}^j \tag{3.9}$$

Where:

- $\epsilon_{\rho,Rx}^j = \delta_{\rho,Rx}^j + \varepsilon_{\rho,Rx}^j + \xi_{\rho,Rx}^j$ is the total error of the code pseudorange in meters with $\xi_{\rho,Rx}^j$ being the remaining ionosphere, troposphere, and satellite clock error

When generalizing (3.9) for all N code pseudoranges, it becomes:

$$\begin{bmatrix} \Delta\rho_{Rx}^1 \\ \vdots \\ \Delta\rho_{Rx}^N \end{bmatrix} = \begin{bmatrix} \frac{\partial \rho_1}{\partial x} & \frac{\partial \rho_1}{\partial y} & \frac{\partial \rho_1}{\partial z} & \frac{\partial \rho_1}{\partial cdt_{Rx}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \rho_N}{\partial x} & \frac{\partial \rho_N}{\partial y} & \frac{\partial \rho_N}{\partial z} & \frac{\partial \rho_N}{\partial cdt_{Rx}} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta cdt_{Rx} \end{bmatrix} + \begin{bmatrix} \epsilon_{\rho,Rx}^1 \\ \vdots \\ \epsilon_{\rho,Rx}^N \end{bmatrix} \tag{3.10}$$

This equation is equivalent to a linear system of the form:

$$\Delta\vec{y} = H\Delta\vec{x} \tag{3.11}$$

Since all code pseudoranges are linearized around a point, (3.5) can be written as:

$$J(\Delta\vec{x}) = (\Delta\vec{y} - H\Delta\vec{x})^T(\Delta\vec{y} - H\Delta\vec{x}) \tag{3.12}$$

To minimize the cost function, its derivative can be computed and find when it is equal to 0.

$$\partial\Big((\Delta\vec{y} - H\Delta\vec{\hat{x}})^T(\Delta\vec{y} - H\Delta\vec{\hat{x}})\Big) = 0 \tag{3.13}$$

Where:

- $\Delta\vec{\hat{x}}$ is the estimated position delta for which the cost function is minimized

From [Blewitt, 2000], it can be shown that the value at which $\Delta\hat{x}$ is minimized is:

$$\Delta\vec{\hat{x}} = (H^T H)^{-1} H^T \Delta\vec{y} \tag{3.14}$$

The $\Delta y$ vector is the generated code pseudoranges after linearization from the receiver, but the $H$ matrix must be computed to retrieve the estimated position delta. $H$ is given as:

$$\begin{bmatrix} \frac{\partial\rho_1}{\partial x} & \frac{\partial\rho_1}{\partial y} & \frac{\partial\rho_1}{\partial z} & \frac{\partial\rho_1}{\partial cdt_{Rx}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial\rho_N}{\partial x} & \frac{\partial\rho_N}{\partial y} & \frac{\partial\rho_N}{\partial z} & \frac{\partial\rho_N}{\partial cdt_{Rx}} \end{bmatrix} = \begin{bmatrix} \frac{x_0 - x_1}{\hat{d}_1} & \frac{y_0 - y_1}{\hat{d}_1} & \frac{z_0 - z_1}{\hat{d}_1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_0 - x_N}{\hat{d}_N} & \frac{y_0 - y_N}{\hat{d}_N} & \frac{z_0 - z_N}{\hat{d}_N} & 1 \end{bmatrix} \tag{3.15}$$

Where:

- $\hat{d}_j$ is the estimated distance between the receiver and the satellite obtained by removing errors from the pseudorange (troposphere, ionosphere, satellite clock bias etc.)

As $\Delta\vec{\hat{x}}$ is the estimated position difference between the linearization point and the true position, the estimated receiver position is written as:

$$\vec{\hat{x_{Rx}}} = \vec{x_0} + \Delta\vec{\hat{x}} \tag{3.16}$$

Where:

- $\Delta\vec{\hat{x_{Rx}}}$ is the estimated receiver position and its clock bias error with regards to the linearization point
- $\vec{x_0}$ is the linearization point

Since the non-linear LSE is used, the linearization process presented from (3.8) to (3.16) needs to be repeated for an infinite amount of iterations so that this estimator would lead to a perfect position assuming there are no errors. However as indicated by the term $\vec{n}$, errors also affect the LSE estimation process. The LSE is only optimal when each element of $\vec{n}$ denoted $n_i$ is assumed as a zero-mean Gaussian noise written as $\sim N(0, \sigma^2)$. Another assumption made is that GNSS measurements are uncorrelated with one another. Hence, the covariance matrix of $n$ is then equal to:

$$R_{LSE} = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma^2 \end{bmatrix} \tag{3.17}$$

The LSE algorithm can also be used to compute the velocity of the receiver by using Doppler measurements. Identically to using code pseudoranges, the Doppler measurement can be linearized around a given point to use the LSE. The states to estimate are the velocities (x, y, and z) and the receiver clock drift as shown in 3.18.

$$\begin{bmatrix} \Delta v_x \\ \Delta v_y \\ \Delta v_z \\ \Delta c\dot{dt}_{Rx} \end{bmatrix} \tag{3.18}$$

The residuals and measurement matrix are given in (3.19) and (3.20).

$$\Delta D^j_{Rx} = \frac{\partial D^j}{\partial v_x}\Delta v_x + \frac{\partial D^j}{\partial v_y}\Delta v_y + \frac{\partial D^j}{\partial v_z}\Delta v_z + \frac{\partial D^j}{\partial \dot{cdt}_{Rx}}\Delta \dot{cdt}_{Rx} + \epsilon^j_{D,Rx} \tag{3.19}$$

$$H = \begin{bmatrix} \frac{\partial D_1}{\partial v_x} & \frac{\partial D_1}{\partial v_y} & \frac{\partial D_1}{\partial v_z} & \frac{\partial D_1}{\partial \dot{cdt}_{Rx}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial D_N}{\partial v_x} & \frac{\partial D_N}{\partial v_y} & \frac{\partial D_N}{\partial v_z} & \frac{\partial D_N}{\partial \dot{cdt}_{Rx}} \end{bmatrix} \begin{bmatrix} \frac{x_0-x_1}{\hat{d}_1} & \frac{y_0-y_1}{\hat{d}_1} & \frac{z_0-z_1}{\hat{d}_1} & 1 \\ \vdots & \vdots & \vdots & \\ \frac{x_0-x_N}{\hat{d}_N} & \frac{y_0-y_N}{\hat{d}_N} & \frac{z_0-z_N}{\hat{d}_N} & 1 \end{bmatrix} \tag{3.20}$$

Then the velocity increment from the linearization point, denoted as $\Delta\vec{\hat{v}}$ in (3.21), can be estimated by solving the same equation as for the position:

$$\Delta\vec{\hat{v}} = (H^T H)^{-1} H^T \Delta\vec{y} \tag{3.21}$$

### 3.2.2 Weighted Least Squares Estimator

The WLSE is a variant of the classical LSE. This estimator still assumes that the noise affecting the measurements is centered Gaussian noise. However, the noise for all measurements is not considered identical on all measurements. Therefore the cost function equates to:

$$J(\vec{x}) = \sum_{i=1}^{N} r_i^2(n_i) \tag{3.22}$$

The measurements are still assumed uncorrelated with a centered Gaussian distribution, [Ruppert and Wand, 1994] shows that (3.14) can be written as:

$$\vec{\hat{x}} = (H^T R^{-1}_{WLSE} H)^{-1} H^T R^{-1}_{WLSE} \vec{y} \tag{3.23}$$

The covariance matrix of the measurement noise is different from the ordinary LSE described above (3.17) and is now equal to:

$$R_{WLSE} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_N^2 \end{bmatrix} \tag{3.24}$$

Each measurement has a different effect on the estimation of $\vec{\hat{x}}$ hence the name weighted least squares. If a measurement has a high variance with respect to other measurement variances, this measurement will be less relied upon during the parameter estimation .

In GNSS, the noise of a code pseudorange increases with lower elevations of the satellite with respect to the user as shown in [Paziewski et al., 2019]. A common mapping function to determine the variance of the measurement is:

$$\sigma_i^2 = \frac{\sigma_D^2}{\sin(el_i)^2} \tag{3.25}$$

Where:

- $\sigma_i^2$ is the variance of measurement $i$ in meters
- $\sigma_D^2$ is the variance of a measurement for a 90 degree elevation in meters
- $el_i$ is the elevation of satellite $i$ with respect to the user in radians

### 3.2.3 Dilution of Precision

To evaluate the confidence in the estimates, the covariance of the LSE estimator must be computed. Identically to the state estimate $\vec{\hat{x}}$, the incoming error will be mapped in the same way:

$$\vec{\hat{e}} = \hat{x} - x \tag{3.26}$$

$$\vec{\hat{e}} = (H^T H)^{-1} H^T \vec{n} \tag{3.27}$$

The covariance of the resulting estimation error is computed as:

$$P = \mathbb{E}(\vec{\hat{e}}\vec{\hat{e}}^T) \tag{3.28}$$

Where:

- $P$ is the covariance matrix of the estimation error

If the covariance of the measurement noise is diagonal, it can be then shown that for the LSE it is equal to [Blewitt, 2000]:

$$\begin{aligned} P &= \sigma^2 (H^T H)^{-1} \\ P &= \sigma^2 Q \end{aligned} \tag{3.29}$$

Where:

- $Q = (H^T H)^{-1}$ is the geometry matrix

Thanks to the geometry matrix, the overall quality of the LSE solution can be computed. These quality indicators are called the dilution of precision.

These indicators are given for the LSE case [Langley, 1999].

$$GDOP = \sqrt{Q_{11} + Q_{22} + Q_{33} + Q_{44}} \tag{3.30}$$

$$PDOP = \sqrt{Q_{11} + Q_{22} + Q_{33}} \tag{3.31}$$

$$HDOP = \sqrt{Q_{11} + Q_{22}} \tag{3.32}$$

$$VDOP = \sqrt{Q_{33}} \tag{3.33}$$

$$TDOP = \sqrt{Q_{44}} \tag{3.34}$$

Where:

- $\sigma_X$ is the standard deviation of the X component (North, East, Up, or Time)
- $Q_{MN}$ is the element of the geometry matrix of element $(M \times N)$

In GNSS, the lower these values are the better the geometry of the satellites is and the better the positioning accuracy is.

## 3.3 Kalman Filter

The Kalman Filter is a recursive algorithm that estimates states based on the knowledge of the parameters of the system model and measurements. This estimation process is recursive as it uses previous information of the system coupled with new incoming measurements to estimate the next values of the chosen states. This fusion of previous state estimates and current observations aims to minimize the mean of the squared error. The elements of the KF that must be chosen by the algorithm designer are briefly described below.

Every KF is composed of a **state vector** which contains the parameters that the user wants to estimate and of measurements coming from external sensors. For GNSS based applications, the position and receiver clock bias are always present (error or absolute) as states, and other states may be added such as the velocity, other sensor errors, etc. GNSS observables presented in 2.5 are used as measurement inputs to the filter. The KF assumes that current state is a linear function of the previous states and white noise [Groves, 2008]. The measurement model is modeled as a linear function of the true state vector and white noise. The true state and true observation model can be expressed as:

$$x(k) = f(x, k-1) + u(k-1) \tag{3.35}$$

$$z(k) = h(x, k) + v(k) \tag{3.36}$$

Where:

- $x$ is the state vector of the system
- $f$ is the state transition function
- $u$ is the state process noise assumed as AWGN
- $z$ is the measurement vector
- $h$ is the measurement function
- $v$ is the measurement noise assumed as AWGN

To each estimated state vector $\hat{x}_k$, is associated a **state covariance matrix**, $P_k$, which illustrates the quality of the state estimation along with the correlation between the state estimates.

The **state transition matrix**, $\Phi_k$, depicts how the model is projected to evolve over time. A process noise covariance matrix, $Q_k$, is also defined to account for the uncertainties of the state evolution that may be unmodeled or changing over time.

The **observation vector**, $z_k$, are measurements coming from one or different sources that are fed to the filter in order to fuse this information with the projected state estimate coming from the state transition matrix. It is possible that measurements come in at different times based on their source and/or frequency update. This is possible in the KF but measurements must then either be on the same time scale or the time bias must be accounted for.

Each measurement vector has its associated **measurement covariance matrix**, $R_k$, depicting the noise of each measurement and the potential correlation between incoming measurements.

The **measurement observation matrix**, $H_k$, links the behavior of the measurements with respect to the state vector.

The Kalman filter assumes that state dynamics and the observation process are linear, as well as the normal distribution of noise in the state dynamics and input measurements [Ghadrdan et al., 2012]. The block diagram in Figure 3.2 depicts the steps taken by the KF to update the state estimates for one iteration.

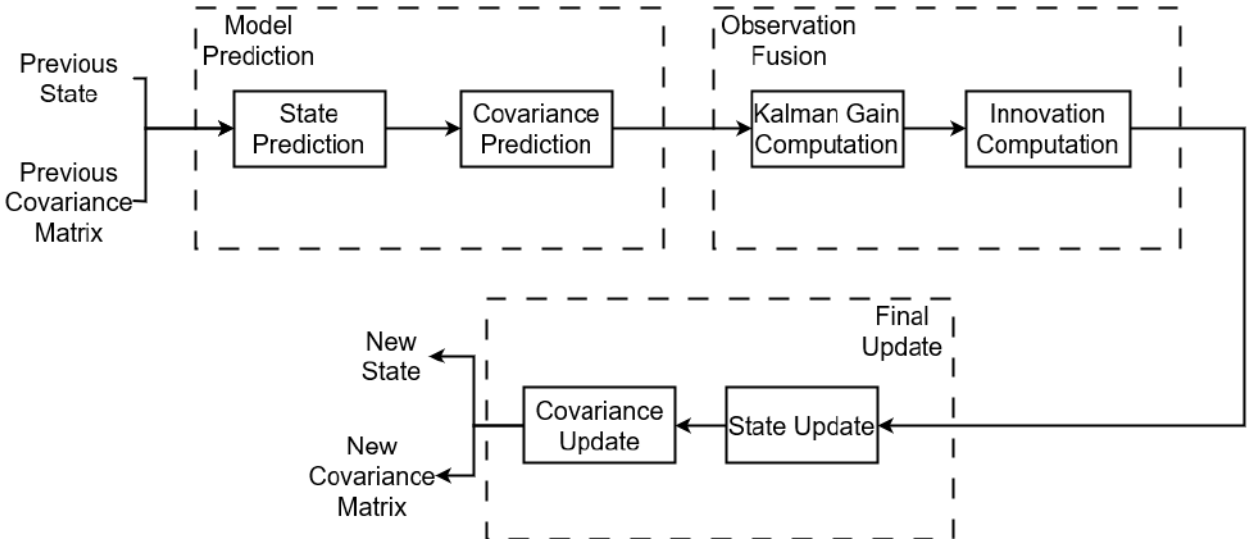


Figure 3.2: Kalman Filter steps during one iteration.

Each step can be mathematically described. The **state prediction** is equal to:

$$\hat{x}_k^- = \Phi_{k-1}\hat{x}_{k-1}^+ \tag{3.37}$$

Where:

- $\hat{x}_k^-$ is the propagated state at iteration $k$
- $x_{k-1}^+$ is the estimated state at iteration $k-1$
- $\Phi_{k-1}$ is the state transition matrix from the previous iteration to the current one

The **covariance prediction** is equal to:

$$P_k^- = \Phi_{k-1}P_{k-1}^+\Phi_{k-1}^T + Q_{k-1} \tag{3.38}$$

Where:

- $P_k^-$ is the propagated covariance matrix at iteration $k$
- $P_{k-1}^+$ is the estimated covariance matrix at iteration $k-1$

- $Q_{k-1}$ is the covariance matrix of the state transition model noise

The **Kalman gain** is computed as:

$$K_k = P_k^- H_k^T \Big( H_k P_k^- H_k^T + R_k \Big)^{-1} \tag{3.39}$$

Where:

- $K_k$ is the Kalman gain matrix
- $H_k$ is the measurement matrix
- $R_k$ is the measurement noise covariance matrix

The **innovation** is provided by:

$$i_k = z_k - H_k \hat{x}_k^- \tag{3.40}$$

Where:

- $i_k$ is the innovation vector
- $z_k$ is the observation vector

Finally, the **updated state** and **covariance matrix** are computed as:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k i_k \tag{3.41}$$

$$P_k^+ = (I_N - K_k H_k) P_k^{-1} \tag{3.42}$$

Where:

- $\hat{x}_k^+$ is the estimated state at iteration $k$
- $P_k^+$ is the estimated covariance state matrix at iteration $k$
- $I_N$ is the identity matrix of size $N \times N$

As mentioned earlier and illustrated by (3.37) and (3.41), the KF assumes linearity between the states and the measurement and prediction models. However, GNSS models are non-linear as shown in 3.2.1. Hence, the KF must be adapted to be used for GNSS purposes.

The extended Kalman filter is a widely used type of KF for non-linear measurements. The EKF linearizes the state prediction around the previous state update and the measurement model is linearized around the state prediction. Assuming the system model and measurement model are not linear with respect to the state vector, they can be written as:

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+) \tag{3.43}$$

$$i_k = z_k - h(\hat{x}_k^-) \tag{3.44}$$

It is possible to linearize (3.43) around the previous state update point so that:

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+) + \left[ \frac{\partial f}{\partial x} \right]_{x=\hat{x}_{k-1}^+} \delta \hat{x}_k^- \tag{3.45}$$

Which can be expressed as:

$$\hat{x}_k^- = \hat{x}_{k-1}^+ + F_{k-1}\delta\hat{x}_{k-1}^+ \tag{3.46}$$

For a discrete EKF, (3.46) can be further simplified into [Groves, 2008]:

$$\hat{x}_k^- = \Phi_{k-1}\delta\hat{x}_{k-1}^+ \tag{3.47}$$

Where:

- $\Phi_k = \exp(F_{k-1}\Delta_T)$ and $\Delta_T$ is the sampling frequency of the state update

And the innovation is linearized around the state prediction:

$$i_k = z_k - h(\hat{x}_k^-) - \left[\frac{\partial h}{\partial x}\right]_{x=\hat{x}_k^-} \delta\hat{x}_k^- \tag{3.48}$$

As the measurement model of the EKF is assumed to be of the form $z(k) = h(k) + w_k$, (3.48) can be simplified as:

$$i_k = H_k\delta\hat{x}_k^- + w_k \tag{3.49}$$

Where:

- $H_k$ is the Jacobian matrix of the function $h$ as a function of the state estimates
- $w_k$ is the measurement noise at iteration $k$

### 3.3.1 GNSS Only

This section will present a conventional EKF using only GNSS measurements: code pseudoranges, carrier phase, and Doppler observables. GNSS satellite positions are computed in the Earth Centered Earth Fixed (ECEF) frame [Zhu, 1994] so the presented EKF computes the position of the receiver in the ECEF as well. Indeed, this avoids coordinate frame rotations. The state vector used in this EKF implementation is equal to:

$$x_k = [r_k^e \;\; v_k^e \;\; dt_{Rx_k} \;\; \dot{dt}_{Rx_k} \;\; dt_{GPS-GAL,k} \;\; ztd_k \;\; N_1 \ldots \; N_n] \tag{3.50}$$

Where:

- $r_k^e$ is the receiver position in the ECEF in meters
- $v_k^e$ is the receiver velocity in the ECEF in meters per second
- $dt_{Rx_k}$ is the receiver clock bias in meters
- $\dot{dt}_{Rx_k}$ is the receiver clock drift in meters per second
- $dt_{GPS-GAL,k}$ is the inter-system bias between GPS and Galileo in meters
- $ztd_k$ is the zenith tropospheric delay in meters
- $N_x$ is the ambiguity term for the corresponding measurement

Each carrier phase ambiguity is estimated in this filter to make use of the accuracy brought by the carrier phase measurements. The zenithal tropospheric delay is also estimated because in (3.4), the wet component is a non-deterministic variable. The total tropospheric delay depends on the elevation and is at its lowest, in a statistical sense, when the satellite is at the zenith with respect to the receiver ($\pi/2$ elevation).

#### 3.3.1.1 State Transition Model

It can be shown (in Appendix A) that the state transition matrix is equal to:

$$\Phi_{k-1} = \begin{bmatrix} I_3 & \Delta_T I_3 & 0 & 0 & 0 & 0 & 0_{1\times N} \\ 0_3 & I_3 & 0 & 0 & 0 & 0 & 0_{1\times N} \\ 0_{1\times 3} & 0_{1\times 3} & 1 & \Delta_T & 0 & 0 & 0_{1\times N} \\ 0_{1\times 3} & 0_{1\times 3} & 0 & 1 & 0 & 0 & 0_{1\times N} \\ 0_{1\times 3} & 0_{1\times 3} & 0 & 0 & 1 & 0 & 0_{1\times N} \\ 0_{1\times 3} & 0_{1\times 3} & 0 & 0 & 0 & 1 & 0_{1\times N} \\ 0_{N\times 3} & 0_{N\times 3} & 0_{N\times 1} & 0_{N\times 1} & 0_{N\times 1} & 0_{N\times 1} & I_N \end{bmatrix} \tag{3.51}$$

Where:

- $I_N$ is the identity matrix of size N
- $0_{N\times M}$ is a zero matrix of size [N, M]

#### 3.3.1.2 Process Noise Covariance Matrix

The process noise covariance matrix $Q_k$ is defined as:

$$Q_{k-1} = \mathbb{E}\Big[u_{k-1}u_{k-1}^T\Big] \tag{3.52}$$

The states are assumed independent and the noise of the states are all assumed to be white. (3.52) is equal to:

$$Q_{k-1} = diag(\begin{bmatrix} 0_3 \\ Q_{vel} \\ 0 \\ Q_{rxDrift} \\ Q_{interGnss} \\ Q_{Tropo} \\ Q_{amb} \end{bmatrix}) \tag{3.53}$$

Where:

- $Q_{state}$ matrices are equal to the variance of the state noise multiplied by the state refresh rate.

The state process noise is zero for both the position and receiver clock bias as their temporal evolution is known and modeled in the state transition model. In the state transition matrix, the acceleration is assumed to be null over the sampling interval which is false; thus, an uncertainty on the velocity evolution is typically used to account for acceleration changes. The process noise of the clock drift is dependent on the oscillator of the receiver. The remaining process noise values are subject to tuning according to the performance of the filter, but it is assumed that the inter-system bias and tropospheric error have slow temporal evolution. The ambiguity process noise value depends on the assumption on the frequency of loss of locks. If loss of locks are frequent, the ambiguity process noise should be higher for decreased convergence time.

#### 3.3.1.3 Observation Model

All three GNSS observables are used in this filter, therefore, there are three different observation models to build. As the three observations are assumed independent, the matrix can be computed as (developed in Appendix A):

$$H_k = \begin{bmatrix} H_{k,\rho} \\ H_{k,\Phi} \\ H_{k,Doppler} \end{bmatrix} \tag{3.54}$$

By reusing (3.48, $H = \left[\frac{\partial h}{\partial x}\right]_{x=x_k^-}$) and the GNSS observable models presented in 2.5, each observation matrix can be written as:

$$H_{k,\rho} = \begin{bmatrix} -\vec{e_{Rx}}^{1} & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -\vec{e_{Rx}}^{N} & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 \end{bmatrix} \tag{3.55}$$

$$H_{k,\phi} = \begin{bmatrix} -\vec{e_{Rx}}^{1} & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & \delta_{1,1} & \dots & \delta_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -\vec{e_{Rx}}^{N} & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & \delta_{N,1} & \dots & \delta_{N,N} \end{bmatrix} \tag{3.56}$$

$$H_{k,Doppler} = \begin{bmatrix} 0_{1\times3} & -\vec{e_{Rx}}^{1} & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{1\times3} & -\vec{e_{Rx}}^{N} & 0 & 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{3.57}$$

Where:

- $C_{ind}$ is equal to 0 for GPS observables and 1 for Galileo observables
- $ztd$ is the zenithal tropospheric delay
- $\delta_{j,i}$ is the ambiguity value for the carrier phase measurement, if and only if $i = j$ then $\delta_{i,i} = 1$ and 0 otherwise

The measurement noise covariance matrix $R$ accounts for noise like errors in the measurements such as the thermal noise, satellite clock noise, multipath variations etc. However, this term does not account for remaining constant biases. In the presented EKF, all measurements are assumed uncorrelated with one another and have the same mapping function as the one presented in (3.25). An element of R can be characterized as:

$$\sigma_i^2 = \frac{\sigma_{obs}^2}{\sin(el_i)^2} \tag{3.58}$$

The whole matrix can then be defined as:

$$R_k = diag(\begin{bmatrix} \sigma_{1,obs1}^2 \\ \vdots \\ \sigma_{N,obs1}^2 \\ \sigma_{1,obs2}^2 \\ \vdots \\ \sigma_{N,obsN}^2 \end{bmatrix}) \tag{3.59}$$

Where:

- $\sigma^2_{i,obsj}$ is the variance of the $i$th measurement for the $j$th different type of measurement

### 3.3.2 GNSS/INS Hybridization

Adding an inertial navigation system (INS) to the GNSS solution is widely implemented. An INS uses an IMU to compute a positioning, orientation, and velocity solution. As mentioned earlier, the fusion of GNSS and INS overcomes some of the shortcomings of both positioning solutions. Despite the added complexity, the coupling of the two is beneficial for the accuracy of the position [Vezinet, 2014]. Three main couplings exist with INS/GNSS fusion are detailed below and are illustrated in Figure 3.3.

1. **Loose coupling**: This method fuses the position solution of the INS and the GNSS receiver together. This is the easiest method to implement but has relatively poor performance especially in challenging GNSS conditions as shown in [Falco et al., 2017] and [Dong et al., 2020].

2. **Tight coupling**: This algorithm combines the GNSS observables with the INS to compute a position with the INS solution considered as the reference position. This solution can operate with less satellites than required as a pure GNSS position is not computed unlike in loose coupling [Dong et al., 2020].

3. **Ultra-tight coupling**: This approach associates the INS solution with the correlator outputs from the tracking loops. The benefit of this method is that the accuracy of tracking loops and observables can be improved thanks to the INS aiding [Jovancevic et al., 2004] but it is the most complex coupling.

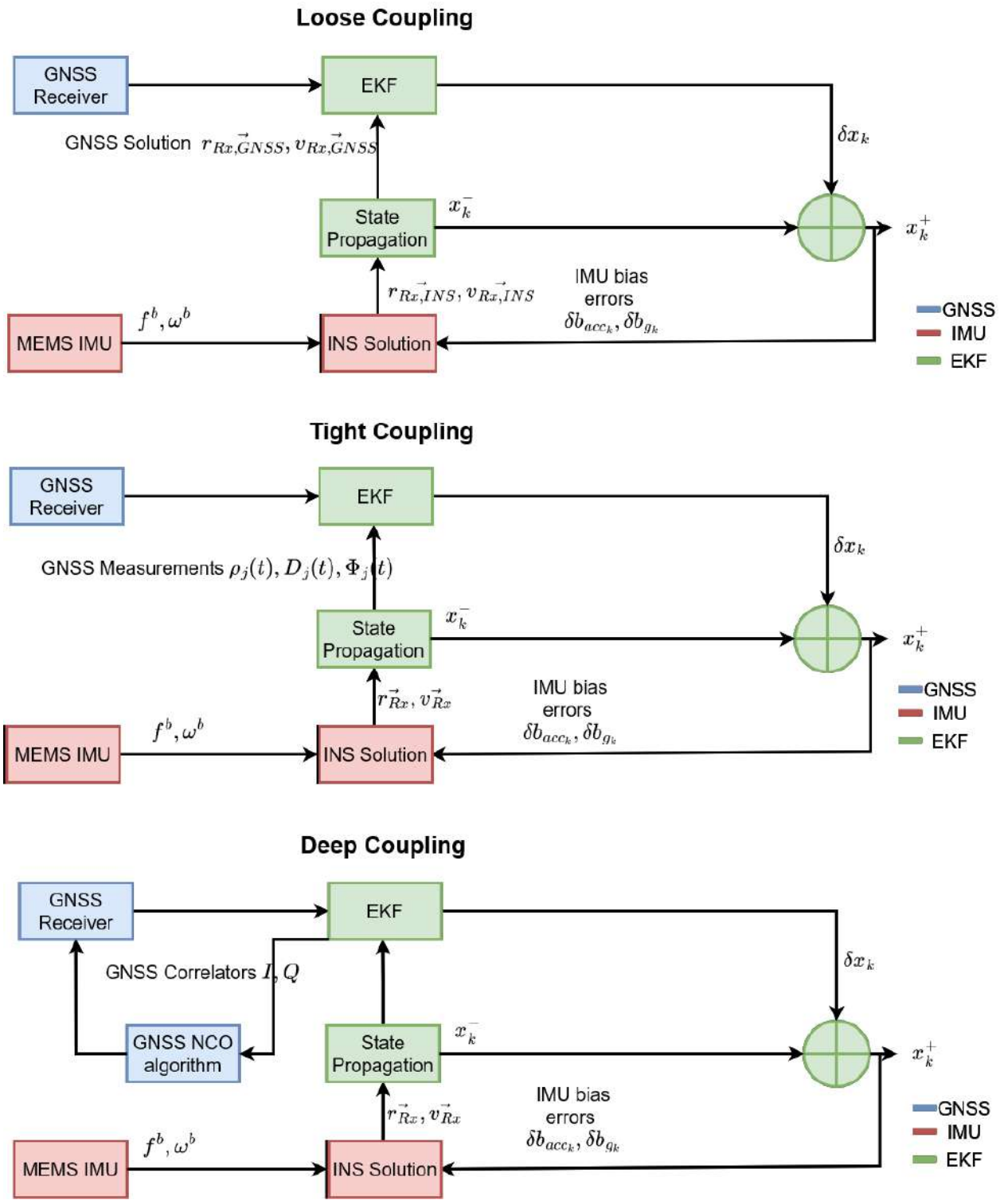


Figure 3.3: Different GNSS/INS hybridization techniques.

This section will present an EKF using an INS tightly coupled to the GNSS observables thanks to its robustness to poor GNSS conditions and medium complexity for implementation on a real-time receiver. This EKF will be an error, closed state filter.

A KF can be in open or closed loop structure. In the case of an error state GNSS/INS

hybridization, the open loop configuration uses the errors of the states to correct the INS solution and results in a corrected and a raw INS solution. This is especially useful when the IMU is of high quality for integrity uses. On the other hand, a closed loop uses the error of the estimated states to correct the INS during the computation process. The error estimates are zeroed each iteration. Closed loop filters are useful to limit the impact of linearization errors when using low-quality IMU [Groves, 2008]. As a MEMS IMU is used in the eHermes, a closed loop filter was designed.

Another design choice for the EKF is whether to have a total state or an error state filter. A total state filter estimates the absolute values of the states whereas an error state filter estimates the error of the states. In GNSS/INS hybridization, using an error state filter is useful to keep the error state values low which ensures the validity of small angle approximations.

The state vector of this tight-coupling fusion is given as:

$$\delta x_k = [\delta r_k^e \;\; \delta v_k^e \;\; \delta dt_{Rx_k} \;\; \delta \dot{dt}_{Rx_k} \;\; \delta dt_{GPS-GAL,k} \;\; \delta ztd_k \;\; \delta N_1 \ldots \;\; \delta N_n \;\; \delta \psi_k^e \;\; \delta b_{acc_k} \;\; \delta b_{gyro_k}] \tag{3.60}$$

Where:

- $\delta \psi_k^e$ are the errors of the receiver attitude in the ECEF in radians.
- $\delta b_{acc_k}$ are the errors of the IMU accelerometer bias in $m/s^2$.
- $\delta b_{gyro_k}$ are the errors of the IMU gyroscope bias in rad/s.

To obtain the total states from the error state EKF, the error states are added to the INS states (except for the attitude angles). This results in:

$$x_{k,Total} = \delta x_k + x_{k,INS} \tag{3.61}$$

And for the attitude update, the attitude error can be multiplied to estimate the rotation matrix from the body to the desired frame:

$$R_b^e = (I_3 - [\delta \psi_k^e \times]) R_{b,INS}^e \tag{3.62}$$

Where:

- $R_b^e$ is the rotation matrix from the body frame to the ECEF.
- $R_b^e$ is the rotation matrix from the body frame to the ECEF computed with the INS estimates.
- $[\delta \psi_k^e \times]$ denotes the skew matrix of the attitude errors in brackets which is equal to: $\begin{bmatrix} 0 & -\delta\psi_{k,3}^e & \delta\psi_{k,2}^e \\ \delta\psi_{k,3}^e & 0 & -\delta\psi_{k,1}^e \\ -\delta\psi_{k,2}^e & \delta\psi_{k,1}^e & 0 \end{bmatrix}$

Then, the attitude angles can then be obtained from the matrix, which are called the Euler angles.

$$\psi_k^e = \begin{bmatrix} arctan2(R_{b3,2}^e, R_{b3,3}^e) \\ -arcsin(R_{b1,3}^e) \\ arctan2(R_{b2,1}^e, R_{b1,1}^e) \end{bmatrix} \tag{3.63}$$

Where:

- $R^e_{b,X,Y}$ is the element [X, Y] of the rotation matrix from the body frame to the ECEF.

To obtain the state estimates $X_{k,INS}$, the INS solution must be computed. Mechanization is the process by which IMU measurements are used in order to compute the displacement, velocity, and rotation of the IMU over time. The particular equations of the mechanization depend on which frame the computation is performed. The main steps are:

1. Update of the attitude from the gyroscope measurements
2. Update of the velocity from the accelerometer measurements
3. Update of the position by integration of the velocity

#### 3.3.2.1 State Transition Model

To compute the state transition matrix, the same reasoning applied in 3.3.1.1. It can be shown (in A) that the state transition matrix is equal to:

$$\Phi_{k-1} =$$

$$\begin{bmatrix}
I_3 & \Delta_T I_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & 0_3 & 0_3 \\
0_3 & -2\Omega_{ie}\Delta_T & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & \Delta_T[-R^e_b f^b\times] & -\Delta_T R^e_b & 0_3 \\
0_{1\times3} & 0_{1\times3} & 1 & \Delta_T & 0 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\
0_{1\times3} & 0_{1\times3} & 0 & 1 & 0 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\
0_{1\times3} & 0_{1\times3} & 0 & 0 & 1 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\
0_{1\times3} & 0_{1\times3} & 0 & 0 & 0 & 1 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\
0_{N\times3} & 0_{N\times3} & 0_{N\times1} & 0_{N\times1} & 0_{N\times1} & 0_{N\times1} & I_N & 0_{N\times3} & 0_{N\times3} & 0_{N\times3} \\
0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & \Delta_T\Omega_{ie} & 0_3 & -\Delta_T R^e_b \frac{\pi}{180} \\
0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & I_3 & 0_3 \\
0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & 0_3 & I_3
\end{bmatrix} \tag{3.64}$$

Where:

- $\Omega_{ie} = \begin{bmatrix} 0 & -\omega_{ie} & 0 \\ \omega_{ie} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ with $\omega_{ie} = 7.292115 \times 10^{-5}$ rad/s being the WGS84 Earth's angular rate
- $R^b_{imu}$ is the rotation matrix from the IMU's frame to the body frame
- $R^e_b$ is the rotation matrix from the body frame to the ECEF
- $f^b$ is the accelerometer reading rotated to the body frame with respect to the inertial frame
- $\omega^b$ is the gyrometer reading rotated to the body frame with respect to the inertial frame
- $[v\times]$ denotes the skew matrix of the vector in brackets which is equal to: $\begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$

#### 3.3.2.2 Process Noise Covariance Matrix

The process noise covariance matrix $Q_k$ is defined as:

$$Q_{k-1} = diag(\begin{bmatrix} Q_{GNSS} \\ Q_{att} \\ Q_{acc} \\ Q_{gyro} \end{bmatrix}) \tag{3.65}$$

Where:

- $Q_{GNSS}$ is the state covariance matrix of (3.53)

The values of the added states of the state covariance matrix ($Q_{att}, Q_{acc},$ and $Q_{gyro}$) depend on the quality of the IMU. In this thesis, a low-cost MEMS IMU was used. To quantify the process noise, the Allan variance of the sensor was determined. The attitude noise was given by the gyroscope standard deviation noise. The accelerometer and gyroscope bias process noise were estimated with the lowest value of the Allan deviation curve as done in [Woodman, 2007].

#### 3.3.2.3 Observation Model

Identically to the GNSS only EKF, all three GNSS observables are used. The observations are still assumed independent. The observation model is similar but takes into account the lever arm between the IMU and the GNSS receiver.

$$H_{k,\rho} = \begin{bmatrix} -e_{\vec{INS}}{}^1 & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 & -e_{\vec{INS}}{}^1 R_b^e(-\vec{p}_{Rx}^b\times) & 0_{1\times3} & 0_{1\times3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ -e_{\vec{INS}}{}^N & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 & -e_{\vec{INS}}{}^N R_b^e(-\vec{p}_{Rx}^b\times) & 0_{1\times3} & 0_{1\times3} \end{bmatrix} \tag{3.66}$$

$$H_{k,\phi} = \begin{bmatrix} -e_{\vec{INS}}{}^1 & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & \delta_{1,1} & \dots & \delta_{1,N} & -e_{\vec{INS}}{}^1 R_b^e(-\vec{p}_{Rx}^b\times) & 0_{1\times3} & 0_{1\times3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ -e_{\vec{INS}}{}^N & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & \delta_{N,1} & \dots & \delta_{N,N} & -e_{\vec{INS}}{}^N R_b^e(-\vec{p}_{Rx}^b\times) & 0_{1\times3} & 0_{1\times3} \end{bmatrix} \tag{3.67}$$

$$H_{k,Doppler} = \begin{bmatrix} 0_{1\times3} & -e_{\vec{INS}}{}^1 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & -e_{\vec{INS}}{}^1 R_b^e(\vec{v}_{INS}^b\times) & 0_{1\times3} & -e_{\vec{INS}}{}^1 R_b^e(\vec{p}_{Rx}^b\times)\frac{\pi}{180} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0_{1\times3} & -e_{\vec{INS}}{}^N & 0 & 1 & 0 & 0 & 0 & \dots & 0 & -e_{\vec{INS}}{}^N R_b^e(\vec{v}_{INS}^b\times) & 0_{1\times3} & -e_{\vec{INS}}{}^N R_b^e(\vec{p}_{Rx}^b\times)\frac{\pi}{180} \end{bmatrix} \tag{3.68}$$

Where:

- $p_{Rx}^b$ is the lever arm between the antenna and IMU in the body frame
- $v_{INS}^b$ is the INS antenna velocity in the body frame

The measurement covariance matrix is identical to the one defined in (3.59).

## 3.4 Robust Estimator

Robust estimators are designed to limit the effect of outliers in the estimation of a given set of parameters. There are three main classes of robust estimators:

1. **M estimators**: Like the LSE, M estimators are a generalisation of maximum likelihood estimators (MLE). They can be expressed as:

$$M_{est} : \hat{\theta}_M = min_\theta \Big( \sum_{i=1}^{N} \rho(\frac{r_i(\theta)}{\hat{\sigma}}) \Big) \tag{3.69}$$

   Where:

   - $\rho$ is the robust loss function
   - $r$ is the residual element
   - $N$ is the number of residuals
   - $\theta$ is the parameter to estimate
   - $\hat{\theta}_M$ is the estimated parameter
   - $\hat{\sigma}$ is the estimated scale of errors

2. **L estimators**: They are obtained by linearly combining order statistics, an example is the median which is an L-estimator. Each observation is arranged according to its value and then multiplied by its associated weight. For the median, it is equal to $x_{((N+1)/2)}$ if $N$ is odd and the average between the $(N/2)$ and $(N+1)/2$ -th order statistics if $N$ is even. L estimators are solved as:

$$L_{est} : \hat{\theta}_L = \sum_{i=1}^{N} w_i x_{(i)} \tag{3.70}$$

   Where:

   - $x_{(i)}$ is the $i$-th order statistic
   - $w_i$ is the $i$-th weight

3. **R estimators**: These estimators minimize the sum of residuals based on assigning a rank to each residual according to its value. They are given as:

$$R_{est} : \hat{\theta}_R = min_\theta \Big( \sum_{i=1}^{N} \rho(R_i) r_i(\theta) \Big) \tag{3.71}$$

   Where:

   - $R_i = R(r_i)$ is the rank of the residual $r_i$

The performance of robust estimators are characterized by three factors:

1. The choice of the **loss function**, popular choices are the Huber or Tukey's biweight functions.

2. The **breakdown point** of the estimator, which is the proportion of outliers contained in the dataset at which the robust estimator yields large erroneous estimates.

3. The **efficiency** of the estimator is its asymptotic behavior and variance when no outliers are present [Zoubir et al., 2018].

In this work, only M estimators were studied for their effect on positioning solutions. As mentioned in [de Menezes et al., 2021], M estimators are easier to handle as a function bounds the estimator. The S and MM estimators presented are variants of the basic M estimator.

### 3.4.1 M Estimator

This estimator assumes that the observations are independent but not necessarily identically distributed [Stefanski and Boos, 2002]. To ensure robustness against outliers the residuals must be bounded, so an estimate of the scale of errors (with its computation given in (3.75) is used along with a bounded loss function. To minimize (3.69), assuming the loss function chosen is differentiable, it can be written as:

$$\frac{\partial}{\partial\theta}\Big(\sum_{i=1}^{N}\rho(\frac{r_i(\theta)}{\hat{\sigma}})\Big) = 0 \tag{3.72}$$

By applying the chain rule:

$$\Big(\sum_{i=1}^{N}\frac{\partial r_i(\theta)/\hat{\sigma}}{\partial\theta}\psi(\frac{r_i(\theta)}{\hat{\sigma}})\Big) = 0 \tag{3.73}$$

Where:

- $\psi$ is the influence function that must be bounded which is equal to $\psi(x) = \frac{\partial}{\partial x}\rho(x)$

To solve this equation, a weight function is defined, which is equal to $w(x) = \psi(x)/x$, to make (3.73) equivalent to a WLSE problem.

$$\sum_{i=1}^{N}\frac{\partial r_i(\theta)/\hat{\sigma}}{\partial\theta}\frac{r_i(\theta)}{\hat{\sigma}}\Big(w(\frac{r_i(\theta)}{\hat{\sigma}})\Big) = 0 \tag{3.74}$$

To estimate the scale of residuals, robust measures of scale such as the inter-quantile range or the median absolute deviation (MAD) need to be used. This is due to their resistance to outliers unlike the variance which is greatly affected by outliers. The MAD can be defined as:

$$\hat{\sigma} = \text{Med}(|x - \text{Med}(x)|) \tag{3.75}$$

This scale estimate can be multiplied by a normalizing constant to make the scale estimate consistent with the standard deviation of a normally distributed dataset or other chosen distributions. For the Gaussian distribution the constant of the MAD is roughly equal to 1.4815 [Rousseeuw and Croux, 1993]. The proof is given in Annex B. Therefore (3.75), for estimated Gaussian distributions, is equal to:

$$\hat{\sigma} = 1.4815\ \text{Med}(|x - \text{Med}(x)|) \tag{3.76}$$

To apply (3.74) for GNSS positioning purposes, the residual is obtained by the linearization of the code pseudorange around a point. The equation is recalled in (3.77). It is worthwhile to note that, identically to the LSE algorithm, robust estimators can be used to compute the velocity of the receiver with Doppler measurement.

$$\Delta\rho^j_{Rx} = \frac{\partial\rho^j}{\partial x}\Delta x + \frac{\partial\rho^j}{\partial y}\Delta y + \frac{\partial\rho^j}{\partial z}\Delta z + \frac{\partial\rho^j}{\partial cdt_{Rx}}\Delta cdt_{Rx} + \epsilon^j_{\rho,Rx} \tag{3.77}$$

Afterwards, the residuals are scaled by the scale estimate. For GNSS purposes, the normalizing constant chosen is for the Gaussian distribution.

Then, based on the selected loss function, the weights for each value are computed. Then the position solution is re-estimated with a process taking into account the new weights, for example the WLSE algorithm. This process is then repeated with the new residuals until a convergence on the position has been reached, which is depicted in Figure 3.4.

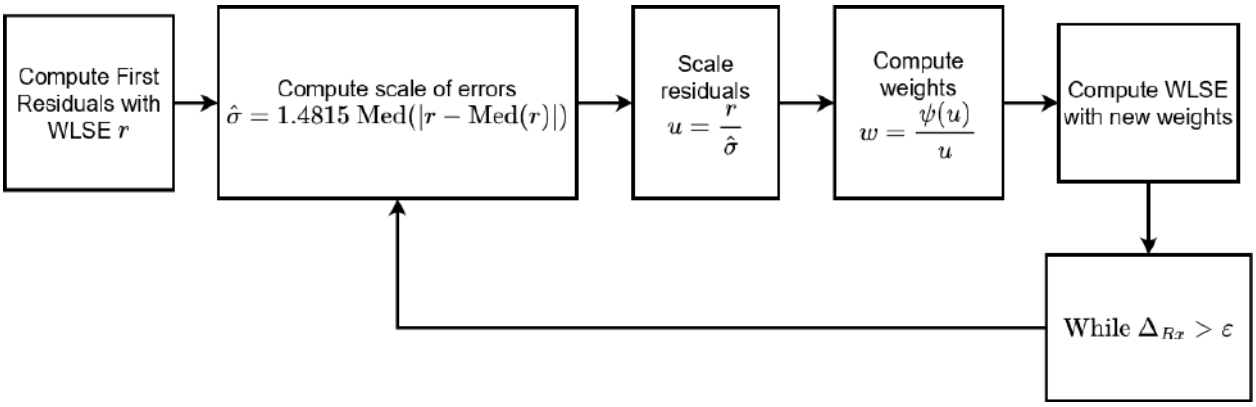


Figure 3.4: M Estimator Algorithm for GNSS positioning.

### 3.4.2 S Estimator

S estimators are a variation of M estimators which minimize the dispersion of the residuals [Toka and Cetin, 2010]. These estimators tend to have high breakdown points but at the cost of low efficiency. The S estimator can be defined as:

$$S_{est} : \hat{\theta}_S = min_\theta\Big(s(r_1(\theta), \ldots, r_N(\theta))\Big) \tag{3.78}$$

The dispersion of $s(r_1(\theta), \ldots, r_N(\theta))$ is defined as the solution of:

$$K = \frac{1}{N}\sum_{i=1}^{N}\rho\left(\frac{r_i(\theta)}{\hat{\sigma}_S}\right) \tag{3.79}$$

Where:

- $K$ is the expected value of the loss function for a standard normal distribution
- $\hat{\sigma}_S$ is the estimated scale of errors with the S estimator

To use the S estimator for GNSS positioning, the algorithm is almost unchanged with respect to M estimators presented in Figure 3.4. First, a targeted breakdown point must be defined for this estimator. The other change comes from estimating the scale residual each iteration instead of computing it from the MAD before scaling the residuals.

### 3.4.3 MM Estimator

MM estimators target a high efficiency and a high breaking point, essentially improving upon the S estimator. To do so, two loss functions are used. One loss function is used to estimate the scale estimate of the residuals like the S estimator. Then the set of parameters is estimated with the S-estimated scale residual. The MM estimator is given as:

$$MM_{est} : \hat{\theta}_M = min_\theta \Big( \sum_{i=1}^{N} \rho_{MM}(\frac{r_i(\theta)}{\hat{\sigma}_S}) \Big) \tag{3.80}$$

Where:

- $\rho_{MM}(x)$ is the loss function used once the scale residual is estimated

To compute the MM estimator applied to GNSS positioning, the first iteration is where the scale estimate of the S estimator is computed using a loss function for the S estimator $\rho_S$. After this scale estimate has converged, the other iterations use $\rho_{MM}$ as the loss function and the estimated scale of errors.

## 3.5 Conclusions

This chapter detailed three of the most widely used positioning techniques for GNSS purposes. Firstly, the basic LSE algorithm was described and how it can be implemented for a GNSS receiver. Secondly, conventional GNSS-only and INS/GNSS tight coupling EKFs were presented and their model detailed. Lastly, robust estimators applied to GNSS positioning were defined. Each presented algorithm has its upsides and downsides.

An LSE based algorithm is simple to implement and fast, but performs poorly when the data does not follow a Gaussian law. For example, when multipath occurs, the LSE performance can be severely impacted. The EKF based solution uses the precise carrier phase measurements and is less sensitive to large biases, but is very sensitive with respect to tuning and slowly growing errors, is computationally heavy, and adds a sensor if the INS coupling is used. Lastly, robust estimators can deal with outlier measurements making them useful for obstructed GNSS environments but lack the LSE efficiency when the data is not severely impacted by unmodeled errors. However, one downside is the difficulty to predict their performance (i.e. positioning error bounds, protection levels etc.).

As the eHermes is a GNSS receiver equipped of a micro-electromechanical systems (MEMS), implementing an EKF with GNSS/INS hybridization maximizes the receiver's hardware. The implemented solution is presented in Chapter 5. However, characterizing the performance of a hybridized solution is challenging due to the difficulty to tune the filter. It has also been observed that the main driver of the accuracy of the solution is the code measurements. Therefore, a focus was placed on GNSS code solutions only to simplify the analysis with the assumption that improving this solution would apply to the hybridized filter.

To improve the GNSS code only solution, a method to limit the impact of large multipath biases on the positioning solution is presented in Chapter 7.

CHAPTER 4

# Convolutional Neural Networks

In this thesis, convolutional neural networks (CNN) were studied within the scope of GNSS multipath classification. As shown in Chapter 2, multipath can severely impact the signal processing of a GNSS receiver. Indeed, multipath impacts the tracking loops of the receiver, more specifically their correlator outputs. Furthermore, multipath depends on the environment of the receiver and modeling it is computationally expensive. The eHermes can has the flexibility to increase the number of correlator outputs and their values are accessible. Since the eHermes is bound to operate in different conditions, determining whether GNSS multipath affects the code pseudorange was investigated. This was done using machine learning, more specifically CNNs.

CNNs were popularized by [Krizhevsky et al., 2012] for their capacity to deal with image inputs. They were inspired by the way animals analyze images [Hubel DH, 1968]. As their name suggests, they make use of the convolution operation to analyze patterns in an image. This convolution process resembles how filters are applied to images for corner edge detection for example. This chapter will highlight the differences between a CNN and an artificial neural network and describe the different layers that constitute a CNN. The importance of a loss function and the gradient descent to recompute the weights of the network will be explored. Lastly, two common ways to limit the model from overfitting will be described.

## Chapter Contents


4.1 Differences with Artificial Neural Networks . . . 59
4.2 Layers . . . 60
4.2.1 Convolutional Layer . . . 60
4.2.2 Pooling Layer . . . 62
4.2.3 Output Layer . . . 62
4.3 Predictions and Training . . . 63
4.4 Weight Update and Gradient Descent . . . 64
4.5 Overfitting . . . 67
4.5.1 Dropout . . . 67
4.5.2 Regularization . . . 68
4.6 Conclusions . . . 69

## 4.1 Differences with Artificial Neural Networks

Machine learning development can also be done using traditional artificial neural networks (ANN) instead of CNNs. In an ANN, each neuron is connected to all the following neurons via a weight and is composed of a bias as illustrated by (4.1) and Figure 4.1.

$$x_i^l = \sum_{k=1}^{N^l} w_{k,i} x_k^{l-1} + b_i^l \tag{4.1}$$

Where:

- $x_i^l$ is the i-th neuron at the l-th layer
- $N^l$ is the number of neurons at the (l-1)-th layer
- $w_{k,i}$ is the weight of the k-th neuron of the previous layer to the i-th neuron of the current one
- $x_k^{l-1}$ is value of the k-th neuron of the previous layer
- $b_i^l$ is the bias value of the i-th neuron at the l-th layer

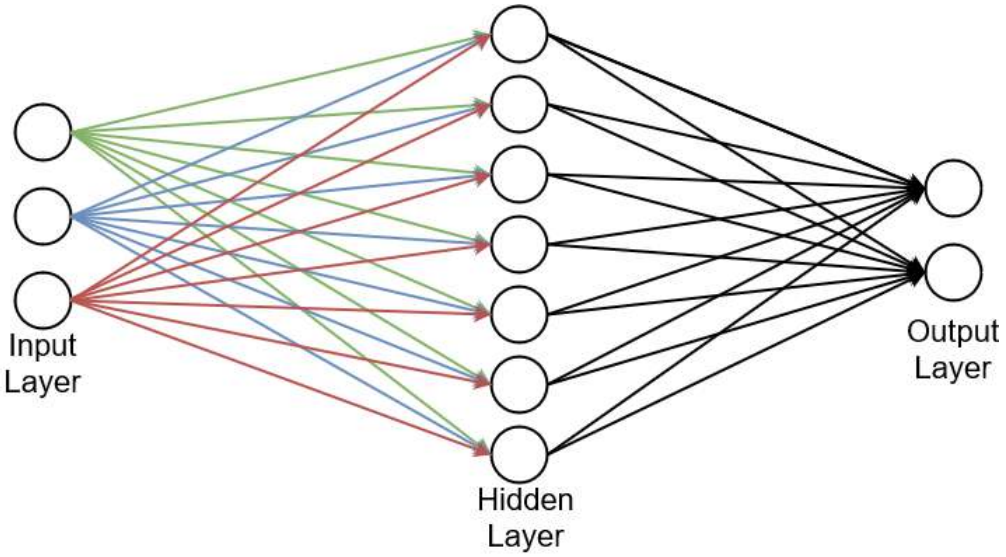


Figure 4.1: Artificial neural network architecture example.

It can be seen that ANNs use many weight connections between neurons. Hence, when the input layer size increases, the number of weights increases as well. The weights and biases are the quantities that are being trained in an ANN to achieve its purpose. When images are used in ANNs, they need to be flattened to ressemble the architecture of Figure 4.1. Images are 2D or 3D inputs, so each pixel is assimilated as an input neuron making the number of parameters to train drastically increase. Another downside of ANN classification is that features need to be identified before training for the model to know which features to rely on and detect.

When it comes to image classification, artificial neural networks were shown to not be very efficient due to their structure [Krizhevsky et al., 2012]. On the other hand, CNNs identify the features deemed notable thanks to filters and then classify images [Alzubaidi et al., 2021]. As in this PhD, images were used to detect multipath and the features indicating what is multipath are not assumed in this work; CNNs were a better fit.

Other types of neural networks exist such as recurrent neural networks (RNN) and generative adversarial network (GAN). RNNs function by using the previous model predictions as memory for the current prediction. It is vastly used in speech recognition or in other applications where there is high correlation between the successive inputs. GANs operate as two models competing, where one's gain is the other's loss. GANs are mostly used in unsupervised learning meaning that the data is not labeled. In this thesis, the input data was labeled as supervised learning methods were used.

## 4.2 Layers

A convolutional neural network is composed of several layers - hidden and visible - making up its architecture. A generic CNN architecture is given in Figure 4.2. These layers are described in the subsections below.

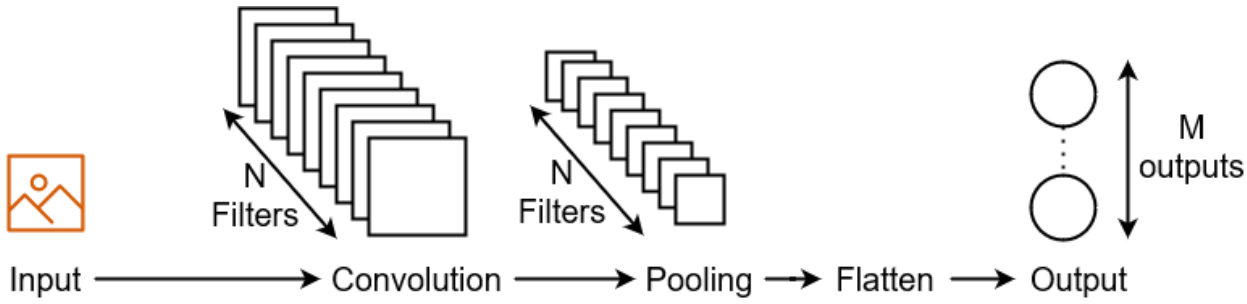


Figure 4.2: CNN Architecture Example.

### 4.2.1 Convolutional Layer

As indicated by the name, convolutional neural networks rely on the convolution operation. The input, for example an image, is a 2D matrix (or 3D if in color). This input is then convoluted with multiple filters, all the same size but with different values. Filters can be thought of as feature detection matrices. In computer vision, kernels can be used to detect edges or sharpen the images [Lampert, 2009]. CNNs are based on the same principle but keep updating the values of their filters to detect the most prominent features of the input images. The number of filters used in a convolutional layer will determine the number of 2D outputs. These 2D outputs are called feature maps as each one was passed through a different filter, thus detecting different features.

The convolution is a time consuming process increasing with the size and desired number of feature maps. A stride can be implemented to skip elements on which to perform the convolution on. The mathematical expression of a convolutional layer is given below

[Carneiro et al., 2017].

$$A_o^l = \sum_{k=1}^{N} W_{o,k}^l * A_k^{l-1} + b_o^l \tag{4.2}$$

Where:

- $k$ is the number of channels in the image (1 for Black and White and 3 for RGB)
- $o$ indicates the number of channels in the output image
- $*$ is the convolution product
- $A_k^{l-1}$ is the image input
- $W_{o,k}^l$ is the filter with which the convolution is performed
- $b_o^l$ is an additional bias

This convolution is illustrated on Figure 4.3 by taking a $5 \times 5$ image and convoluting it with a $3 \times 3$ matrix with no stride yielding a matrix of $3 \times 3$. This process is repeated for the specified number of filters in the layer giving N feature maps.

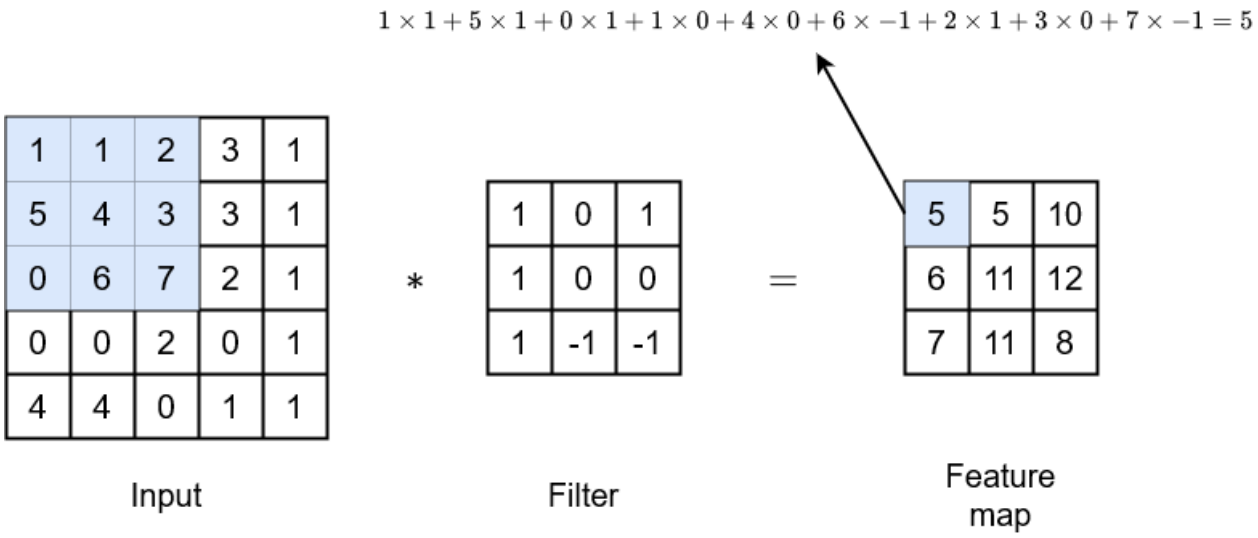


Figure 4.3: Convolution illustration.

Once the feature maps are generated from the convolutional layer, an activation function is used to transform the values of the maps. Numerous activation functions have been proposed, such as the rectified linear unit (ReLU) [Agarap, 2018]. These functions are generally non-linear to adapt to a wider variety of data. Indeed, a combination of linear functions is a linear function itself as shown in (4.3), so the network would not need several layers.

$$f(\lambda x + y) \neq \lambda f(x) + f(y) \; \forall \; \lambda, x, y \in \mathbb{R} \tag{4.3}$$

On top of being non-linear, activation functions need to be differentiable. These functions may not be continuously differentiable like the ReLU, but can still be used for the backpropagation of weights using the gradient descent described in 4.4.

### 4.2.2 Pooling Layer

Pooling layers are used to down-sample feature maps in order to reduce the computational load, created by the convolutional layers. Pooling is most often done as max pooling or average pooling. The pooling layer uses a filter of specified size ($N \times N$) that slides over the whole feature map with a given step called stride. Max pooling selects the highest value of the feature map. Average pooling takes the average value of the map. Max pooling brings out the most prominent feature of the pooling region for a given image while average pooling keeps most of the information. One does not necessarily perform better than another and their uses depend on the input data [Zafar et al., 2022]. The max and average pooling formulas are given in (4.4) and (4.5) respectively.

$$A_o(s,t)^l = max(A_o^{l-1}(\alpha_l s + i, \beta_l t + j)) \ \forall i \in [1, N_1] \text{ and } \forall j \in [1, N_2] \tag{4.4}$$

$$A_o(s,t)^l = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} A_o^{l-1}(\alpha_m s + i, \beta_m t + j) \tag{4.5}$$

Where:

- $N_1$ is the size of the pooling window in the X axis
- $N_2$ is the size of the pooling window in the Y axis
- $\alpha_m$ is the stride of the pooling window in the X axis
- $\beta_m$ is the stride of the pooling window in the Y axis

The two pooling methods are illustrated in Figure 4.4.

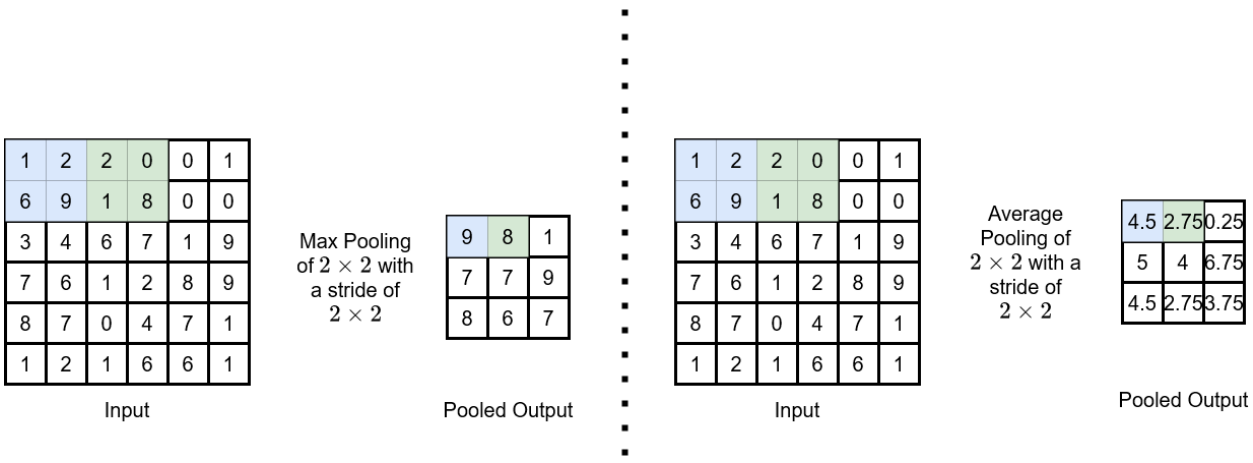


Figure 4.4: Max Pooling and Average Pooling example.

### 4.2.3 Output Layer

Convolutional and pooling layers output feature maps that have several dimensions; yet, it is not possible to classify them directly. So, the final layer in a CNN is flattened to a 1-D layer. This layer is composed of one or more neurons to classify the data. The number of neurons will influence the activation function used. For binary classification, the sigmoid

function, given in (4.6) should be used as it returns a probability between 0 and 1 for the data to belong in either class 0 or class 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.6}$$

On the other hand, for multi-class classification, the softmax function should be implemented and can be given as:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{4.7}$$

Where:

- $N$ is the number of output neurons

## 4.3 Predictions and Training

To decide how many convolutional, poolings layers, and which activation functions to use, there are several factors to take into account. One factor is the complexity of the problem. Indeed, if the problem at hand is very complex, then there should be multiple convolutional layers. One downside of adding convolutional layers is that the network will take much more computational time due to the increased number of convolutions. Another downside is that by increasing the number of convolutional layers (and any layer with trainable weights in them) will lead to more data needed to train the network.

Pooling layers are not mandatory in a CNN but recommended as they decrease the size of feature maps without losing valuable information for the network. The choice of activation function is dependent on the problem at hand as well as performance. The goal is to choose a function that limits the vanishing gradient problem presented in section 4.4.

Once a first network has been built using estimates of what could be the best network for the given problem, it needs to be trained and tested. To evaluate a network based on supervised learning, this implies that the input data has been pre-labeled without the help the network. When the data is fed to the built network, it will yield predictions. The prediction is defined as the output of the neural network. The difference between the prediction and the labeled truth is the error. The goal of the network is to minimize this error using a loss function, which is presented in section in section 4.4. This recurring process is called the training of the network.

The network is usually trained on most of the available data (∼70-80%) multiple times. This means that the training data will be presented several times to the network to keep updating its knowledge as it gained knowledge about the problem from the previous time. The number of time the data is presented for training is called the number of epochs. Once the network has been trained, it is then referred to as a model as its characteristics and weights do not change anymore. The model is then tested on the remaining data (∼20-30%) to evaluate its performance on unseen data. Sometimes validation data can also be used during training (∼10-20%) to optimize hyperparameters.

Once the performance of the model is characterized, it may occur that it performs badly or is affected by overfitting, detailed in section 4.5. There may be a lack of data for the correct training of the CNN and/or a bad network architecture. To fix bad network architectures, networks can be built gradually. This means that a first model can be built using one convolutional layer with one pooling layer and output layer before moving on to more complex architectures if need be.

## 4.4 Weight Update and Gradient Descent

Loss functions are primordial in all neural networks as they yield the error between the prediction and the truth. During training, this function is minimized. Many loss functions exist, and some are better suited according to the application. For regression, the mean squared error is popular while the categorical cross entropy is used for multi-classification [Zhang and Sabuncu, 2018] and the binary cross entropy for two classes. The binary cross entropy function is written as:

$$J(\hat{y}) = -\frac{1}{2}\sum_{i=0}^{1} y_i \log(\hat{y_i}) \tag{4.8}$$

Where:

- $J$ is the loss function
- $y$ is the truth of 0 or 1 corresponding to the class
- $\hat{y}$ is the probability corresponding to the class

$\hat{y}$ is the resulting probability from the CNN; therefore, this value is obtained by passing through all the layers of the network. (4.8) can be rewritten as:

$$J(\hat{y}) = -\frac{1}{2}\sum_{i=0}^{2} y_i \log(f^l(W^l f^{l-1}(W^{l-1} \dots)) \tag{4.9}$$

Where:

- $f^l$ is the activation function at layer $l$
- $W^l$ are the weights at layer $l$

The goal is to minimize the loss function, to do so the gradient descent algorithm is applied. The gradient descent is an iterative algorithm consisting in computing the gradient of a given function to update its parameters to move closer towards its minimum. In the case of neural networks, the predictions are updated to move the loss function towards its minimum. Its expression can be given as [Ruder, 2017]:

$$\hat{y}_{n+1} = \hat{y}_n - \eta \nabla_{\hat{y}_n} J(\hat{y}) \tag{4.10}$$

Where:

- $\hat{y}_n$ is the probability prediction at iteration n
- $\eta$ is the learning rate
- $\nabla$ is the gradient operator

The learning rate is an important parameter as it influences by how much the gradient descent updates the prediction values. The gradient descent is illustrated on Figure 4.5 for a simple quadratic loss function with two different learning rates.

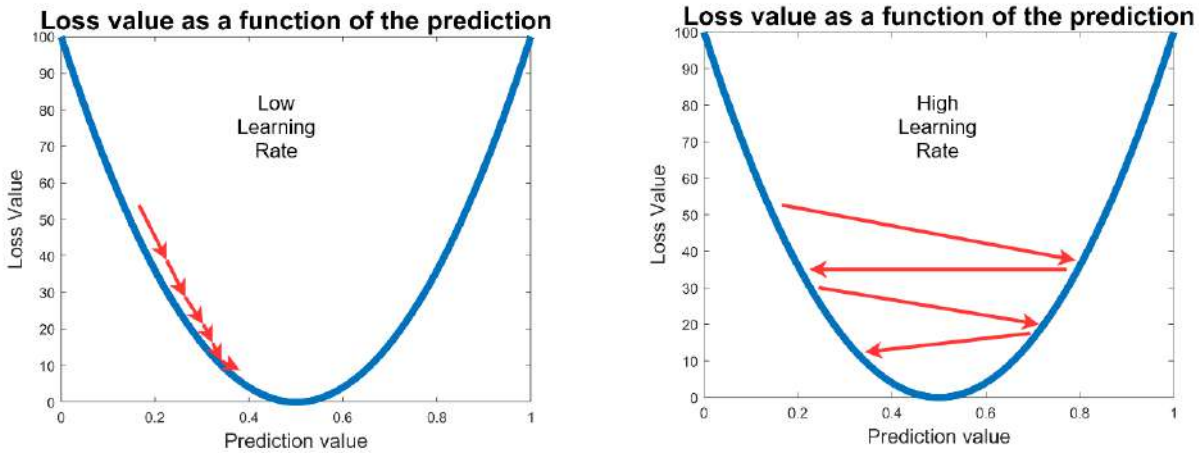


Figure 4.5: Gradient Descent technique for low and high learning rates.

It can occur that due to a low value of the gradient, the predictions are barely updated and the loss function does not reach the actual minimum. This is known as the vanishing gradient problem. On the other hand, the predictions can be updated by large values which causes the gradient to increase. This is known as the exploding gradient. To reduce the impact of this phenomenon, some methods have been proposed such as adding momentum to the gradient descent. This results in a weighted update equal to [Ruder, 2017]:

$$\hat{y}_{n+1} = \hat{y}_n - \eta \nabla_{\hat{y}_n} J(\hat{y}) + \alpha \Delta \hat{y}_n \tag{4.11}$$

Where:

- $\alpha$ is the exponential decay factor between 0 and 1
- $\Delta \hat{y}_n = \hat{y}_n - \hat{y}_{n-1}$ is the difference in the previous estimation

The gradient descent is computationally demanding. For, the stochastic gradient descent (SGD) computes the gradient descent by randomly selecting a training sample of the dataset and performs the gradient descent and updates the weights of the network directly. To reduce the computational load, variants of the SGD are implemented in machine learning algorithms. A common algorithm is the mini-batch gradient descent. It does not compute the gradient descent on the whole dataset but splits the dataset into random batches of chosen size and performs the gradient descent algorithm on these batches. The weights are updated after each batch is processed by the gradient descent. This procedure is repeated for all batches to reduce the overall loss function. Having a large batch size causes the model to be slow and will have poor generalization. On the other hand, a small batch size will

have faster convergence but could reach sub optimal performance [Masters and Luschi, 2018].

Another method to improve the effectiveness of the stochastic gradient descent is by using the Adam: adaptive moment estimation optimizer presented in [Kingma and Ba, 2017]. This optimizer adapts the learning rate of the gradient descent for a given training sample based on the running averages of the gradient and on the second moment of the gradient. This leads to a more accurate estimation of the minimum. Indeed, adapting the learning rate helps when some data parameters are more sensitive than others. The two running averages are given in (4.12) and (4.13).

$$\begin{aligned} m_{n+1} &= \beta_1 m_n + (1-\beta_1)\nabla_{\hat{y}_n} J(\hat{y}) \\ \hat{m} &= \frac{m_{n+1}}{1-\beta_1} \end{aligned} \tag{4.12}$$

$$\begin{aligned} v_{n+1} &= \beta_2 v_n + (1-\beta_2)\Big(\nabla_{\hat{y}_n} J(\hat{y})\Big)^2 \\ \hat{v} &= \frac{v_{n+1}}{1-\beta_2} \end{aligned} \tag{4.13}$$

Where:

- $m_{n+1}$ is the running average of the gradient
- $v_{n+1}$ is the running average of the second moment of the gradient
- $\hat{m}$ is the exponential average of the gradient
- $\hat{v}$ is the exponential average of the second moment of the gradient
- $\beta_1$ is the forgetting factor for the gradient
- $\beta_2$ is the forgetting factor for the second moment of the gradient

The prediction update is then computed as [Kingma and Ba, 2017]:

$$\hat{y}_{n+1} = \hat{y}_n - \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{4.14}$$

Where:

- $\epsilon$ is a small scalar to avoid division by 0

Finally, to update the weights of each layer, the chosen gradient descent method can be applied but on each layer. Each weight computation depends on the following weight value. This is called backpropagation and is achievable thanks to the chain rule. The weight update for each layer can be defined similarly to (4.10).

$$\hat{w}^j_{n+1} = \hat{w}^l_n - \eta\Big(f'^l \cdot \nabla_{\hat{w}^{l+1}_{n+1}} J(\hat{y})\Big) \tag{4.15}$$

Where:

- $\hat{w}^j_{n+1}$ is the weight value for the iteration $n+1$ on the layer $j$

## 4.5 Overfitting

A common problem in ML is overfitting. Overfitting is when a model works well for the training data but performs poorly when fed new data. This is due to the model adapting too much to the training data. The model then creates a function that minimizes the loss function of the training data very well. However, the validation and/or test loss functions can not be minimized to the same extent. The most frequent reasons of overfitting are:

- The trained data is not fully representative of the potential input data
- The network is too complex for the problem at hand
- The network has been trained for too many iterations
- Data leakage, which is when the network has access to information during training that it will not have when predicting.

A good rule of thumb is to compare the loss function values of the training and validation dataset to check overfitting. This section details two widely implemented methods that reduce the risk of overfitting.

### 4.5.1 Dropout

Dropout deactivates a neuron during the training phase with a given probability (chosen in the implementation) to reduce reliance on certain connections. Dropout does not occur when the model predicts outputs after the training phase. As shown by [Srivastava et al., 2014], this is an effective way to reduce the likelihood of overfitting. However, introducing a low dropout will not reduce overfitting nor increase the performance while a high dropout will lead to longer convergence.

This method is illustrated in Figure 4.6 for a CNN, after being flattened resembling the architecture of an ANN, with a hidden layer of neurons.

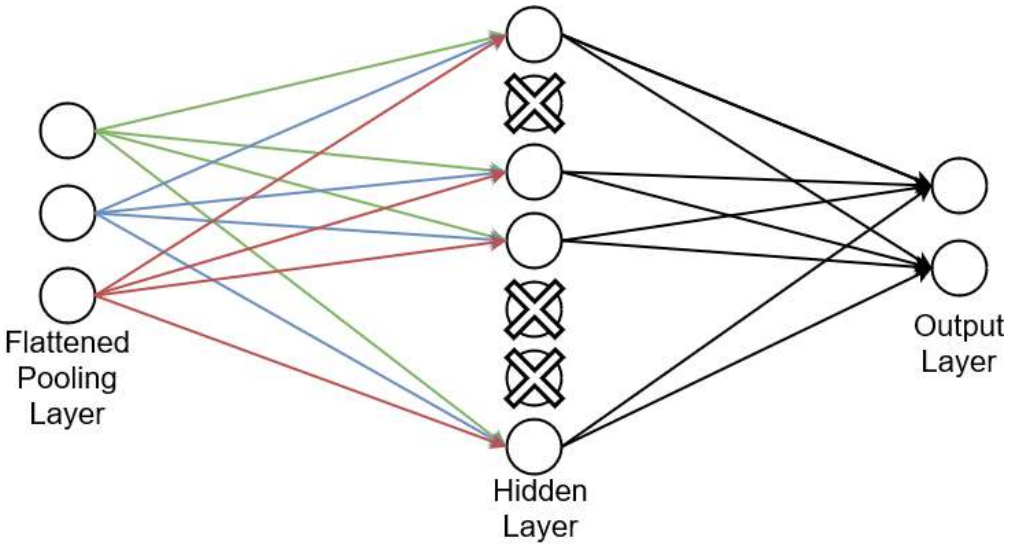


Figure 4.6: Dropout example for a flattened CNN.

### 4.5.2 Regularization

Another method to limit overfitting is by using L1 and/or L2 regularization to the weights [Ying, 2019]. On the weights, L1 regularization, also known as the LASSO regression, can be defined as:

$$L1_{reg}(\theta) = J(\theta) + \lambda \sum_{j=1}^{N} |w_j| \tag{4.16}$$

Where:

- $J(\theta)$ is the existing cost function
- $\lambda$ is the regularization factor
- $w_j$ are the weights of the network
- $N$ is the number of weights in the network
- $L1_{reg}(\theta)$ is the new cost function with L1 regularization

And L2 regularization, also known as RIDGE regression, can be defined as:

$$L2_{reg}(\theta) = J(\theta) + \lambda \sum_{j=1}^{N} w_j^2 \tag{4.17}$$

L1 regularization drives the number of used features to be small causing the model to be sparse. On the other hand, L2 regression enables a faster learning of the network [Ng, 2004]. Hence, a combination of both, called the elastic net presented in [Zou and Hastie, 2005], not only reduces overfitting but also takes advantage of both regression methods. The new loss function becomes:

$$E_{NET}(\theta) = J(\theta) + \lambda_{L1} \sum_{j=1}^{N} |w_j| + \lambda_{L2} \sum_{j=1}^{p} w_j^2 \tag{4.18}$$

Where:

- $J(\theta)$ is the existing cost function
- $\theta$ is the parameter to minimize
- $\lambda$ is the regularization factor (L1 or L2)
- $E_{NET}(\theta)$ is the new cost function with L1 and L2 regularization

Reducing overfitting via regularization comes at the expense of adding a bias in the estimated parameters. Therefore, the choice in magnitude of $\lambda$ depends on the tolerated bias for the dataset. In machine learning, this hyper parameter, $\lambda$, is often empirically determined by testing different values and analyzing which one performs best on a given network with the other hyper parameters of the model constant.

## 4.6 Conclusions

In this chapter, the architecture of a CNN was detailed as well as the different implementation choices to make when designing a network. Indeed, for the model to perform as intended, many factors come into play such as the activation functions, number of layers, loss function etc. All these implementation choices are trade-offs to the computational complexity, the time, and amount of data required to train the network with the desired performance. This chapter also presented two methods to limit overfitting of the model to its training data.

A focus was placed on using CNNs for classifying images in this chapter. Indeed, in this thesis, CNNs were used to classify whether the correlator outputs were affected by multipath or not. The correlator outputs were dependent on time and delay and were in the form of 2D images. The CNNs implemented in this PhD are detailed in Chapters 6 and 7.

CHAPTER 5

# Proposed TDCP based EKF

The positioning algorithm to be implemented in the eHermes needs to be as lightweight as possible and also be accurate in challenging conditions. Therefore, the LSE algorithm or its variants are not sufficient. An EKF was chosen as the positioning algorithm for the receiver. However, as already mentioned, to reap the benefit of the precise carrier phase measurement, the ambiguity has to be dealt with, for example being estimated in the EKF states as shown in Chapter 3.

By differencing two consecutive carrier phase measurements, the accuracy of these observables can be used to improve the positioning accuracy without having additional states which speeds up the computation time of the filter. The downside of only using TDCPs is that this comes at the expense of absolute positioning. This chapter presents an EKF that still uses code pseudoranges, Doppler measurements but adopts time differenced carrier phase measurements instead of raw carrier phase measurements as in 3.3.2 which led to the publication [Guillard et al., 2022]. The formation of these TDCP measurements will be presented as well as the changes on the EKF models to take into account these observables. Since cycle slips still affect TDCPs, two methods to detect them will be detailed. Then, the datasets on which the filter was tested will be presented along with the corresponding results.

## Chapter Contents


5.1 Formation of TDCP Measurements . . . 71
5.2 State of the Art EKF/PPP Algorithms . . . 72
5.3 Adaptation of the EKF . . . 73
5.3.1 Measurement correlation . . . 74
5.3.2 Adapted EKF Equations . . . 74
5.3.3 Error State vector . . . 75
5.3.4 State Transition Model and State Process Noise Matrix . . . 76
5.3.5 Observation Model . . . 76
5.4 Cycle Slip Detection . . . 77
5.4.1 EKF based Cycle Slip Rejection . . . 78
5.4.2 Wavelength based Cycle Slip Rejection . . . 78
5.5 Method and Scenarios . . . 79
5.5.1 Synchronization of sensors . . . 80
5.5.2 Reference Trajectory . . . 81

5.5.3 Differences between Datasets . . . . . 82
5.6 Results . . . . . 83
5.6.1 IGS Dataset . . . . . 84
5.6.2 SPAN IMU Datasets . . . . . 86
5.6.3 MEMS IMU Datasets . . . . . 92
5.7 Conclusions . . . . . 99

## 5.1 Formation of TDCP Measurements

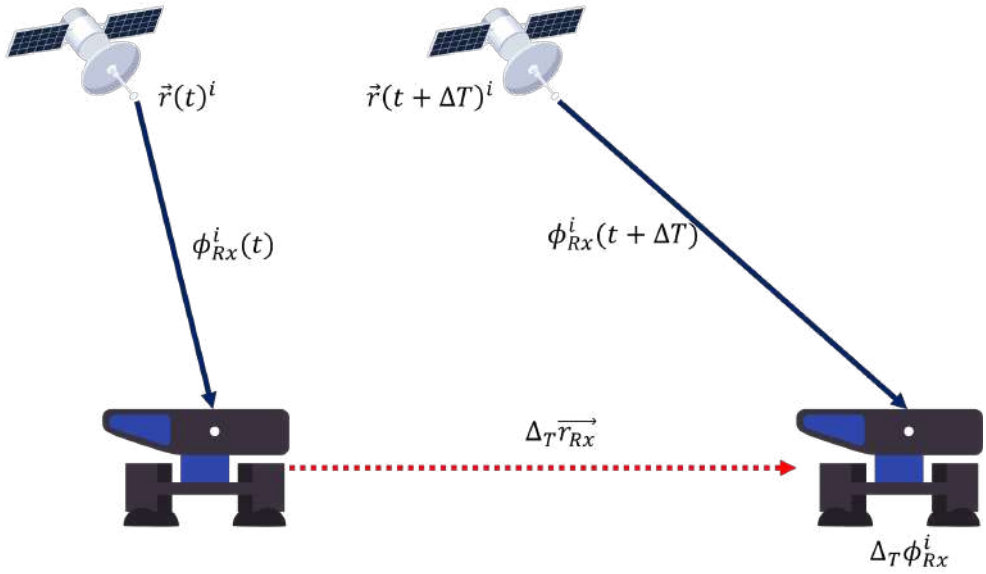


Figure 5.1: Formation of TDCP Measurements.

The time differenced carrier phase differences two consecutive carrier phase measurements to effectively eliminate the ambiguity term in the absence of cycle slips as shown in Figure 5.1. Thus, the TDCP still benefits from the precision of the carrier phase measurement with a lower implementation complexity. Indeed, when using TDCPs over carrier phase measurements in a hybridized GNSS/INS EKF, a lower state vector size is needed since the ambiguities are no longer estimated. The TDCP measurement can be obtained as:

$$\Delta_T\phi^i_{Rx} = \phi^i_{Rx}(t+\Delta T) - \phi^i_{Rx}(t) \tag{5.1}$$

$$\begin{aligned}\Delta_T\phi^i_{Rx} = (\vec{r}^{\,i}(t+\Delta T) - \vec{r}_{Rx}(t+\Delta T)) \cdot \vec{e_{Rx}}^{\,i}(t+\Delta T) - (\vec{r}^{\,i}(t) - \vec{r}_{Rx}(t)) \cdot \vec{e_{Rx}}^{\,i}(t) \\ +\Delta_T(cdt_{Rx}) - \Delta_T(cdt^i) + \Delta_T T^i_{Rx} + \Delta_T I^i_{Rx} + \Delta_T\delta^i_{\phi,Rx} + \Delta_T\varepsilon^i_{\phi,Rx}\end{aligned} \tag{5.2}$$

As long as $\Delta_T$ is not too big, the tropospheric, ionospheric, and multipath temporal evolutions are considered to be negligible [Park and Kee, 2010]. These error sources are included in the TDCP residual noise term ($\Delta_T E^i_{Rx}$).

$$\begin{aligned}\Delta_T\phi^i_{Rx} = \vec{r}^{\,i}(t+\Delta_T) \cdot \vec{e_{Rx}}^{\,i}(t+\Delta_T) - \vec{r}^{\,i}(t) \cdot \vec{e_{Rx}}^{\,i}(t) - \Delta_T\vec{r}_{Rx} \cdot \vec{e_{Rx}}^{\,i}(t+\Delta_T) - \\ \vec{r}_{Rx}(t)\Delta_T \cdot \vec{e_{Rx}}^{\,i} + \Delta_T(cdt_{Rx}) - \Delta_T(cdt^i) + \Delta_T E^i_{Rx}\end{aligned} \tag{5.3}$$

Where $\Delta_T E^i_{Rx} = \Delta_T T^i_{Rx} + \Delta_T I^i_{Rx} + \Delta_T \delta^i_{\phi,Rx} + \Delta_T \varepsilon^i_{\phi,Rx}$ represents the temporal change of GNSS errors as well as the noise.

## 5.2 State of the Art EKF/PPP Algorithms

Most PPP based EKF solutions use raw carrier phase measurements and do not difference them such as [Li, 2019], [Gao et al., 2017], and [Vana et al., 2019].

[Li, 2019] proposes a dual frequency PPP GNSS only EKF that estimates carrier phase ambiguities. Their algorithm uses GPS and GLONASS measurements; the latter are based on frequency-division multiple access. Thus, they also estimate the inter-frequency biases between GLONASS measurements as states. In the EKF states, they account for cycle slips by adding additional states in the EKF. They try to detect cycle slips on each frequency and satellite by using the geometry free combination $\phi_{GF} = \phi_1 - \phi_2$ and computing a threshold based on the cycle slip state and covariance values.

[Gao et al., 2017] designed a tightly coupled solution GNSS/INS EKF algorithm. They used multi constellation data and first order ionosphere free combinations on the code pseudoranges, carrier phase and Doppler measurements. They estimate the same states as in (3.60) plus the scale factor of the accelerometers and gyroscopes. They account for the inter-system bias of constellations and inter-frequency biases separately in order to speed up the convergence of the filter.

[Vana et al., 2019] also processes ionosphere free code and carrier phase measurements using a low-cost MEMS IMU and GNSS receiver. They estimate the same states as presented in (3.60).

On the other hand, some authors have already discussed the potential benefit of using TDCPs instead of raw carrier phase measurements such as [Kim et al., 2020], [Soon et al., 2008], and [Kai et al., 2021].

[Kim et al., 2020] used uncombined TDCPs on multiple constellations and frequencies. The TDCPs used in this paper were also differenced with a chosen reference satellite to remove the receiver clock drift. Their algorithm tightly coupled the single differenced TDCPs with a high quality INS without other measurements. Therefore, their position was externally initialized, and the proposed algorithm worked incrementally. They excluded cycle slips with a modified version of the method described in 5.4.2. As they removed the effect of the receiver clock drift, they estimated the receiver position, velocity, attitude, and IMU biases.

[Soon et al., 2008] used TDCPs on GPS L1 C/A as a replacement of carrier phase measurements. However, his method did not include a method to detect cycle slips. This algorithm uses a tight coupling between the INS and the GNSS receiver. Only the TDCPs are used as GNSS measurements. The position of the receiver is first initialized and then iteratively updated with the proposed EKF.

[Kai et al., 2021] implements a tightly coupled EKF with INS/TDCP measurements similarly

to [Kim et al., 2020]. Indeed, they compute single differenced TDCPs, on GPS L1 C/A and L2, by subtracting each TDCP with the TDCP of a chosen reference satellite and estimate the same states. They use this EKF for incremental positioning with an initialized position from a receiver. They deal with cycle slips based on the innovation magnitude and re-adapt the weights of the EKF measurement matrix.

The algorithm proposed in this chapter compares the use of uncombined and first order iono-free TDCPs which has not been investigated in the mentioned papers. Furthermore, two methods of cycle slip detection are proposed and analyzed according to different scenarios. It is not mentioned in the literature whether consecutive TDCPs are used which would result in a correlation of measurements and should be accounted for. In this implementation, TDCPs are used every two epochs to avoid this problem and keep the complexity of the filter as low as possible. Furthermore, TDCPs are rarely used alongside pseudoranges and Doppler measurements for absolute positioning as most implementations use them as incremental positioning tools from a predetermined position such as [Kim et al., 2020] and [Soon et al., 2008]. As this algorithm is designed to work in real time on a GNSS receiver, the computational gain of using TDCPs is also discussed.

## 5.3 Adaptation of the EKF

The overall architecture of the presented TDCP based EKF is illustrated in Figure 5.2.

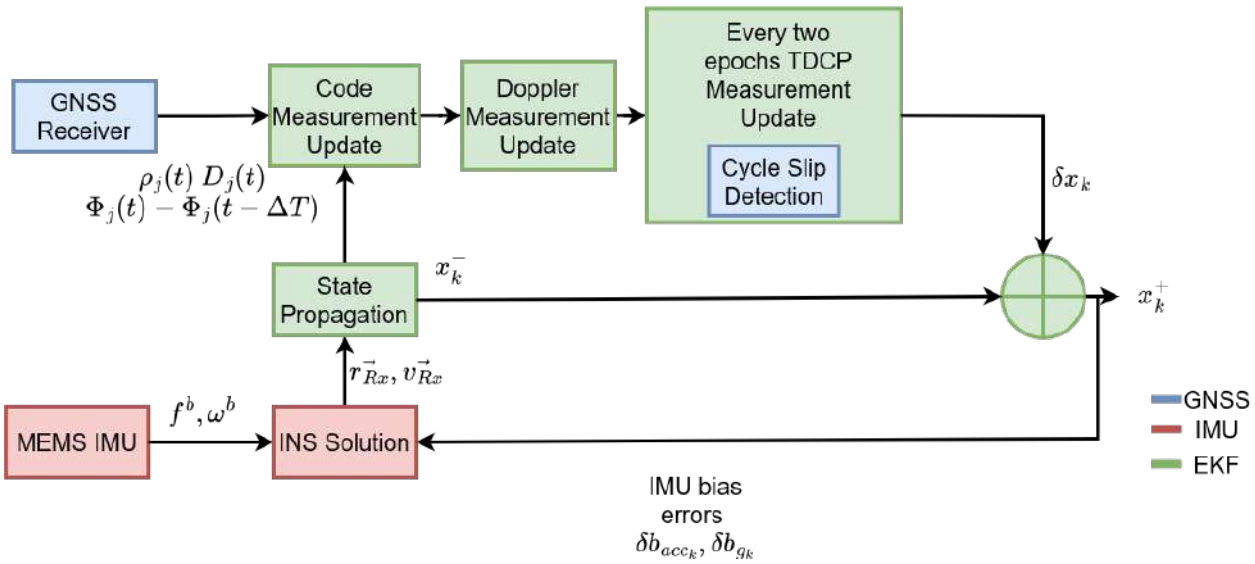


Figure 5.2: EKF Filter Architecture for one epoch.

As it will be shown in section 5.3.2, the equations for the measurement update are different for TDCPs than for code pseudoranges and Doppler measurements. Hence, the extended Kalman filter was designed so that the measurements were treated separately. To further reduce the computational burden, the observables were processed sequentially as done in [Welch and Bishop, 1997]. Indeed, finding the inverse of a MxM (with $M \in \mathbb{N} \setminus \{0, 1\}$) matrix is never computed except for the state transition matrix, $\Phi_{k-1}^{-1}$. The other benefit is that the filter can have two sets of measurement update equations depending on the

observable treated at a given instant: one for TDCPs and the other set for code and Doppler measurements.

### 5.3.1 Measurement correlation

As shown in (5.1), TDCPs are constructed as the difference of two consecutive carrier phase measurements; therefore, two consecutive TDCPs use the same carrier phase measurement as highlighted in (5.4).

$$\begin{aligned} \Delta_T \phi^i_{Rx}(t) &= \boldsymbol{\phi^i_{Rx}(t)} - \phi^i_{Rx}(t - \Delta T) \\ \Delta_T \phi^i_{Rx}(t + \Delta T) &= \phi^i_{Rx}(t + \Delta T) - \boldsymbol{\phi^i_{Rx}(t)} \end{aligned} \tag{5.4}$$

Hence, two consecutive TDCPs are mathematically correlated. [Teunissen, 1996] proposed to decorrelate double difference carrier phase measurements by applying transformation matrices. This method can be applied for TDCP measurements as well and is shown in (5.5):

$$z = Z^T y \tag{5.5}$$

Where:

- $z$ is the transformed measurement
- $Z$ is the transformed matrix
- $y$ is the original measurement

Similarly to double difference carrier phase measurements, the transformation matrix to decorrelate TDCPs must have the three following properties with the demonstration shown in [Teunissen, 1996]:

1. All elements of $Z$ are integers
2. All elements of $Z^{-1}$ are integers
3. The determinant of $Z$ is equal to $\pm 1$

However this transformation also changes the TDCP measurement input to the EKF. Thus, the observation matrix would have to take into account the transformation of the decorrelated TDCP measurement.

In the proposed implementation, another solution was chosen: performing the TDCP measurement update once every two epochs. Although, the potential benefit of using TDCP every epoch is lost, this ensures that measurements remain uncorrelated as well as a simpler implementation, and saves computational power.

### 5.3.2 Adapted EKF Equations

In this thesis, it is assumed that the state process noise and measurement noise are not correlated in time, yielding:

$$E[w_{k-1} v_k^T] = 0 \tag{5.6}$$

The incorporation of the TDCP measurements invalidates this assumption as these observables are obtained from positions of different time epochs (current and past) yielding that $w_{k-1}$ and $v_k$ are correlated [Brown, 2012]. As shown in (3.3), the true state is given as:

$$x_k^- = \Phi_{k-1} x_{k-1}^+ + w_{k-1} \tag{5.7}$$

Rewriting (5.7), it yields:

$$x_{k-1}^+ = \Phi_{k-1}^{-1} x_k^- - \Phi_{k-1}^{-1} w_{k-1} \tag{5.8}$$

The measurement model may also be rewritten as a function of the current and previous states:

$$z_k = H_k x_k^- + J_k x_{k-1}^+ + v_k \tag{5.9}$$

Where:

- $J_k$ is a measurement matrix equal to $J_k = -H_k$

By plugging (5.8) into (5.9), the new measurement model is given as:

$$z_k^{'} = (H_k + J_k \Phi_{k-1}^{-1}) \hat{x}_k^- + (-J_k \Phi_{k-1}^{-1} w_{k-1} + v_k) \tag{5.10}$$

Since $R_k$ is computed as the covariance of $v_k$, the updated measurement noise matrix is given by:

$$R_k^{'} = E\left[(-J_k \Phi_{k-1}^{-1} w_{k-1} + v_k)(-J_k \Phi_{k-1}^{-1} w_{k-1} + v_k)^T\right] \tag{5.11}$$

After simplification, this equates to:

$$R_k^{'} = R_k + J_k \Phi_{k-1}^{-1} Q_{k-1} {\Phi_{k-1}^{-1}}^T J_k^T \tag{5.12}$$

This also has repercussions on the Kalman Gain and state covariance matrix update steps, with the new formulas given in (5.13) and (5.14). This is detailed in Appendix A.

$$K_k^{'} = (P_k^- H_k^T + \Phi_{k-1} P_{k-1} J_k^T)(H_k P_k^- H_k^T + R_k + J_k P_{k-1} \Phi_{k-1}^T H_k^T + H_k \Phi_{k-1} P_{k-1} J_k^T + J_k P_{k-1} J_k^T)^{-1} \tag{5.13}$$

$$P_k^+ = P_k^- - K_k L_k K_k^T \tag{5.14}$$

Where:

- $L_k = (H_k P_k^- H_k^T + R_k + J_k P_{k-1} \Phi_{k-1}^T H_k^T + H_k \Phi_{k-1} P_{k-1} J_k^T + J_k P_{k-1} J_k^T)$

### 5.3.3 Error State vector

The size of the state vector of the hybridized solution is smaller compared to the one presented in section 3.3.2 as it does not contain the error of the estimated ambiguities. The state vector is then equal to:

$$\delta x_k = [\delta r_k^e \;\; \delta v_k^e \;\; \delta dt_{Rx_k} \;\; \delta \dot{dt}_{Rx_k} \;\; \delta dt_{GPS-GAL,k} \;\; \delta ztd_k \;\; \delta \psi_k^e \;\; \delta b_{acc_k} \;\; \delta b_{gyro_k}] \tag{5.15}$$

### 5.3.4 State Transition Model and State Process Noise Matrix

With respect to the state transition matrix of the hybridized EKF with carrier phase measurements, , the ambiguity term is no longer present and the state transition matrix, $\Phi_{k-1}$ is equal to:

$$\Phi_{k-1} = \begin{bmatrix} I_3 & \Delta_T I_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_3 & 0_3 & 0_3 \\ 0_3 & 1-2\Omega_{ie}\Delta_T & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & \Delta_T[-R_b^e f^b\times] & -\Delta_T R_b^e & 0_3 \\ 0_{1\times3} & 0_{1\times3} & 1 & \Delta_T & 0 & 0 & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 1 & 0 & 0 & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 & 1 & 0 & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 & 0 & 1 & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 1+\Delta_T\Omega_{ie} & 0_3 & -\Delta_T R_b^e \frac{\pi}{180} \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_3 & 0_3 & I_3 \end{bmatrix} \tag{5.16}$$

Identically, the state process noise is as presented in (3.65) without the ambiguity state noise. Its expression is:

$$Q_{k-1} = diag(\begin{bmatrix} 0_3 \\ Q_{vel} \\ 0 \\ Q_{rxDrift} \\ Q_{interGnss} \\ Q_{Tropo} \\ Q_{att} \\ Q_{acc} \\ Q_{gyro} \end{bmatrix}) \tag{5.17}$$

### 5.3.5 Observation Model

The observations are still assumed independent even with the addition of TDCP as carrier phase measurements are no longer used. Despite the equation change for the observation model for TDCPs, its observation matrix and covariance matrix are still defined classically.

$$H_{k,\rho} = \begin{bmatrix} -e_{\vec{INS}}{}^1 & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & -e_{\vec{INS}}{}^1 R_b^e(-\vec{p}^b_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -e_{\vec{INS}}{}^N & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & -e_{\vec{INS}}{}^N R_b^e(-\vec{p}^b_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \end{bmatrix} \tag{5.18}$$

$$H_{k,Doppler} = \begin{bmatrix} 0_{1\times3} & -e_{\vec{INS}}{}^1 & 0 & 1 & 0 & 0 & -e_{\vec{INS}}{}^1 R_b^e(\vec{v}^b_{INS}\times) & 0_{1\times3} & -e_{\vec{INS}}{}^1 R_b^e(\vec{p}^b_{Rx}\times)\frac{\pi}{180} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_{1\times3} & -e_{\vec{INS}}{}^N & 0 & 1 & 0 & 0 & -e_{\vec{INS}}{}^N R_b^e(\vec{v}^b_{INS}\times) & 0_{1\times3} & -e_{\vec{INS}}{}^N R_b^e(\vec{p}^b_{Rx}\times)\frac{\pi}{180} \end{bmatrix} \tag{5.19}$$

$$H_{k,TDCP} = \begin{bmatrix} -e_{\vec{INS}}{}^1(t) & e_{\vec{INS}}{}^1(t) - e_{\vec{INS}}{}^1(t-\Delta T) & 0 & 1 & 0 & 0 & -e_{\vec{INS}}{}^1(t) R_e^b(t)(-\vec{p}^b_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -e_{\vec{INS}}{}^N(t) & e_{\vec{INS}}{}^N(t) - e_{\vec{INS}}{}^N(t-\Delta T) & 0 & 1 & 0 & 0 & -e_{\vec{INS}}{}^N(t) R_e^b(t)(-\vec{p}^b_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \end{bmatrix} \tag{5.20}$$

The observation covariance matrix for code pseudoranges and Doppler measurement is unchanged. The covariance for TDCP measurements is still only composed of diagonal elements. The variance of a TDCP measurement can be given as the variance of the remaining TDCP error, $\Delta_T E^i_{Rx}$, chosen here as $2\sigma^2_\phi$.

## 5.4 Cycle Slip Detection

As highlighted by (5.1), the ambiguous term present in carrier phase measurements is absent of TDCPs under one condition: no cycle slips occur during the sampling interval. If a cycle slip occurs during this time interval, (5.1) can be re-written as:

$$\Delta_T \phi^i_{Rx} = \phi^i_{Rx}(t + \Delta T) - \phi^i_{Rx}(t) + \Delta N \lambda \tag{5.21}$$

The TDCP filter does not estimate the ambiguity nor does it try to estimate the ambiguity change, so ideally the cycle slip affected TDCP should be discarded by the filter. To do so, two methods are implemented in the filter:

1. EKF rejection based on the innovations and their covariance as proposed in [Tanil, 2022]
2. Threshold comparison between the wavelength of the signal and the expected observation noise of the innovation implemented in [Kim et al., 2020]

In a hypothesis test, there are four distinguishable cases shown in Table 5.1.

| Expected / Truth | $H_0$ | $H_1$ |
|---|---|---|
| $H_0$ | $P_N$ | $P_{Fa}$ |
| $H_1$ | $P_{Md}$ | $P_D$ |

Table 5.1: Hypothesis Test Probabilities

Where:

- $P_N$ is the probability of not detecting a cycle slip when there is none
- $P_{Fa}$ is the probability of detecting a cycle slip when there is none
- $P_{Md}$ is the probability of not detecting a cycle slip when there is one
- $P_D$ is the probability of detecting a cycle slip when there is one

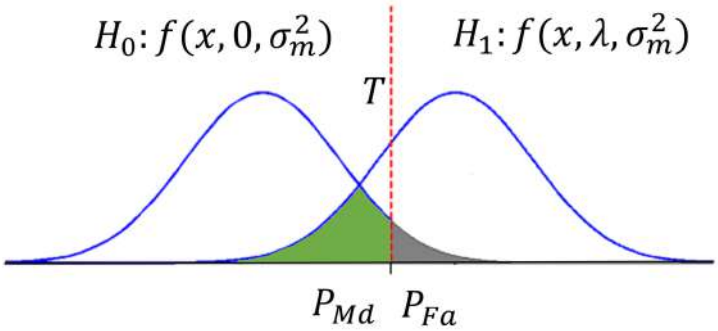


Figure 5.3: Hypothesis Test for cycle slips.

As illustrated by Figure 5.3 and Table 5.1, there is a trade-off between a low missed detection and a low false alarm probability. A false alarm in this case will reduce the TDCP measurement availability as it will be excluded from the EKF process whereas a missed detection will result in a cycle slip being processed in the EKF.

### 5.4.1 EKF based Cycle Slip Rejection

The EKF rejection is not specifically made to detect cycle slips but rather to monitor the innovations and exclude abnormally large innovations. Cycle slips can produce large innovations for TDCPs that exceed a threshold determined based on the expected measurement noise, given in (5.12). First the innovation covariance is computed:

$$I_{k,cov} = (H_k + J_k\Phi_{k-1}^{-1})P_{k-1}^{-}(H_k + J_k\Phi_{k-1}^{-1})^T + R \tag{5.22}$$

The innovations are normalized by the innovation covariance and then squared, resulting in the metric to be compared to a given threshold as given by [Tanil, 2022]:

$$M = I_k^T I_{k,cov}^{-1} I_k \tag{5.23}$$

M has a chi square distribution with a number of degrees of freedom equal to the number of measurements processed. As the presented filter processes measurements sequentially, it equals one in this case. The selection of the threshold depends on the desired sensitivity vs false alarm rate. In this project, the threshold is obtained by fixing a measurement rejection rate as follows:

$$T = Inv - \chi^2(1 - P_{EKF_{rej}}) \tag{5.24}$$

Where:

- $T$ is the threshold
- $P_{EKF_{rej}}$ is the desired probability at which the filter rejects a measurement set at 0.1 in this method.

Finally, the decision to exclude or not the measurement was made as such:

$$\text{Decision} = \begin{cases} M \geq T \rightarrow \text{Reject measurement} \\ M < T \rightarrow \text{Accept Measurement} \end{cases} \tag{5.25}$$

### 5.4.2 Wavelength based Cycle Slip Rejection

This method is a slightly modified version of the algorithm proposed in [Du and Gao, 2012] as they were using double differenced TDCPs which influenced their noise model. The smallest cycle slip that can exist in (5.21) is $\Delta N = 1$. So, the detector should be able to detect a change superior or equal to the wavelength of the processed signal (or signals if using ionosphere free TDCPs).

(5.2) is re-used but also adding the cycle slip to the measurement. Then by using the predicted TDCP measurement from the filter, the two measurements are differenced (which is also the innovation). This yields:

$$M = \Delta_T\phi_{Rx}^i - \Delta_T\phi_{INS}^i \tag{5.26}$$

To detect a cycle slip, the terms other than the displacement of the receiver affecting $M$ must be accounted for. The difference between the predicted and observed satellite and receiver clock drift must be small.

With the variance of M being:

$$\sigma_M^2 = \text{var}(\Delta_T c\dot{dt}_{Rx}) + \text{var}(\Delta_T \vec{r}_{Rx_{INS}}) + \sigma_{\Delta_T E}^2 \tag{5.27}$$

Where $\text{var}(\Delta_T c\dot{dt}_{Rx}), \text{var}(\Delta_T \vec{r}_{Rx_{INS}}), \sigma_{\Delta_T E}^2$, and $\sigma_M^2$ are the variance of the clock drift difference, the variance of the IMU's positioning error, the variance of the remaining errors, and the variance of the metric M respectively.

To decide whether a cycle slip occurs, a Neyman-Pearson test is implemented. A NP test makes two hypotheses [Neyman and Pearson, 1933]:

- $H_0$: No Cycle slip occurs, there is only noise.
- $H_1$: A Cycle slip occurs: there is noise and at least a change of one wavelength.

The detection threshold can then computed as:

$$T = \lambda \tag{5.28}$$

Where:

- $\lambda$ is the wavelength of the signal.

The hypothesis $H_0$ can be modeled as a centered chi squared distribution, as the variance of M is a sum of Gaussian variables [Rahman et al., 2015]. On the other hand, the hypothesis $H_1$ can be also be modeled as a chi squared distribution but with a non-centrality parameter that is equal to $\lambda$.

The decision on the cycle slip presence is given as follows:

$$\text{Decision} = \begin{cases} M \geq T \rightarrow \text{Reject measurement} \\ M < T \rightarrow \text{Accept Measurement} \end{cases} \tag{5.29}$$

## 5.5 Method and Scenarios

The filter was run on several datasets that were collected either using IGS data from the TLSE reference station or using a vehicle in Toulouse, France (in different environments). Dynamic datasets include open sky scenarios, mid-urban environments, and deep urban scenarios. On the other hand, the IGS data is collected in an open-sky, static environment and has dual frequency observables. The precise clock and orbit products (rapid), and phase center offset files were also downloaded and used in the filter's navigation solution to emulate a real-time PPP solution.

**IGS Dataset**
The IGS dataset was used to verify that the filters were performing as expected. Indeed,

the IGS station is in an open sky environment, static with a geodetic antenna. The GNSS receiver used to generate the observables is the Trimble NETR9 on the L1 (1575.42 MHz) and L5 (1176.45 MHz) bands. Due to the lack of IMU data from this IGS station, this version of the EKF filter was implemented in GNSS-only. The observables were available at a rate of 1 Hz and gathered on the 4th of October 2021 for a duration of one hour.

**Dynamic Datasets**
The dynamic datasets were obtained with the same equipment but the IMU for all datasets. A Novatel passive antenna was used and connected to a uBlox F9P receiver generating the measurements at a rate of 5 Hz. There were two IMUs used. One was a low-cost MEMS: the Bosch BMI160 at a rate of 200 Hz and the other was a high quality IMU from the Novatel SPAN at a rate of 200 Hz. The GNSS observables are obtained on Galileo (E1b + E5b) and GPS (L1 C/A + L2). Both types of IMUs were tested to evaluate whether the quality of the IMU influenced the conclusions of the proposed TDCP filter.

### 5.5.1 Synchronization of sensors

Since the MEMS IMU is not time synchronized with the uBlox receiver, both sensors had to be synchronized for the EKF measurement update to be coherent in time. Thus, all observations have been referenced to GPS system time (GPST) in post processing. In the following section, the synchronization process between the IMU and GNSS receiver is explained. This is further illustrated by Figure 5.4.

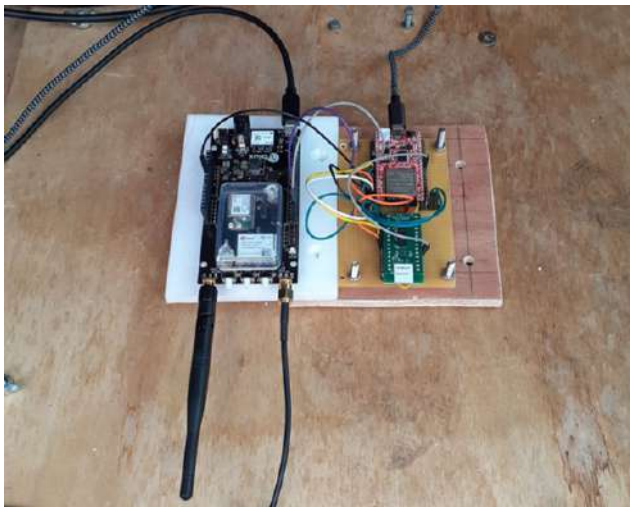

Figure 5.4: GNSS and IMU Synchronization Set-up.

The BMI160 IMU chip is configured to provide measurements at 200 Hz. The availability of new measurements is signaled to the host micro-controller (ESP32) via the BMI160's data-ready interrupt output. The host micro-controller then reads the BMI160's output registers via serial peripheral interface.

To enable synchronization to GPST, the ESP32 also receives the uBlox F9P's "Timepulse" output, triggering an interrupt on the ESP32 and the recording of a time stamp. In addition, the ESP32 decodes the corresponding UBX NAV-PVT message received from the uBlox F9P on its serial port to retrieve the GPST seconds of week corresponding to the most recent

"Timepulse" signal. Both pieces of information allow us to track the time-varying offset between the ESP32's CPU time and GPST. In post processing, a cubic spline is fitted to the observed time offset to obtain GPST time for all recorded CPU time stamps.

Observations of most GNSS receivers are given in the receiver's time, and the F9P is no exception. The unknown receiver clock offset is typically simply solved as part of the estimation. However, to process GNSS observations at the correct point in time when running the navigation filter in post processing, the GNSS observations must be referenced to the same time scale as the observations of the IMU. To this end, to generate the dataset, the receiver clock bias is estimated using the reference solution position and satellite ephemeris and the GPST time stamps of each epoch of GNSS observations are computed.

### 5.5.2 Reference Trajectory

#### 5.5.2.1 IGS Dataset

The reference position of the IGS TLSE station is given by the IGS website.

#### 5.5.2.2 Dynamic Datasets

The SPAN reference system consists of a high-performance IMU and a GNSS receiver. In post processing with Novatel's Inertial Explorer software, inertial and differentially-corrected GNSS observations are combined in a forward-backward smoother. The SPAN GNSS receiver tracks GPS and Glonass constellations whose observations were used in post processing to obtain a more accurate reference trajectory.

#### 5.5.2.3 Map Reference Trajectories

Figures 5.5, 5.6, 5.7, and 5.8 depict the trajectories of the static data collect from the IGS data and the dynamic datasets using the high-end SPAN IMU for INS/GNSS hybridization.

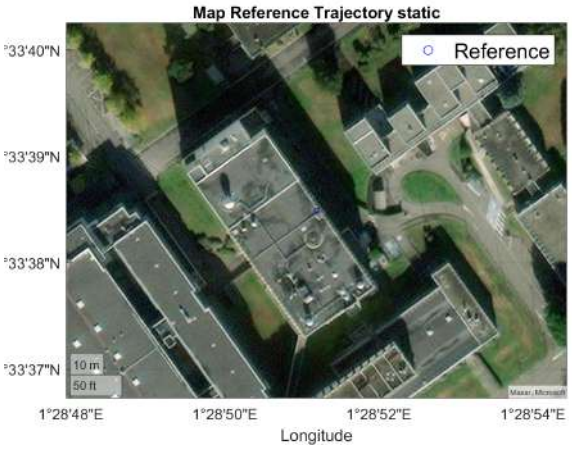


Figure 5.5: Map reference trajectory for the static dataset of the 4th of October 2021.

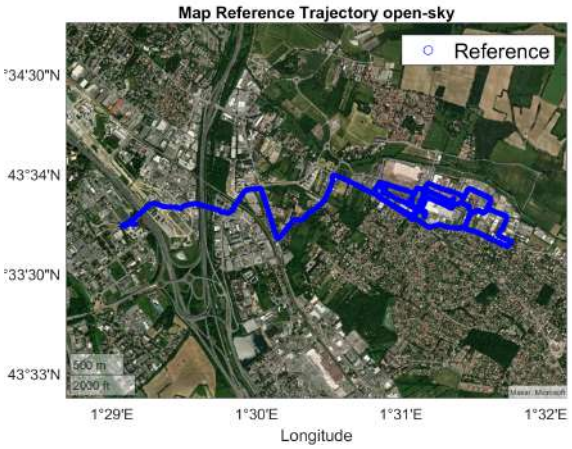


Figure 5.6: Map reference trajectory for the open-sky dataset of the 7th of October 2020.

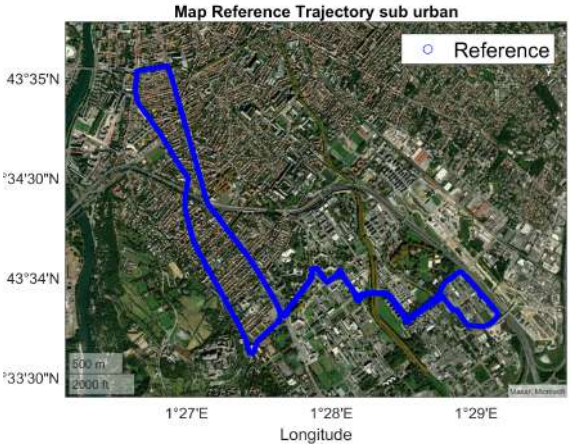


Figure 5.7: Map reference trajectory for the sub urban dataset of the 16th of December 2020.

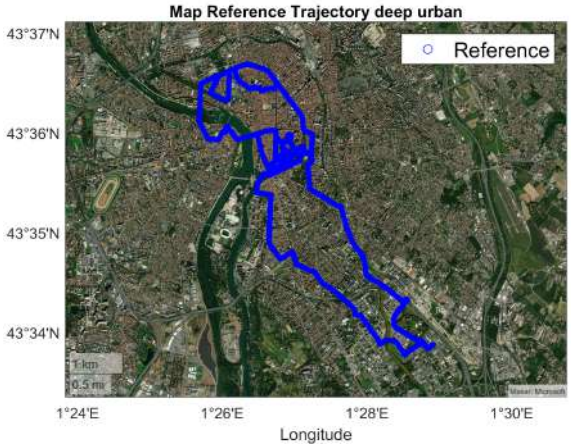


Figure 5.8: Map reference trajectory for the deep urban dataset of the 27th of June 2023.

### 5.5.3 Differences between Datasets

To quantify the benefit of using TDCP measurements, several environments had to be explored in the datasets. Hence, there are two open sky, two mid-urban environments, and two deep urban scenarios. The dates and duration of each scenario is given below.
For the high-end SPAN IMU data collects:

- 7th of October 2020: Highway/open-sky of 1 hour 45 minutes
- 16th of December 2020: Sub Urban of 2 hours 15 minutes
- 27th of June 2023: Deep Urban of 2 hours 30 minutes

For the MEMS IMU data collects:

- 29th of October 2020: Mid-Urban of 12 minutes
- 23rd of April 2021: Deep Urban of 15 minutes

- 21st of September 2021: Highway with few bridges for a total of 20 minutes

## 5.6 Results

This section is divided into two subsections: the IGS dataset and the dynamic data collects. The first results presented are from the IGS data collect to both explore the addition of TDCPs in a static open sky environment and the performance of the presented filters in "ideal" conditions.

The second part of the results are for the dynamic datasets which highlight the benefit of using TDCP measurements in different scenarios.

To illustrate the benefit of using TDCPs in the EKF instead of raw carrier phase measurements, variants of the EKF were also implemented. Indeed, a filter using only code pseudoranges and Doppler measurements was designed along with a filter using code + Doppler + carrier phase measurements that estimates the ambiguities. The code and Doppler only filter is the baseline filter while the code + Doppler + carrier phase measurements filter is the filter to compare the proposed TDCP filter. Furthermore, the TDCP filter will be evaluated by using either uncombined TDCPs or first order ionosphere free ones. Both types of measurements will be tested by the cycle slip rejection algorithms - the EKF based rejection and the wavelength methods detailed in 5.4.

The results will also assess the computation times of the three different filters to present the computational gain of using TDCP measurements instead of raw carrier phase measurements in the EKF.

These filters are denoted with different names according to their versions:

- v2: Uses Code + Doppler measurements in the EKF measurement update, with the states of (5.15)
- v3: Uses Code + Doppler + Carrier phase measurements in the EKF measurement update, with the states of (3.60)
- v3TDCP: Uses Code + Doppler + TDCP measurements in the EKF measurement update, with the states of (5.15). When the EKF cycle slip rejection based method is used, it will be denoted as $\text{v3}, \text{I}_{rej}$, and the wavelength method will be denoted as $\text{v3}, \lambda_{rej}$. The benefit of using first order iono-free TDCPs versus uncombined TDCPs is also investigated: iono-free measurements are denoted as $\text{v3}_{TDCP_{IF}}$ and uncombined ones as $\text{v3}_{TDCP_{Unc}}$.

The dynamic results were obtained using dual frequency observables on GPS (L1 C/A and L2) and Galileo (E1-B + E5b). The full results will be presented in tables while the CDFs of the ambiguity estimating filter and only of the best performing TDCP filter will be presented for the sake of readability. The data collects using the IMU of the SPAN for the INS hybridization are to be seen as the main data collects of this work. The results using the MEMS IMU are to be seen as tests to emulate the eHermes conditions and are too short to draw definite conclusions from.

### 5.6.1 IGS Dataset

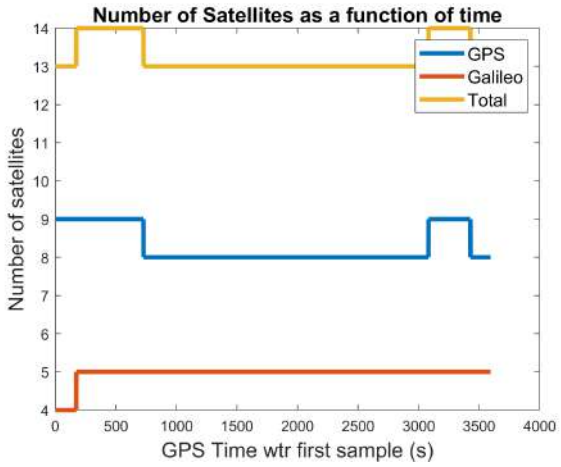


Figure 5.9: Number of satellites as a function of time.

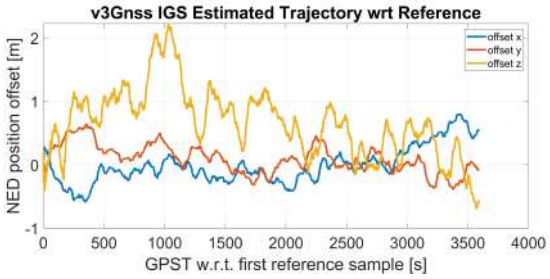


Figure 5.10: IGS time series v3.

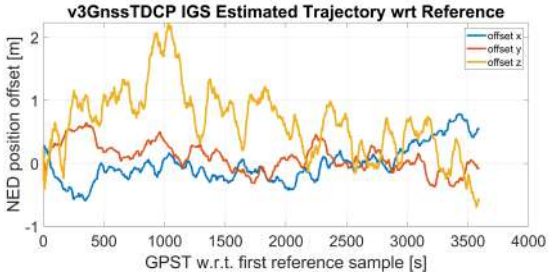


Figure 5.11: IGS time series v3TDCP.

Figure 5.9 represents the number of satellites over the observation time. The number of satellites is identical for both the TDCP based filter and the conventional ambiguity estimation filter. It can be seen that the number of satellites is steady thanks to the open sky and static conditions.

Figures 5.10 and 5.11 depict the estimated position with respect to the reference in the North East Down (NED) coordinate frame as a function of time for the ambiguity estimation and TDCP filter respectively.

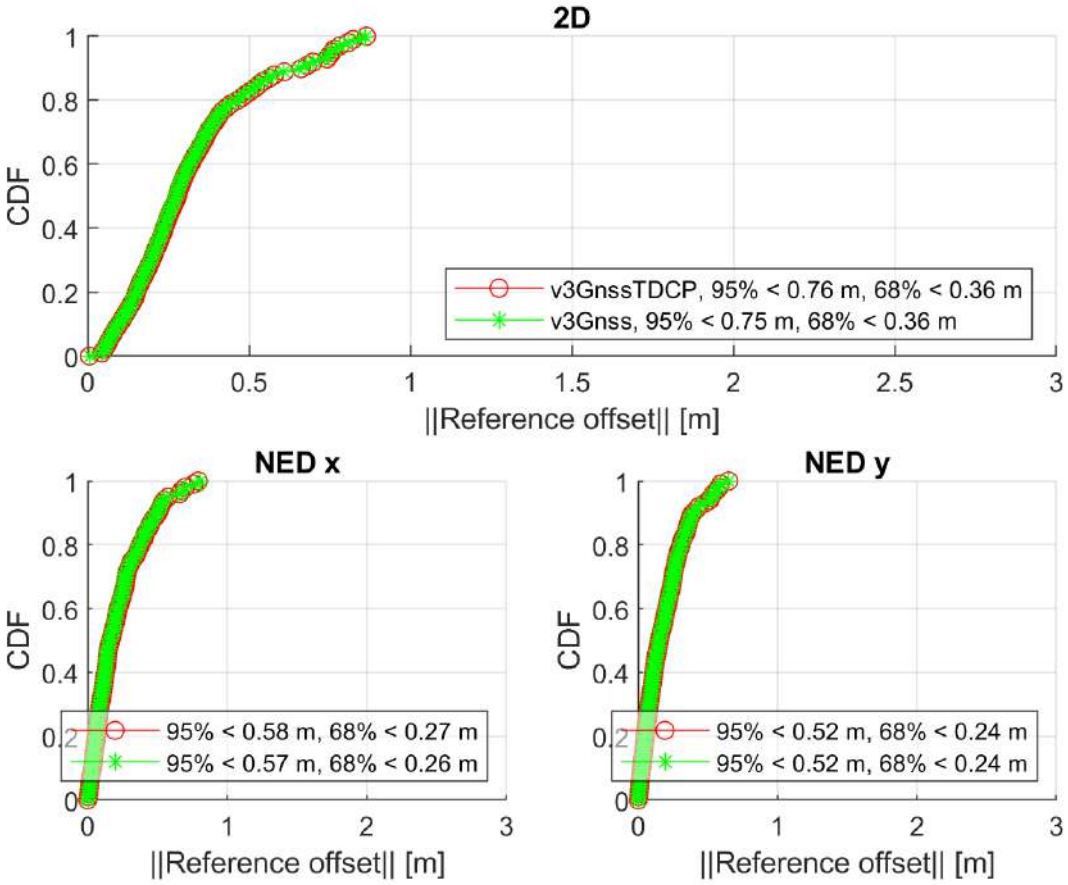


Figure 5.12: IGS CDFs.

Figure 5.12 shows the cumulative distribution function (CDF) of the position error in meters for the x and y axes of the NED (north and east). This batch of results was generated using dual frequency observables on GPS (L1 C/A and L2) and Galileo (E1-B + E5b) that were combined to remove the first order ionospheric term.

Both filters display low positioning error and validate the proposed implementation. However, the benefit of TDCP measurements in a static environment seems to be non-existent as the performance of the TDCP filter is equal to the one without its inclusion (v2). This is not surprising as TDCPs estimate the distance between two consecutive epochs. As there is no distance being traveled, the code pseudoranges take over the precision of the filter.

Moreover, the performance of the filter is not improved by adding raw carrier phase measurements either. This can be explained by the high quality of code pseudoranges. Indeed, the IGS station uses a choke ring antenna which is robust to multipath [Tranquilla et al., 1994] and is in static, open-sky.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 0.36 | 0.76 | 0.28 |
| v3 | 0.36 | 0.75 | 1.00 |
| $v3_{TDCP_{IF}}, I_{rej}$ | 0.36 | 0.76 | 0.32 |
| $v3_{TDCP_{Unc}}, I_{rej}$ | 0.36 | 0.76 | 0.43 |
| $v3_{TDCP_{IF}}, \lambda_{rej}$ | 0.36 | 0.76 | 0.33 |
| $v3_{TDCP_{Unc}}, \lambda_{rej}$ | 0.36 | 0.76 | 0.38 |

Table 5.2: IGS CDF results for all configurations.

It is noteworthy that for static conditions with a steady total of 13 satellites, the TDCP filter is much faster than when using carrier phase measurements regardless of the nature of the TDCPs (first order ionosphere free or uncombined) and the exclusion method. This mainly results from a much lower number of states in the state vector.

### 5.6.2 SPAN IMU Datasets

This section presents the dynamic data collects obtained from using the IMU of the SPAN system for the INS/GNSS hybridization.

#### 5.6.2.1 Open-Sky

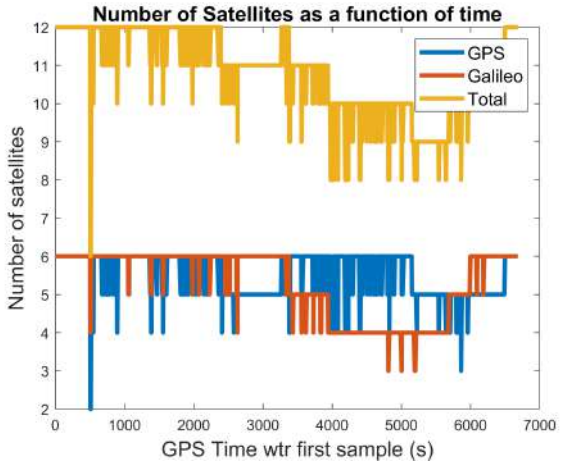


Figure 5.13: Number of satellites as a function of time.

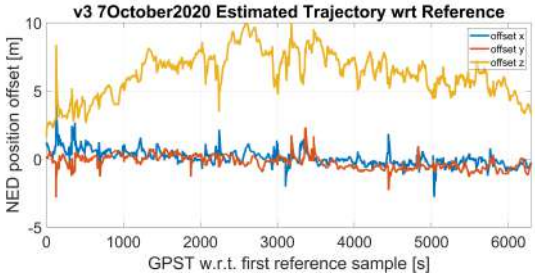


Figure 5.14: Highway Time series v3.

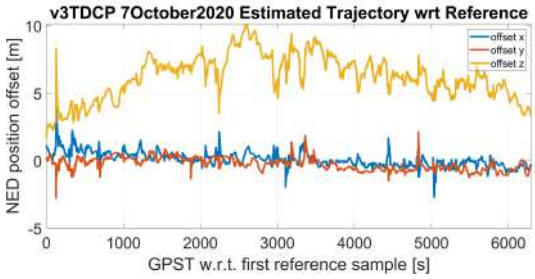


Figure 5.15: Highway Time series v3TDCP.

Figure 5.13 illustrates the number of satellites used as a function of time for both filters. There is a noticeable drop of satellites around the 300 seconds of observations which is due to signal blockage. Figures 5.14 and 5.15 show the estimated trajectories of both filters.

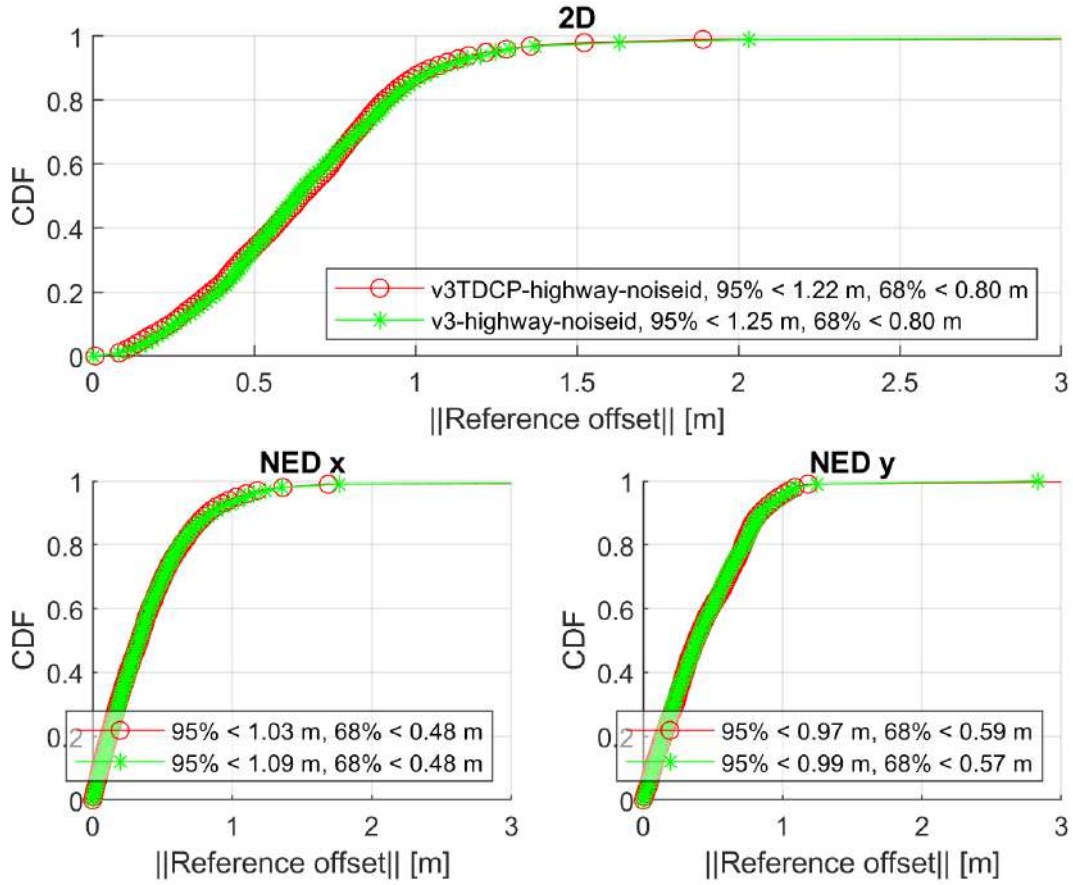


Figure 5.16: Highway CDFs (with an uncombined TDCP filter with EKF based rejection).

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 0.80 | 1.27 | 0.32 |
| v3 | 0.80 | 1.25 | 1.00 |
| $\text{v3}_{TDCP_{IF}}, \text{I}_{rej}$ | 0.79 | 1.24 | 0.48 |
| $\text{v3}_{TDCP_{Unc}}, \text{I}_{rej}$ | 0.80 | 1.22 | 0.57 |
| $\text{v3}_{TDCP_{IF}}, \lambda_{rej}$ | 0.80 | 1.26 | 0.41 |
| $\text{v3}_{TDCP_{Unc}}, \lambda_{rej}$ | 0.80 | 1.24 | 0.57 |

Table 5.3: Highway CDF results for all configurations.

The results from Table 5.3 highlight that in open-sky conditions, even in a dynamic scenario, the precision of the TDCP filter is mostly dictated by the code and Doppler observables. Indeed, the accuracy of the code and Doppler (v2) filter is very similar to the one of the TDCP filters. The difference in accuracy is not big enough to claim, in this case, an added value from the TDCPs. Furthermore, the different configurations on the TDCP filter do not provide any significant difference between the configurations. This conveys a limited use of the TDCP measurements in low obstructed environments.

#### 5.6.2.2 Sub-urban

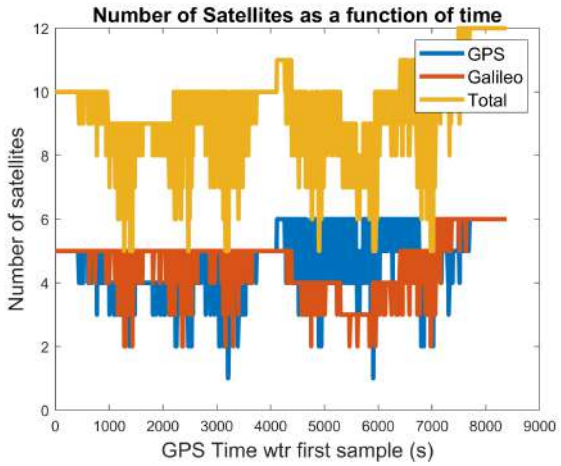


Figure 5.17: Number of satellites as a function of time.

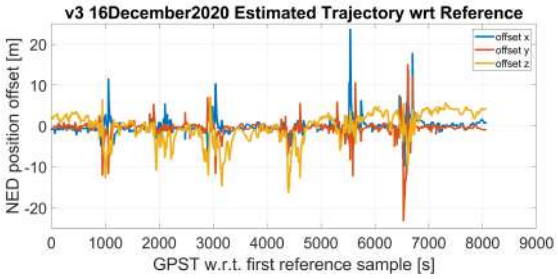


Figure 5.18: Sub-Urban Time series v3.

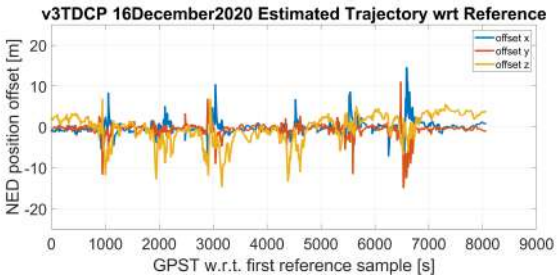


Figure 5.19: Sub-Urban Time series v3TDCP.

Figure 5.17 depicts the number of satellites over the dataset for the filters. It can be noted that the number of satellites have frequent significant drops. This is reflected in the estimated trajectories of the filters in Figures 5.18 and 5.19 with positioning error spikes.

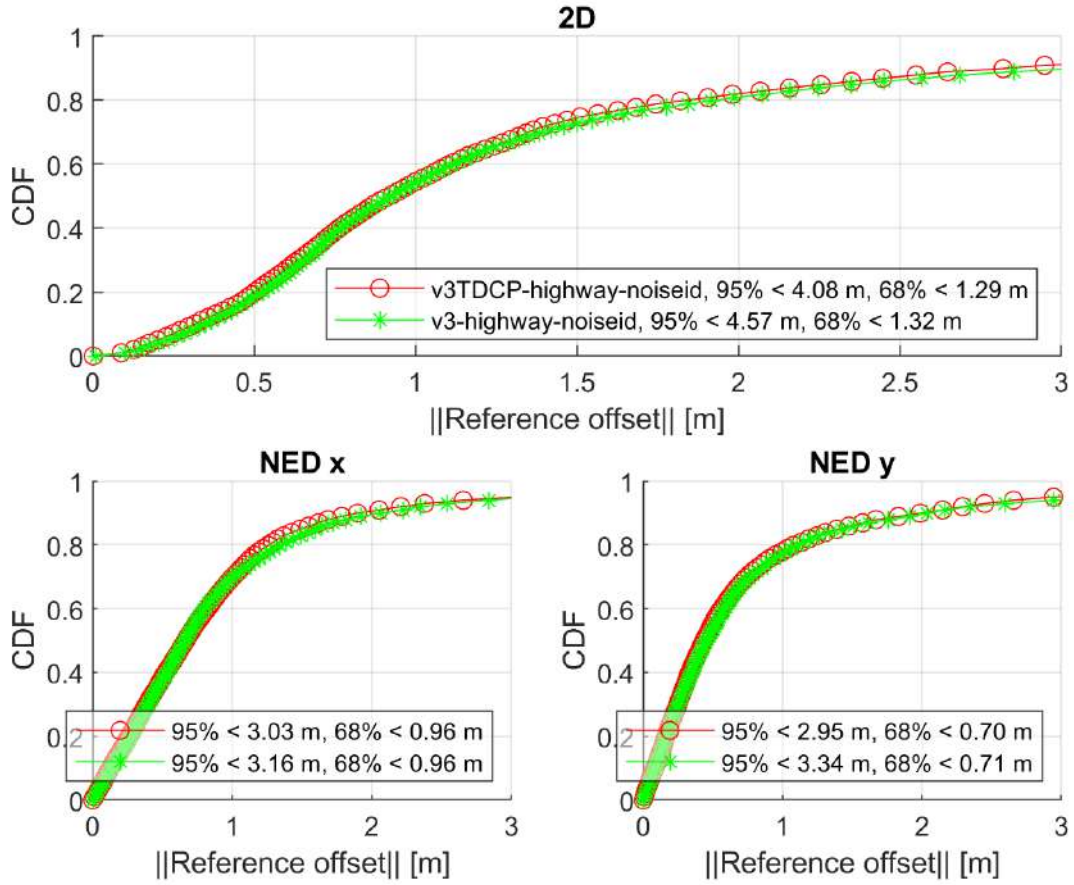


Figure 5.20: Sub-Urban CDFs (with an uncombined TDCP filter with EKF based rejection).

The CDF of both filters from this data collect is presented in Figure 5.20. For this scenario, the TDCP filter provides a 10.7% positioning accuracy gain with respect to the carrier phase measurement for the 95% horizontal error.

This can be explained by the several signal obstructing moments as shown by Figure 5.17. The occurrences at $\sim$ 1000, $\sim$ 2500, $\sim$ 5500 and $\sim$ 6500 seconds were slightly smoothed out by the TDCP filter as highlighted by Figure 5.19. The spikes contained in the carrier estimation filter are also present in the v2 filter. This implies that this comes from the code measurements which are likely affected by multipath. On the other signal obstructions, the TDCP filter does not outperform the v3 filter. For most of the obstruction, there were a low number of visible satellites; hence, the accuracy of the filter relies more heavily on the drifting INS.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 1.37 | 4.71 | 0.25 |
| v3 | 1.32 | 4.57 | 1.00 |
| $v3_{TDCP_{IF}}, I_{rej}$ | 1.35 | 4.22 | 0.34 |
| $v3_{TDCP_{Unc}}, I_{rej}$ | 1.29 | 4.08 | 0.36 |
| $v3_{TDCP_{IF}}, \lambda_{rej}$ | 1.38 | 4.44 | 0.44 |
| $v3_{TDCP_{Unc}}, \lambda_{rej}$ | 1.35 | 4.23 | 0.58 |

Table 5.4: Sub-urban CDF results for all configurations.

The benefit of TDCP measurements in suburban environments is that it slightly outperforms a simple ambiguity estimation process while being 64% faster, when using uncombined TDCPs with an innovation based rejection method. Furthermore, the code and Doppler only filter is outperformed by 13.8% in 95% of the horizontal error by the TDCP filter. This further highlights that TDCPs can help smooth out the solution when obstacles are present. In Table 5.4, it is shown that the filters processing uncombined TDCPs outperform the iono-free TDCP based filters. This implies that there is an added benefit to using more TDCP measurements in the EKF than fully removing the first order ionosphere effect.

The TDCP filter is still much faster than the ambiguity estimation filter despite a lower number of satellites with respect to the IGS dataset indicating that the computational gain comes from the lower number of states in the EKF.

#### 5.6.2.3 Deep Urban

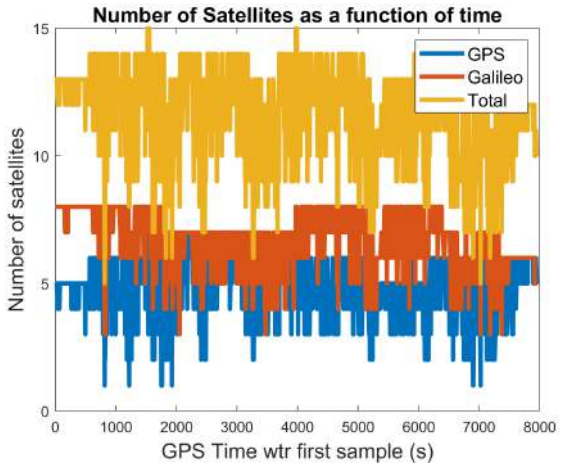


Figure 5.21: Number of satellites as a function of time.

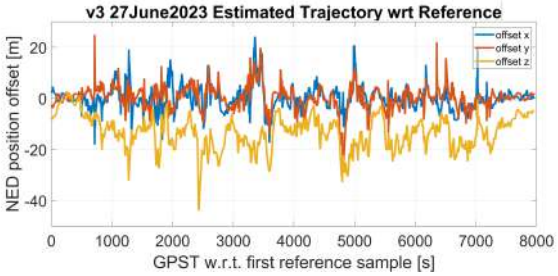


Figure 5.22: Urban Time series v3.

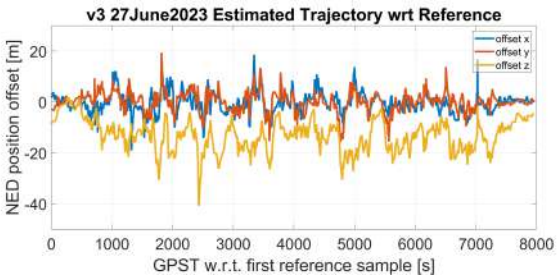


Figure 5.23: Urban Time series v3TDCP.

Figure 5.21 shows the number of satellites as a function of time for the deep urban dataset. The number of satellites used varies frequently with low points of 5 satellites being reached

around the 1000 and 7000 second marks. It can be seen in 5.23 that the v3TDCP filter is less affected by the sudden drop of satellites (e.g. at $\sim$ 1000 seconds) unlike the v3 filter (5.22) in which several positioning error spikes are present.

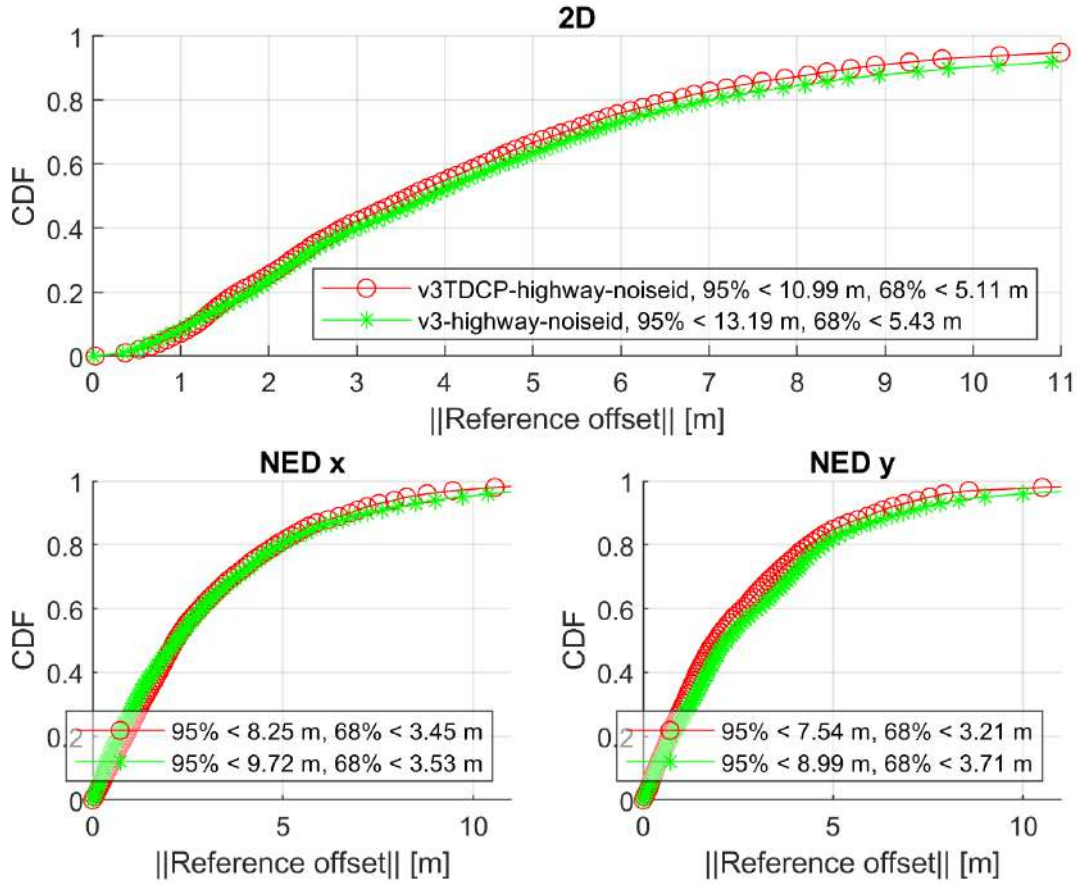


Figure 5.24: Urban CDFs (with an uncombined TDCP filter with EKF based rejection).

As highlighted by Figure 5.24, the uncombined TDCP filter with innovation based rejection brings a significant positioning improvement in deep urban conditions with respect to the ambiguity filter. As shown by Figures 5.22 and 5.23, the time series of the TDCP filter is much smoother thanks to the lack of ambiguity estimation. Indeed, this frequent reset in the EKF, due to cycle slips, leads to position uncertainty. The constant reset is highlighted by the constant varying number of satellites in Figure 5.21. This scenario demonstrates that TDCP measurements are most useful in heavily obstructed environments.

During cycle slips, TDCP measurements are only excluded for that epoch and then the filter goes back to using them as the position increments are computed. The filter is on average 57% faster (in MATLAB) than the ambiguity estimating filter and performs 16.7% better for the 95% horizontal error. Finally, the TDCP filter is 7.6% and 33.3% more accurate for 68% and 95% of the horizontal error respectively than the v2 filter while only taking 39% longer per epoch. This illustrates the perks of including TDCPs in the filter as the environment of the receiver becomes more obstructed.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 5.53 | 16.49 | 0.26 |
| v3 | 5.43 | 13.19 | 1.00 |
| $v3_{TDCP_{IF}}, I_{rej}$ | 5.07 | 12.45 | 0.33 |
| $v3_{TDCP_{Unc}}, I_{rej}$ | 5.11 | 10.99 | 0.43 |
| $v3_{TDCP_{IF}}, \lambda_{rej}$ | 5.66 | 19.17 | 0.46 |
| $v3_{TDCP_{Unc}}, \lambda_{rej}$ | 5.28 | 13.66 | 0.47 |

Table 5.5: Urban CDF results for all configurations.

Table 5.5 further highlights that the TDCP filter can outperform v3. However, it also demonstrates that the tuning of the filter is very important. There is a big positioning discrepancy between the filter using ionosphere-free TDCPs and excluding them with innovation monitoring and the one based on wavelength difference. When excluding iono-free TDCPs with the latter, the wavelength of the combined measurement is reduced as shown by (5.30).

$$\lambda_{IF} = c\frac{f_1 - f_2}{f_1^2 - f_2^2} \tag{5.30}$$

Where:

- $f_1$ is the carrier frequency of the first signal in Hertz
- $f_2$ is the carrier frequency of the second signal in Hertz
- $\lambda_{IF}$ is the wavelength of the first order iono-free combination signal in meters

For GPS L1 C/A, its signal has a wavelength of approximately 19 cm while the iono-free combination of GPS L1 C/A and GPS L2 has a wavelength of around 10 cm. So, by factoring the shorter wavelength and the increased noise due to the combination [Salós et al., 2010], the cycle slip detection method presented in 5.4.2 becomes even more stringent. This leads to a high rejection of TDCPs (15% vs 1%), most of which are not affected by cycle slips and thus a very degraded solution.

For uncombined TDCPs as well, the lambda based rejection method is stricter as it rejects 4% of uncombined TDCPs whereas the innovation method rejects 1%. The lambda rejection method also rejects useful TDCP measurements which leads to a sub-optimal use of TDCP measurements.

### 5.6.3 MEMS IMU Datasets

This section presents the results of the data collects when using a MEMS IMU for the INS/GNSS hybridization in an effort to emulate the eHermes. Due to the relatively short duration of these datasets, firm conclusions cannot be made, and the conclusions from the SPAN datasets prevail.

#### 5.6.3.1 Highway Data Collect

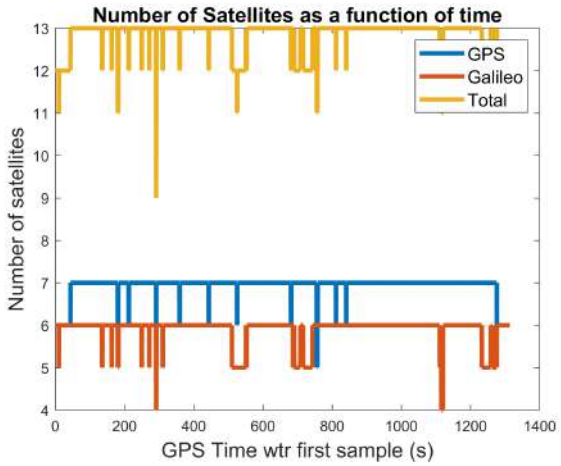


Figure 5.25: Number of satellites as a function of time.

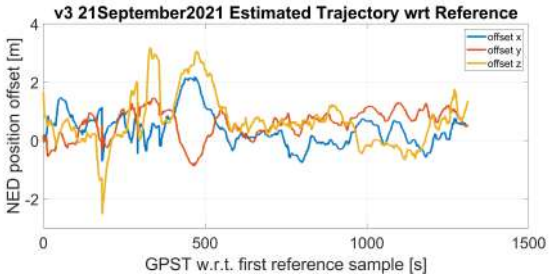


Figure 5.26: Highway Time series v3.

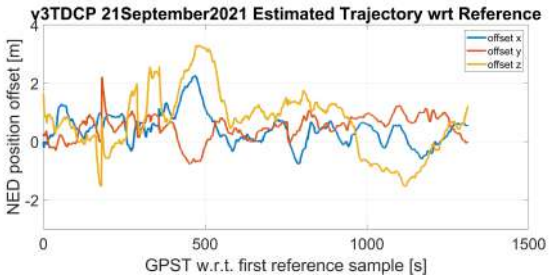


Figure 5.27: Highway Time series v3TDCP.

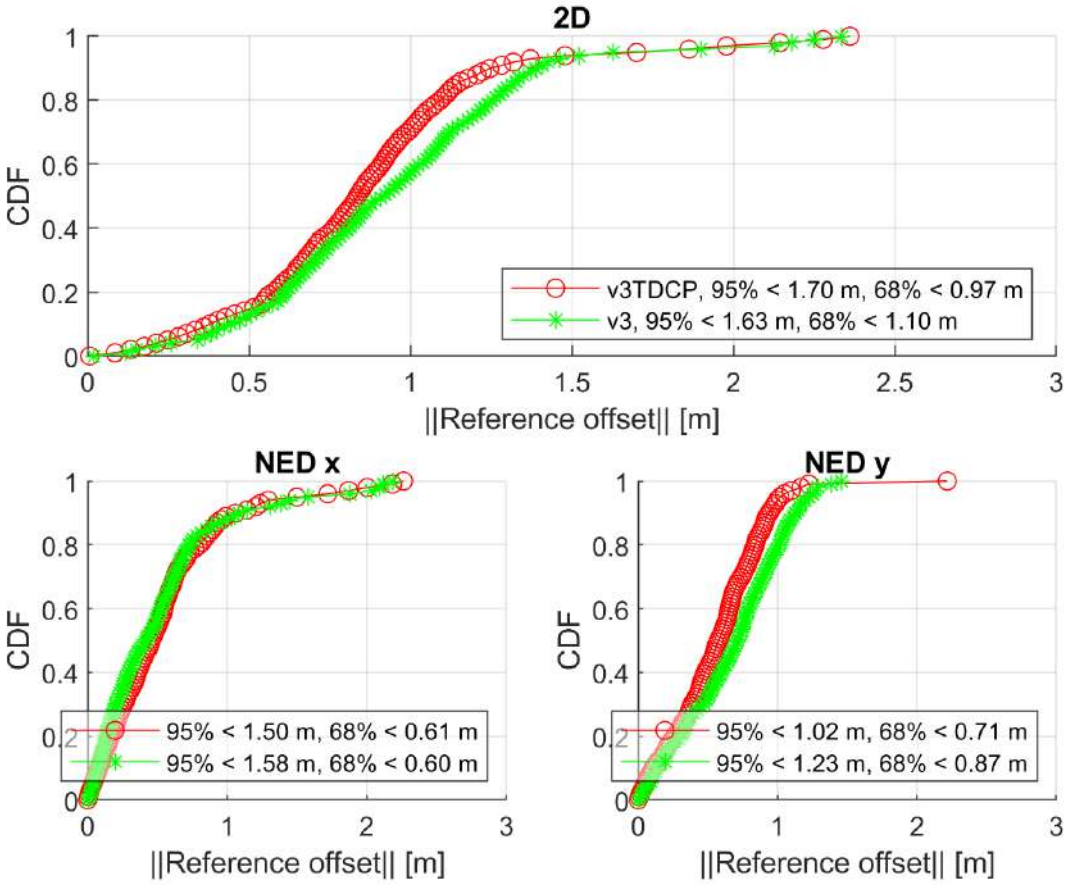


Figure 5.28: Highway CDFs.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 1.00 | 0.97 | 0.28 |
| v3 | 1.10 | 1.63 | 1.00 |
| $\text{v3}_{TDCP_{IF}}, \text{I}_{rej}$ | 0.98 | 1.70 | 0.32 |
| $\text{v3}_{TDCP_{Unc}}, \text{I}_{rej}$ | 0.97 | 1.70 | 0.43 |
| $\text{v3}_{TDCP_{IF}}, \lambda_{rej}$ | 0.99 | 1.73 | 0.33 |
| $\text{v3}_{TDCP_{Unc}}, \lambda_{rej}$ | 1.01 | 1.71 | 0.38 |

Table 5.6: Highway CDF results for all configurations.

The results from Table 5.6 and Figure 5.28 highlight that the precision of the TDCP filter is dictated by the code and Doppler observables. Furthermore, the different configurations on the TDCP filter do not provide any significant difference between the configurations.

#### 5.6.3.2 Sub-urban Data collect

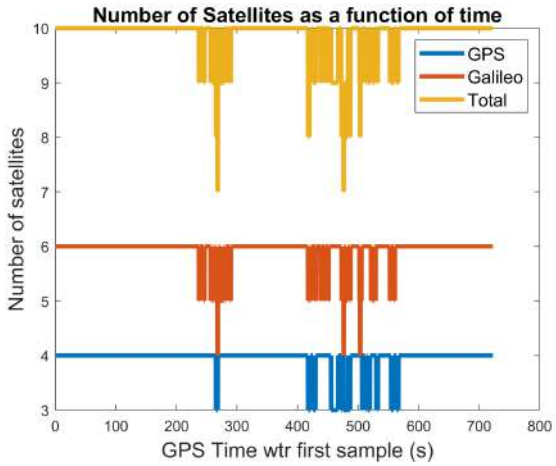


Figure 5.29: Number of satellites as a function of time.

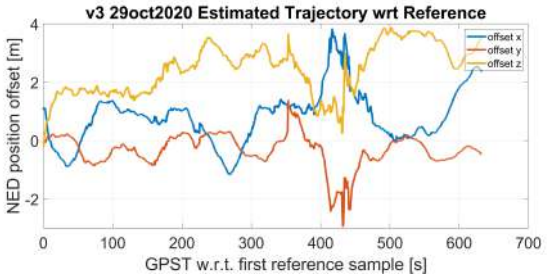


Figure 5.30: Sub-Urban Time series v3.

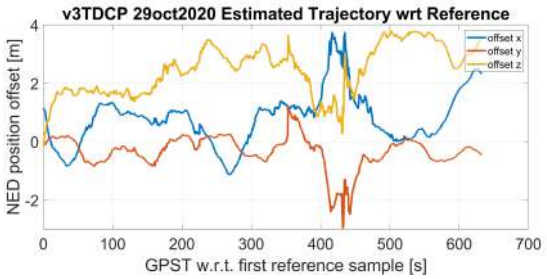


Figure 5.31: Sub-Urban Time series v3TDCP.

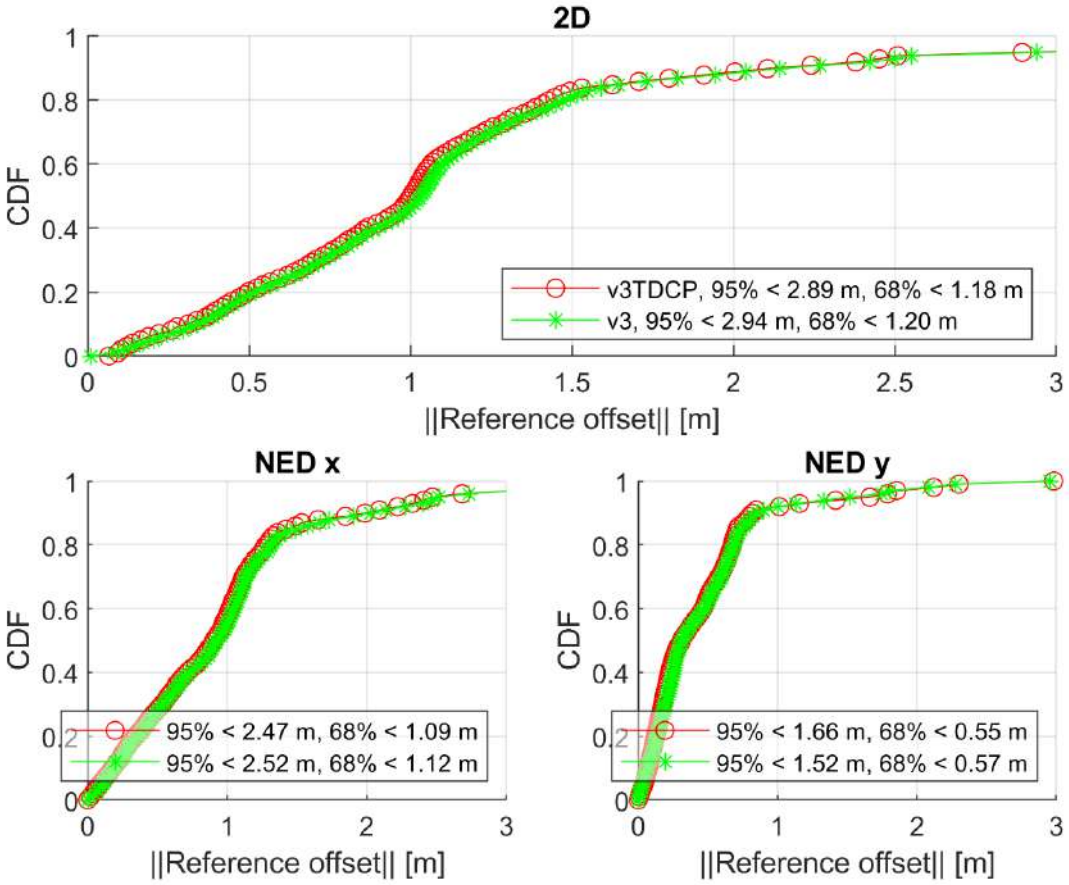


Figure 5.32: Sub-urban CDFs.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 1.21 | 3.11 | 0.28 |
| v3 | 1.20 | 2.94 | 1.00 |
| $\text{v3}_{TDCP_{IF}}, \text{I}_{rej}$ | 1.18 | 2.89 | 0.32 |
| $\text{v3}_{TDCP_{Unc}}, \text{I}_{rej}$ | 1.18 | 3.05 | 0.43 |
| $\text{v3}_{TDCP_{IF}}, \lambda_{rej}$ | 1.19 | 2.92 | 0.33 |
| $\text{v3}_{TDCP_{Unc}}, \lambda_{rej}$ | 1.18 | 3.01 | 0.38 |

Table 5.7: Sub-urban CDF results for all configurations.

For this scenario, the TDCP filter provides a marginal gain with respect to the carrier phase measurement. The benefit of TDCP measurements in suburban environments is that it matches the performance of a simple ambiguity estimation process while being 66% faster. Furthermore, the code and Doppler only filter is outperformed by 7% in 95% of the time by the TDCP filter. This further highlights that TDCPs can help smooth out the solution when obstacles are present.

#### 5.6.3.3 Deep Urban Data collect

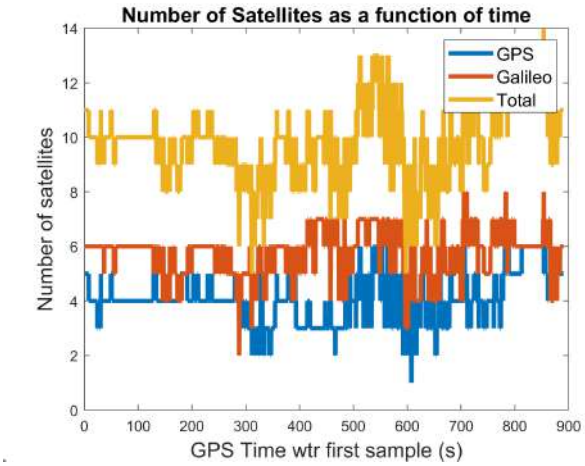


Figure 5.33: Number of satellites as a function of time.

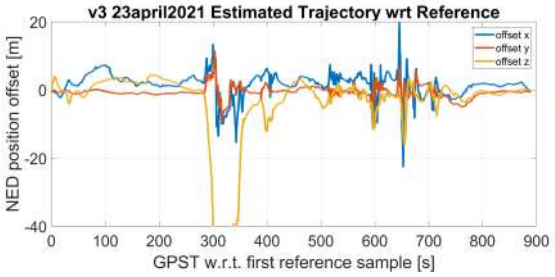


Figure 5.34: Urban Time series v3.

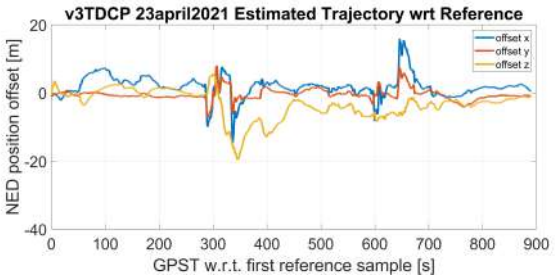


Figure 5.35: Urban Time series v3TDCP.

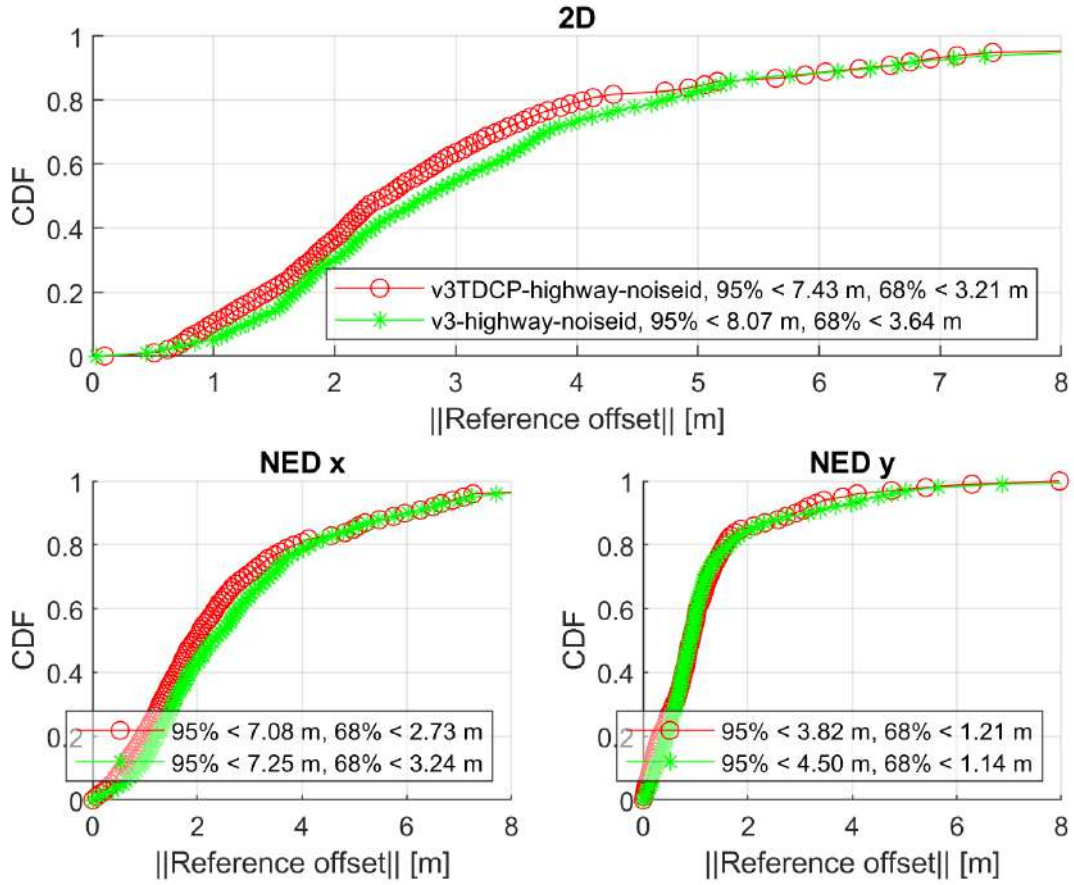


Figure 5.36: Urban CDFs.

| Filter Version | 68% Horizontal Error | 95% Horizontal Error | Normalized Exec Time |
|---|---|---|---|
| v2 | 5.21 | 15.52 | 0.28 |
| v3 | 3.64 | 8.07 | 1.00 |
| $v3_{TDCP_{IF}}, I_{rej}$ | 3.21 | 7.43 | 0.32 |
| $v3_{TDCP_{Unc}}, I_{rej}$ | 3.31 | 11.60 | 0.43 |
| $v3_{TDCP_{IF}}, \lambda_{rej}$ | 4.74 | 25.5 | 0.33 |
| $v3_{TDCP_{Unc}}, \lambda_{rej}$ | 3.72 | 9.17 | 0.38 |

Table 5.8: Urban CDF results for all configurations.

As shown by Figures 5.34 and 5.35, the time series of the TDCP filter is much smoother thanks to the lack of ambiguity estimation. On top of this accuracy gain of 11%, the filter is on average 66% faster (in MATLAB) than the ambiguity estimating filter. Finally, the TDCP filter is 39% and 52% more accurate for 68% and 95% of the epochs respectively than the v2 filter while only taking 25% longer per epoch.

Table 5.8 further conveys that the tuning of the filter is very important. There is a big positioning discrepancy between the filter using ionosphere-free TDCPs and excluding them with innovation Monitoring and the one based on wavelength difference. As discussed on

the SPAN datasets in section 5.6.2, the wavelength of the combined measurement is reduced as shown by (5.30), which impacts the correct detection of cycle slips when using the wavelength difference method.

It can also be seen that the filter using ionosphere-free TDCPs performs better on the 95% horizontal positioning than the one using uncombined TDCPs with the same cycle slip detection method (innovation based). As the data collect is of 15 minutes, the 95% positioning error is based on the worst 45 seconds of data. This short amount of time is too small to conclude that the ionosphere-free TDCP performs better than the uncombined one. There could have been a TDCP that was affected by a cycle slip and not excluded in this time which could cause the filter to reconverge. These two filters also perform similarly for 68% of the horizontal positioning error which is more telling on their overall performance for this dataset.

With more data, it is presumed that the uncombined filter using innovation rejection of cycle slips would perform similarly or better than the iono-free TDCP filter as shown in 5.6.2.

## 5.7 Conclusions

This chapter presented how to form time differenced carrier phase measurements as well as how to use them in an EKF based solution. Existing PPP solutions were also discussed along with the differences with the presented one. The TDCP filter algorithm was detailed along with the different implementation variations. This filter was characterized and pitted against a conventional ambiguity estimation filter to evaluate their performance. It was shown that as the conditions degraded, the impact of TDCP measurements increased and even outperformed the simple ambiguity estimation filter. It was shown that the best TDCP filter configuration was with uncombined TDCPs and an innovation based test to detect cycle slips. Furthermore, the computational load of the ambiguity filter was compared with the TDCP filter which is 57% faster in deep urban conditions.

The goal was to develop a lightweight algorithm that could also perform in challenging conditions given the limited computational resources of the eHermes receiver. Hence, the proposed algorithm can be used for the receiver of 3D Aerospace as a baseline algorithm that does not overly consume computational resources presented in [Guillard et al., 2022]. Even though, the absolute accuracy of the proposed solution could be improved by adding more complex algorithms, it was observed that the main driver of the solution accuracy is the quality of code pseudoranges. Thus, Chapters 6 and 7 investigate a method to mitigate the impact of low quality code pseudoranges on the positioning solution.

CHAPTER 6

# CNN Based Multipath Detector

As the eHermes aims at working in urban environments, GNSS multipath is bound to affect the measurements. To answer this challenge, a study was done to develop a method that was as lightweight as possible while keeping a good multipath detection. Since the correlator values of the eHermes can be retrieved from the tracking loops, correlators were used for multipath detection. Indeed, these parameters had already shown promising results in signal quality monitoring and ML.

This chapter details the detection of multipath using a CNN model that resulted in [Guillard et al., 2023]. The model has been trained reconstructing the discriminator function to compute the tracking bias. Based on the magnitude of the tracking bias over a time interval, the data was labeled as affected by multipath or not. This method accounts for the time correlation of multipath and loop filters to detect multipath. Furthermore, the number of correlators used in the CNN to detect multipath is varied to quantify the benefit of using less correlators for the same goal. The computational gain of using CNNs with respect to the labeling method is also investigated and discussed.

## Chapter Contents


6.1 State of the Art Multipath Detection . . . 101
6.2 Data Labeling Classification . . . 102
6.3 Method and Scenarios . . . 107
6.3.1 Static Dataset . . . 107
6.3.2 Dynamic Dataset . . . 108
6.4 Signal Deformation . . . 109
6.5 Inputs . . . 109
6.5.1 Cross Correlation . . . 110
6.5.2 Correlator Configurations . . . 111
6.6 CNN Architecture . . . 112
6.7 Results . . . 113
6.7.1 Galileo E1-B . . . 115
6.7.2 GPS L1 C/A . . . 117
6.8 Conclusions . . . 120

## 6.1 State of the Art Multipath Detection

Multipath detector algorithms are not always machine learning based. Indeed, [Pagot et al., 2015], [Pagot et al., 2017], [Suzuki et al., 2020], and [Matera et al., 2019] all use different techniques to mitigate/exclude multipath reflections.

[Pagot et al., 2015] proposes three different techniques to investigate GNSS distortions including multipath. The first technique proposed was to observe chip transitions over a given time window to average out the noise affecting the samples. They also observe the correlation function to quantify the difference in correlator outputs. The last technique was to use the S-crossing of the discriminator to quantify the bias as detailed in (2.40).

[Pagot et al., 2017] used in-phase correlators to detect the deformation of the signal. To do so, they generated correlators with different delays and then compute metrics -ratio, difference ratio, sum ratio. These metrics are then compared to determined threshold to decide whether the signal is affected by distortions including multipath.

[Suzuki et al., 2020] use a horizontally rotating antenna to mitigate LOS and NLOS multipath errors. They mitigate LOS reflections by using the assumption that the relative phase between the LOS signal and its reflections change when the antenna is moving. They exclude NLOS signals by looking at the incidence angle of the signal as NLOS signals are assumed to arrive at different angles from LOS signals.

[Matera et al., 2019] tries to isolate and characterize the multipath residual from the pseudorange by using a referencing station to remove the atmospheric terms and receiver clock bias from a filtering process. They also use a fisheye camera to exclude satellites that are estimated as NLOS.

[Bétaille et al., 2013] use 3D maps to classify the potential multipath sources into categories and simplify the input map. They then use this information to detect NLOS satellites. Finally, these detected satellites can either be excluded from the position solution or corrected to be used in their positioning algorithm.

With the rise of machine learning and the difficulty of modeling multipath behavior, machine learning has been used towards multipath detection recently. A few research proposals are detailed below.

[Munin et al., 2020] and [Blais et al., 2022] generate synthetic images composed of the autocorrelation function for one tracking iteration. These images are obtained thanks to a multipath simulator which generates synthetic multipath with desired properties. Then, they are fed to a CNN using supervised learning to classify whether the ACF image is affected by multipath or not.

[Suzuki and Amano, 2021] implemented both an ANN and a SVM for multipath but to detect NLOS. They labeled their data using a fisheye camera and represented the satellite in the image to see whether its LOS signal was obstructed. They fed correlator outputs images to the ANN. For the SVM, their inputs were the number of peaks in the ACF, distribution of

the delay of the maximum correlation, and the signal strength as a function of the elevation angle of the satellite.

[Quan et al., 2018] details a multipath detection technique based on CNNs. Their inputs to the CNN are the time series of a linear combination of pseudoranges and carrier phase measurements as well as the CN0 over that time span. This is in the form of an image sent to the CNN. The approach was trained using labeled, simulated data and then tested to real data.

[Hsu, 2017] uses GNSS data collected in urban and open-sky conditions to classify the pseudorange in three categories: NLOS measurement, multipath, and no multipath. To do so, they used a SVM with the following features: received signal strength and its variation, the pseudorange residual, and the difference between the delta pseudorange and pseudorange rate. To label their data, they used raytracing to determine whether the data was multipath or NLOS and used the open-sky data collect as the no-multipath data.

The approach proposed in this chapter detects multipath on a time series of correlators at different correlation delays. Indeed, the mentioned articles, that use ML techniques, do not take the temporal correlation of consecutive correlator values induced by both multipath and the loop filters. Furthermore, they do not propose theoretical classification to train the networks. Here, the data labeling is based on finding the tracking bias of the S-curve of the discriminator similar to the method proposed in [Pagot et al., 2015]. The labeling method also takes into account the materials of the reflecting surfaces frequently encountered in urban conditions to set a threshold on what is considered multipath.

To justify the use of ML, the performance of the CNN model is compared to the performance achieved when using the theoretical data labeling method. An important parameter for the computational complexity and the performance of the theoretical labeling method is the number of correlators available at each tracking loop epochs. It was assumed that CNNs would be able to work with limited numbers of correlators. The computational gain obtained by using CNNs is also investigated and analysed. However, the computational cost of computing correlators is out of the scope of the presented study.

## 6.2 Data Labeling Classification

As presented in section 2.4.3.2, the tracking bias is computed by finding the delay corresponding to the zero crossing of the discriminator function. Then, multiplying the time bias by the speed of light, the tracking bias is obtained. Once this bias is passed through the low-pass filter of the tracking process, this filtered bias is what is propagated onto the pseudorange measurement. This process is given in (6.1) - (6.3).

$$\exists\ t_0 \in [-T_{obs}, T_{obs}] \text{ such that } D(t_0) = 0 \tag{6.1}$$

$$Tr_b = ct_0 \tag{6.2}$$

$$\rho_b = LPF(Tr_b) \tag{6.3}$$

Where:

- $T_{obs}$ is the maximum delay for which the discriminator is observed
- $c$ is the speed of light in m/s
- $Tr_b$ is the tracking bias in meters
- $LPF$ is the low-pass filter of the DLL modelled by a Butterworth Filter of order 2 with a bandwidth of $B_{DLL}$
- $\rho_b$ is the pseudorange bias induced from the tracking bias in meters

Figure 6.1 illustrates the tracking bias computed using (6.1)-(6.3) and the pseudorange error from multipath on synthetic data. Their magnitude are similar and almost always yielded the same multipath decision (99% similitude) when using the multipath criterion presented in this section.

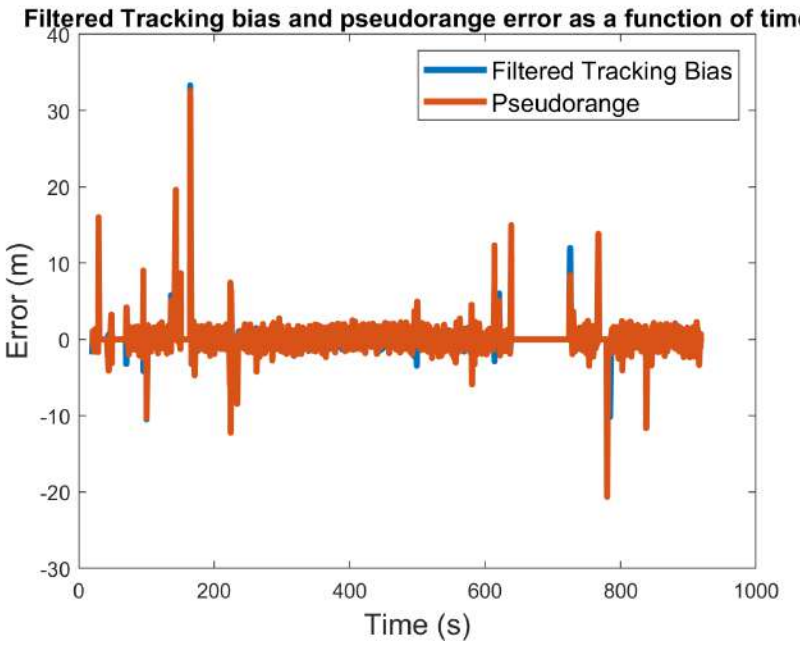


Figure 6.1: Multipath error on the pseudorange vs tracking bias error as a function of time on synthetic data.

In this method, up to 101 correlators were used to reconstruct the ACF of the PRN code. Only non-coherent discriminators, more specifically the EMLP presented in (2.32), will be studied as this is the most robust type of discriminator since it does not assume any PLL lock.

Since the DLL noise is assumed to follow a Gaussian law with a known distribution ($\epsilon_{Noise} \sim N(0, \sigma^2_{Noise})$), it implies that:

$$\epsilon_{noise} \leq 3 \times \sigma_{Noise} \text{ for } 99.7\% \text{ of the time} \tag{6.4}$$

The EMLP discriminator was used in this study, so the variance of the discriminator noise is equal to:

$$\sigma^2_{Noise} = L_c^2 \frac{B_{DLL} E_L}{2C/N_0} \left[ 1 + \frac{2}{(2 - E_L) N T_s C/N_0} \right] \tag{6.5}$$

The transient errors are ignored so the tracking bias is assumed to be only caused by multipath and noise which can then be expressed as [Braasch, 2017]:

$$\rho_b \approx \epsilon_{MP} + \epsilon_{Noise} \tag{6.6}$$

Hence, any bias exceeding $3 \times \sigma_{Noise}$ can be considered as caused by multipath. However, labeling every error exceeding this threshold would lead to a lot of measurements being classified as affected by multipath especially in urban and suburban environments. If excluded, this would only leave the most accurate measurements for positioning but would surely leave very few measurements for the positioning solution.

If too many satellites are excluded due to being labeled as affected by multipath, it may occur that:

- Not enough satellites to compute a position
- An increase in the dilution of precision values

On the other hand, if faulty measurements are not excluded from the positioning, then the positioning solution cannot be trusted.

To establish a new threshold, the idea was to find a value that would be small enough to not cause a large bias in the pseudorange but also large enough to not cause availability issues due to frequent multipath labeling.

As the largest availability versus multipath challenge occurs in degraded environments, the multipath sources can be identified. Indeed, multipath will often come from buildings, the road, or other vehicles.

When looking at attenuation factors of common surfaces at normal incidence, a threshold can be established based on materials that yield a tolerable multipath error. The values of these materials, for the L1 frequency, are given below and available in [Braasch, 2017].

| **Surface** | **Attenuation Factor (dB)** |
|---|---|
| Asphalt | -18.3 |
| Brick | -9.24 |
| Concrete | -7.87 |
| Glass | -7.51 |
| Tinted Glass | -0.446 |

Table 6.1: Reflection and attenuation factors for common urban surfaces at normal incidence on L1 frequency

By retrieving the attenuation factor from the table, the maximum amplitude of the multipath coming from this material can be known. Therefore, the multipath error enveloppe (MEE)

can be computed for a given material. The algorithm to compute the MEE is given in 2.4.3.2 The maximum induced multipath bias by a given material is given as:

$$b_{Material} = max(|M_{EE}(t_{MP}; E_L, \alpha_{Material})|) \tag{6.7}$$

Where:

- $M_{EE}$ is the multipath error enveloppe value for a given early late spacing and relative multipath amplitude factor
- $t_{MP}$ is the multipath delay in chips
- $b_{Material}$ is the maximum multipath error caused by a reflection from a given material
- $\alpha_{Material}$ is the relative multipath amplitude factor of a given material

For example, the MEE of brick is given in Figure 6.2 for an EL spacing of 0.04 chips for Galileo E1-B (left) and the MEE of brick given on Figure 6.2 (right) is for an EL spacing of 0.125 chips for GPS L1 C/A.

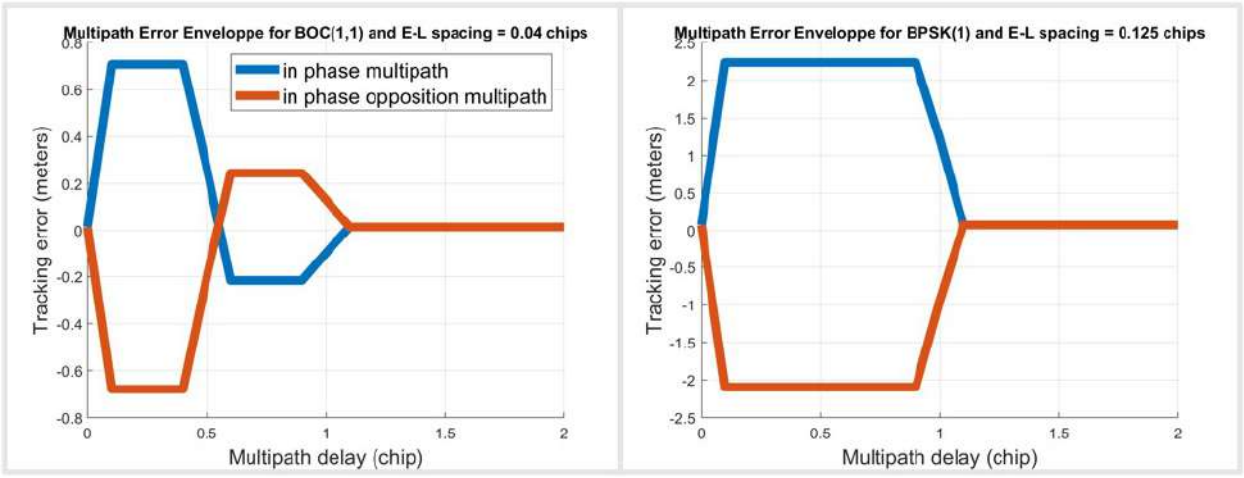


Figure 6.2: Multipath Error Enveloppe for a brick induced multipath for an E-L spacing of 0.04 chips on Galileo E1-B (left) and for an E-L spacing of 0.125 chips on GPS L1 C/A (right).

As assumed in [Zhao et al., 2013], the most commonly encountered materials in challenging environments will be: asphalt, glass, tinted glass, brick, and concrete. Brick is assumed to not be the most common building material. Since it has a high attenuation coefficient, it was decided that any multipath that has a lower amplitude than the ones generated from brick would be tolerated. However, this method still functions if the chosen material is different as its selection depends on the desired multipath detection sensitivity. Relating to Figure 6.2 (left), this means that any multipath, for a DLL with an EL spacing of 0.04 chips, that causes a bias smaller than 0.708 meters would be tolerated.

The early late spacing used on GPS L1 C/A was higher than for Galileo E1-B as a low EL spacing on L1 C/A led to a prolonged convergence of the tracking loops of the receiver. This change also induces an increase in the bias of (6.8) meaning that the threshold for multipath classification is higher on GPS L1 C/A than Galileo E1-B ($b_{Material} = 2.23$ meters).

As shown by [Prochniewicz and Grzymala, 2021], GPS L1 C/A is more affected by multipath than Galileo E1-B so this rise in threshold for L1 C/A makes sense to keep a high availability.

Thus, the threshold for multipath can be written as:

$$T = 3 \times \sigma_{Noise} + b_{Brick} \tag{6.8}$$

Where:

- $b_{Brick}$ is the maximum multipath error caused by a reflection from brick (2.23 and 0.708 meters for GPS L1 C/A and Galileo E1-B in this study)

This gives the following classification:

$$\text{Decision} = \begin{cases} |\rho_b| \geq T \rightarrow \text{Multipath} \\ |\rho_b| < T \rightarrow \text{Not Multipath} \end{cases} \tag{6.9}$$

As shown in [Vergara et al., 2009], the DLL noise increases due to the presence of multipath. However, The increase of the noise is assumed to be accounted for by the addition of $b_{brick}$.

The decision given in (6.9) will be the method used to classify multipath and train to the convolutional neural network presented in section 6.6.

By reconstructing the ACF of each tracking epoch with multiple correlators, the discriminator function can be obtained with great accuracy. However, if the discriminator function is subsampled, the tracking error has to be linearly interpolated at the zero crossing. The further the correlators are spaced from one another, the less accurate the interpolation is. This phenomenon is depicted in red on Figure 6.3.

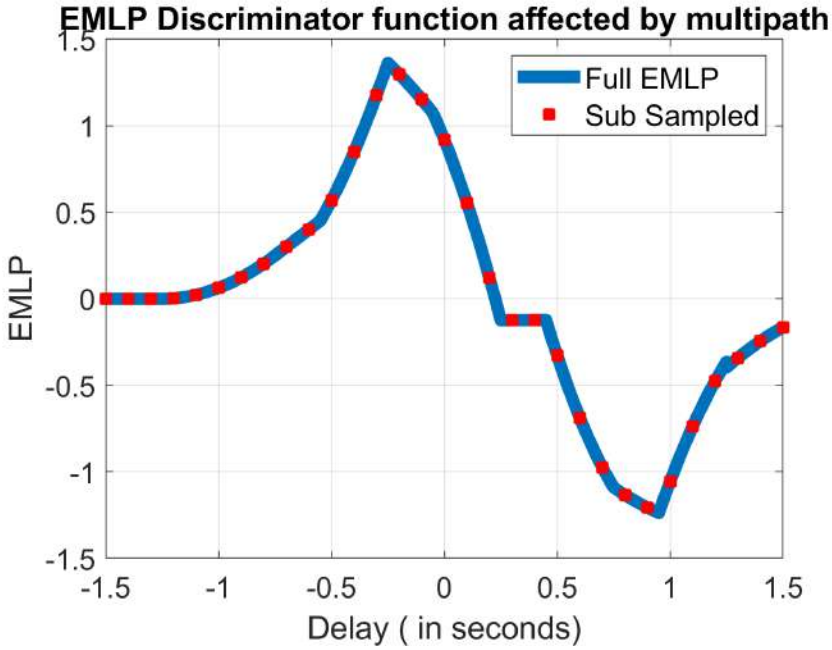


Figure 6.3: EMLP discriminator affected by some multipath fully sampled (blue) and under sampled (red).

When the number of correlators is insufficient or does not cover enough of the ACF, the delay may not be computed due to the lack of zero crossing. In this study, when this occurs (only with low correlator resolutions), the delay is assumed to be zero to not reset the low pass filter.

The discriminator function can have several zero crossings. The zero crossing that is the closest to the prompt index and with a negative slope is chosen as the stable lock point from which to compute the tracking bias - since the EMLP discriminator is used. On the presented dataset, when this occurs, the second zero crossing is much farther in terms of delay than the one chosen as the stable point with respect to the prompt correlator.

## 6.3 Method and Scenarios

The static data was used to validate that the filtered tracking bias was a valid approach as a theoretical classification algorithm. This dataset was obtained in open-sky on the L1 frequency band on GPS and Galileo with a 200 MHz sampling frequency.

The dynamic data were collected in all challenging conditions (sub-urban and deep urban). This was to obtain multipath errors to detect. The dataset is on the L1 Frequency band on GPS and Galileo with a 200 MHz sampling frequency. This large sampling frequency was used as it enabled a better reconstruction of the ACF over the span of the collect. An IFEN SX3 was used as the GNSS receiver [IFEN, 2023]. The IFEN SX3 is a GNSS receiver that is capable of tracking all constellations and frequencies. It is very modular and can record raw data to then replay with different configurations. Indeed, the number and placement of correlators are selectable, along with the loop filter bandwidths, E-L spacing, etc.

### 6.3.1 Static Dataset

This data was collected at ENAC, in Toulouse France, with a static Leica AR20 choke ring antenna. It is not possible to fully remove multipath reflections or avoid any NLOS signals. However, by placing the antenna at the top of a building in open-sky condition, its effect was mitigated as best as possible. The goal of the dataset was to check that the computed pseudorange bias ($\rho_b$) was almost always under the $3 \times \sigma_{Noise}$ threshold. This would indicate that, in optimal conditions, where multipath reflections should rarely occur, this method of multipath classification is valid. The distribution of the pseudorange bias in this scenario is given in Figures 6.4 and 6.5.

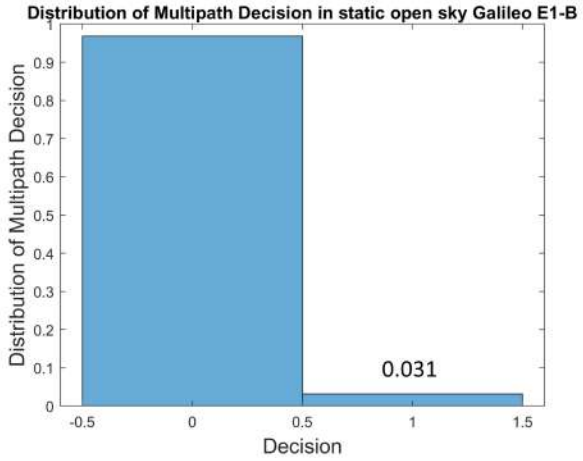


Figure 6.4: Static Distribution of Multipath Decision for Galileo E1-B.

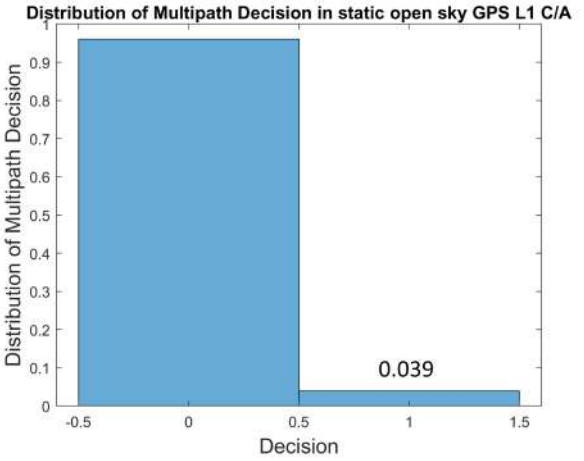


Figure 6.5: Static Distribution of Multipath Decision for GPS L1 C/A.

As the number of detected multipath has the right order of magnitude (3% observed, vs 1% expected from theory), it can be concluded that the classification method is valid. Furthermore, most of the multipath detected in static conditions occurs while the loops are converging at the beginning of the dataset.

The IFEN receiver was run with the same parameters as for the dynamic scenario, given in Table 6.2. This data was used to verify that most, if not all, of the multipath was under the multipath classification presented in section 6.2.

### 6.3.2 Dynamic Dataset

This dataset was obtained with a car. The collect was in the city center of Toulouse, France, in a mix of urban and sub-urban environments for a total duration of approximately 40 minutes. The neural network had more than 4000 images to be trained upon, for each signal. There were more than 4 million correlator outputs used per signal, and the satellite geometry was diverse. Having more multipath scenarios would likely be beneficial for the networks, but the number of images and correlators was deemed sufficient to validate the proof of concept.

The assumptions made on the material types as most buildings are made out of either concrete or bricks with glass windows and the road in asphalt. The IFEN SX3 was connected to a Novatel GPS 704-X pinwheel antenna. The data was replayed in post-processing to retrieve the full correlators for each epoch as it cannot be done in real-time with this large number of correlators. The main parameters for each signal is given in Table 6.2.

| | GPS L1 C/A | Galileo E1-B |
|---|---|---|
| E-L Spacing (chips) | 0.125 | 0.04 |
| DLL Bandwidth (Hz) | 1 | 1 |
| Integration Time (ms) | 1 | 4 |
| Number of correlators | 101 | 101 |
| Correlator Sampling Frequency (MHz) | 200 | 200 |

Table 6.2: IFEN SX3 Settings for GPS L1 C/A and Galileo E1-B.

## 6.4 Signal Deformation

When first plotting the tracking biases and its filtered version, it was noticed that they were not centered around zero. These biases were slightly offset by a different value depending on the satellite number and constellation. Because of this, setting a threshold would not be possible according to the decision given in (6.9). This tracking bias is caused by the digital and/or analog hardware distortions [Song et al., 2020]. [Pagot et al., 2015] used the S-crossing of the discriminator amongst other techniques to quantify the bias. [Pagot et al., 2017] used in-phase correlators to detect the deformation of the signal. In [Fan Liu, 2006], their solution estimated the satellites and receiver hardware biases by estimating the bias caused on each correlator. However, it required several identical receivers to do so, which was not possible in this study. Thus, the proposed solution here is a modified version from [Fan Liu, 2006] and estimates the satellite and receiver biases as one bias altogether.

This method assumes that the hardware and receiver biases are constant over the duration of the dataset. They are also assumed to be the same over other data collects even though they will vary according to temperature and the incidence angle. These supposedly small variations are assumed to be accounted for in the standard deviation of the noise, and so in the threshold of (6.8). The method is presented in Figure 6.6.

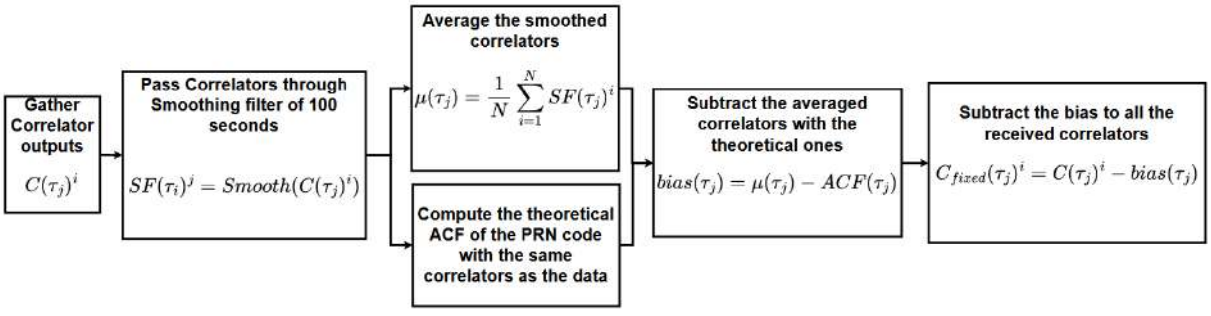


Figure 6.6: PRN Deformation Correction Algorithm.

## 6.5 Inputs

The images that are input to the CNN are the correlator values as a function of the chip delay (on the y axis) with respect to the prompt correlator and time (on the x axis) to yield 2D images. As detailed in section 6.2, the filtered tracking bias is used to label data as

multipath or not. Figure 6.7 represents the tracking bias and its filtered version for a given satellite on Galileo E1-B (left) while an example, from the training dataset, of both not multipath and multipath are given on Figure 6.7 (right) for the 101@200 correlator variation (colorized).

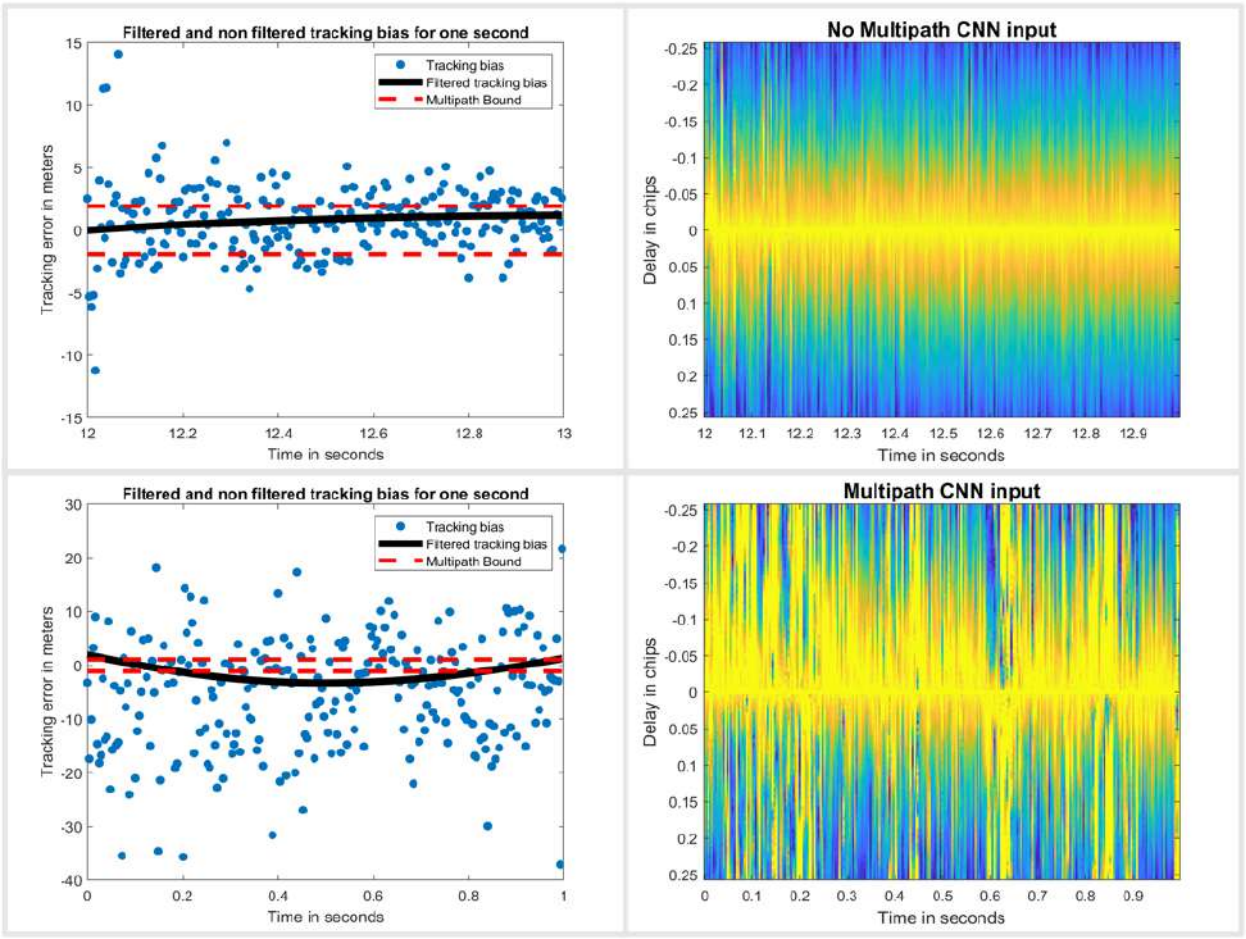


Figure 6.7: Filtered (black) and non filtered (blue) tracking bias with its multipath bounds (red) for Galileo E1-B on PRN 3 (left) and their corresponding CNN inputs (right).

To reduce the computational cost of using multiple correlators and evaluate the CNN performance over the number of correlators, several set of images were generated with different correlator sampling frequencies. The CNN architecture and labeled data remain identical no matter the correlator configuration. However, each correlator configuration induces a training period with the corresponding images of the chosen correlator configuration. The different configurations are given in 6.5.2.

### 6.5.1 Cross Correlation

CNNs can also be affected by overfitting when the input data is too correlated. Hence, special care was taken to ensure that the input images were not correlated with one another. This correlation can come from the DLL filter and is dependent on the bandwidth. Indeed, the filtered pseudorange biases depend on the previous ones. To make sure that they were generally not correlated with one another, the cross correlation factor - given in (6.10) - was

computed between the filtered tracking bias outputs.

$$\rho_{x,y} = \frac{Cov(x,y)}{\sqrt{\sigma_x^2 \sigma_y^2}} \tag{6.10}$$

The cross correlation factor for a given Galileo satellite on E1-B is given in Figure 6.8. This figure represents the cross correlation factor of a sample of the filtered tracking bias over a second with a delayed the filtered tracking bias over a second. After one second, the cross correlation is small; therefore, images spaced out of one second each can be considered as uncorrelated.

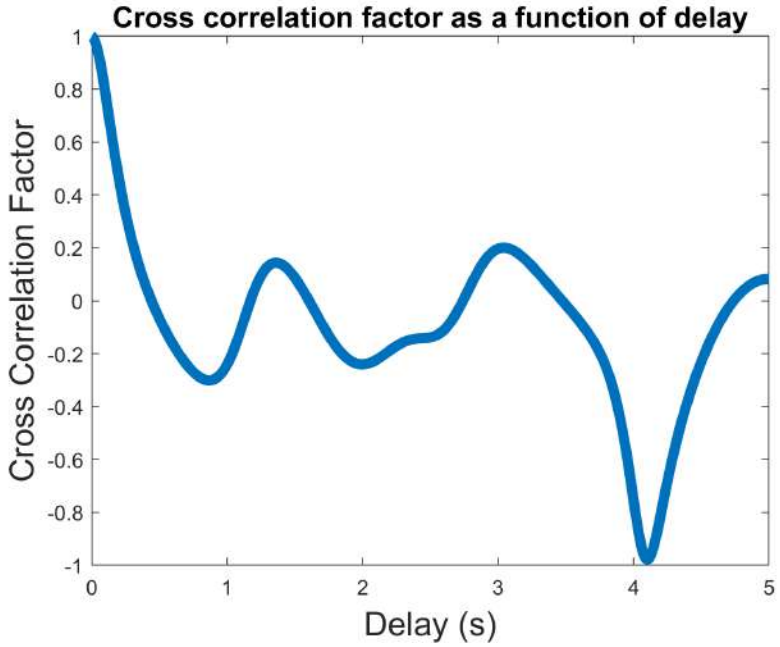


Figure 6.8: Cross correlation factor of a sample of the filtered tracking bias over 5 seconds.

### 6.5.2 Correlator Configurations

This section presents the different correlator configurations for Galileo E1-B, and the ones for GPS L1 C/A. In order to investigate the benefits of using correlator images to detect multipath, the number of correlators used in images was varied. To obtain an accurate data labeling, the pseudorange bias has been computed with the most amount of correlators – 101 at 200 MHz. In total, there are 7 correlator variations presented in Table 6.3.

| Configuration Name | Number of correlators | Correlator Sampling Frequency (MHz) |
|---|---|---|
| 101@200 | 101 | 200 |
| 51@100 | 51 | 100 |
| 51@200 | 51 | 200 |
| 51@200&100 | 17 & 34 | 200 & 100 |
| 25@50 | 25 | 50 |
| 25@200 | 25 | 200 |
| 25@200&50 | 9 & 16 | 200 & 50 |

Table 6.3: Correlator configurations for GPS L1 C/A and Galileo E1-B.

Figure 6.9 illustrates the 101@200 correlators configuration along with the three configurations using 25 correlators by comparing the range of values obtained with the different variations versus the full theoretical autocorrelation function for GPS L1 C/A.

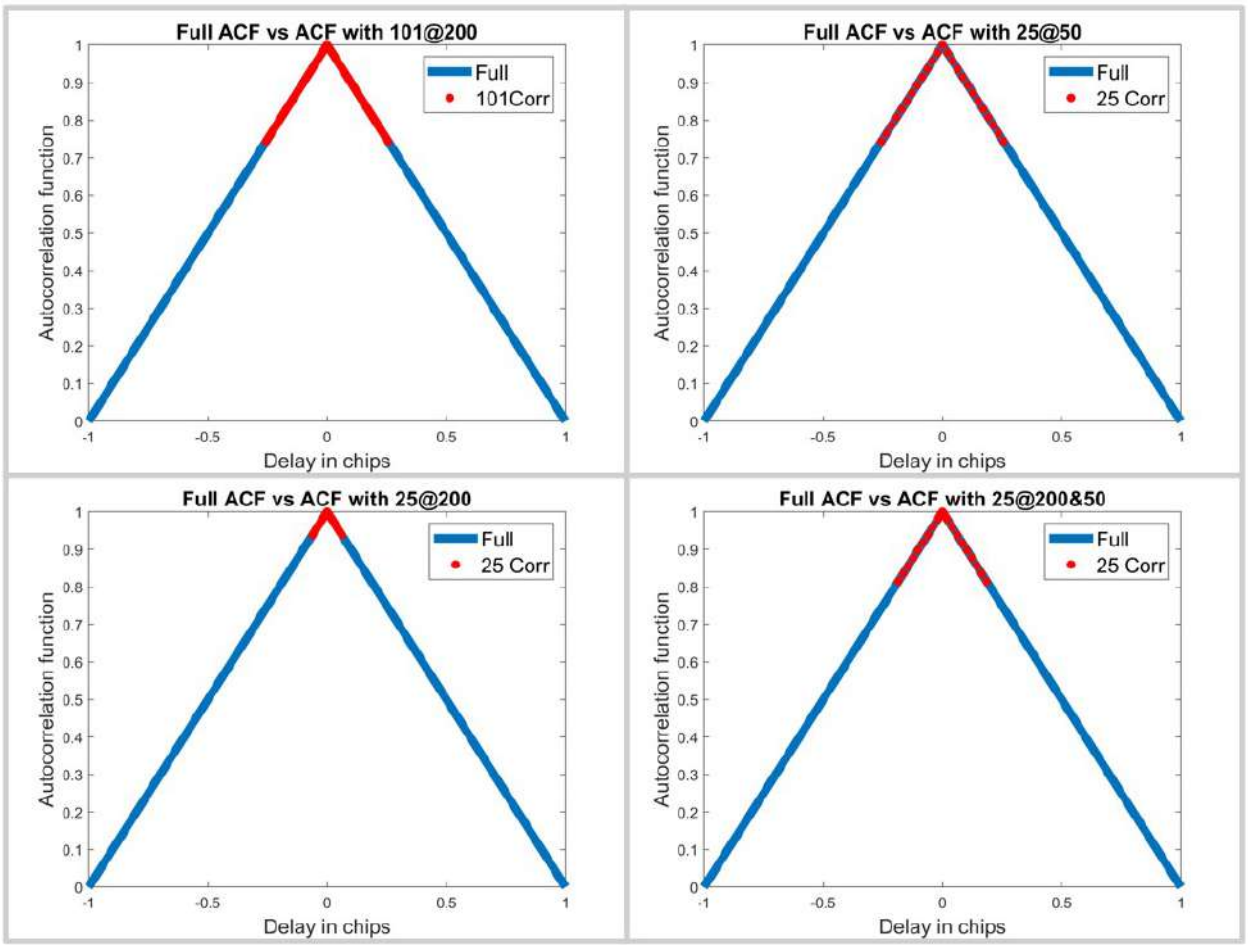


Figure 6.9: Illustration of some correlator variations for GPS L1 C/A.

## 6.6 CNN Architecture

To construct the CNN for multipath mitigation, Tensorflow [Abadi et al., 2016] and Keras [Chollet, 2015] were used as the backbone. Tensorflow is an open source deep learning library

owned by Google that enables one to develop a machine learning algorithm from end to end. Keras is also an open source machine learning library that runs on top of Tensorflow and allows one to easily build networks and train models. The images are grey-level to reduce the number of parameters to be trained, with a 256x256 pixel size. As there are at most 101 correlators used, the images were upscaled on the correlator delay axis. On the temporal axis, the slightly upscaled for E1-B (250 values per correlator position per second) while they were downscaled for GPS L1 C/A (1000 values per correlator position per second). The images are then passed to the CNN model described below. This model is a simple one; it was observed that a complex CNN was not needed to successfully detect multipath from the correlator images. Indeed, with respect to [Munin et al., 2020] who also uses CNN for multipath detection, the presented network uses one less convolutional layer making it more lightweight. Furthermore, special attention was put on overfitting by implementing dropout and L1 and L2 regularization while [Munin et al., 2020] and [Quan et al., 2018] do not mention this threat. As multipath and the tracking loop filters can introduce correlation between data inputs, overfitting can be even more prominent when using post-tracking outputs.

The first layer is a convolutional layer made of 20 filters of size 3x3 with a stride of 1 computed. This layer uses the ReLU activation function. Then this layer is followed by a max pooling layer of size 2x2 with a stride of 2x2. The outputs of the 20 filter maps are flattened (passed into 1D) to a layer of 1 neurons with a sigmoid activation function which yields the result for the image:

- $0 \rightarrow$ Not Multipath
- $1 \rightarrow$ Multipath

This layer is implemented with a dropout rate of 0.5 meaning that each neuron connection has a 50% chance of being ignored at each epoch. This layer is also regularized with the elastic net method given in (4.18) (L1 and L2 regularization with $\lambda = 0.001$ for both). The loss function chosen in this architecture is the binary cross entropy. The batch size was of 64.

The full architecture is given in Figure 6.10.

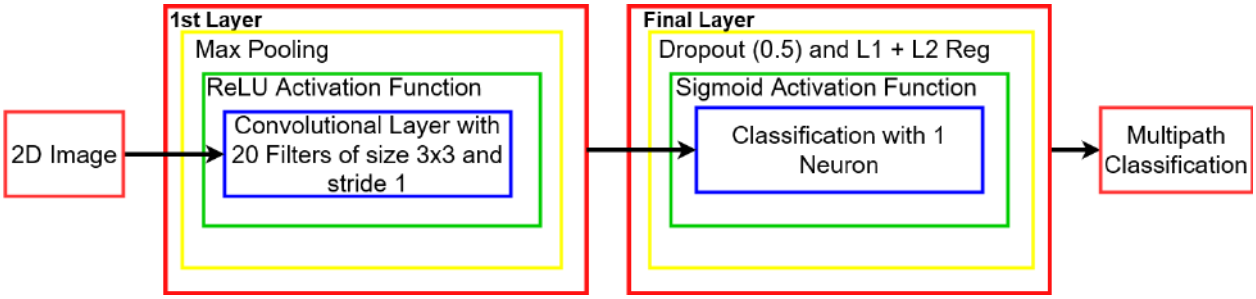


Figure 6.10: CNN Architecture for Multipath Detection.

## 6.7 Results

To train a binary classifier, the two classes – here multipath and no multipath – need to be equally represented. Otherwise this can lead to the network over predicting the dominant

class. This leads to a class imbalance in the datasets. There exist several countermeasures for this condition as given in [Buda et al., 2017]. The chosen method is to down-sample the dominant class. Indeed, the no multipath class will be reduced in size to roughly match the multipath one. To still train all images, the no multipath images will be divided in several datasets and the final results presented in the following sections will be the average of all the trained models.

As discussed in Chapter 4, overfitting makes the model overly optimistic on training data and perform worse on test and validation data. Thus, it is important to make sure that the proposed model is not overfitting. A good metric to check overfitting is the difference of the training loss with the validation loss of the binary cross entropy function. The accuracy and loss values of the 101@200 configuration on Galileo E1-B are presented in Figure 6.11. As the difference in loss value is very small at the end of the training epochs (∼0.05), this displays the fact that the model is not affected by overfitting.

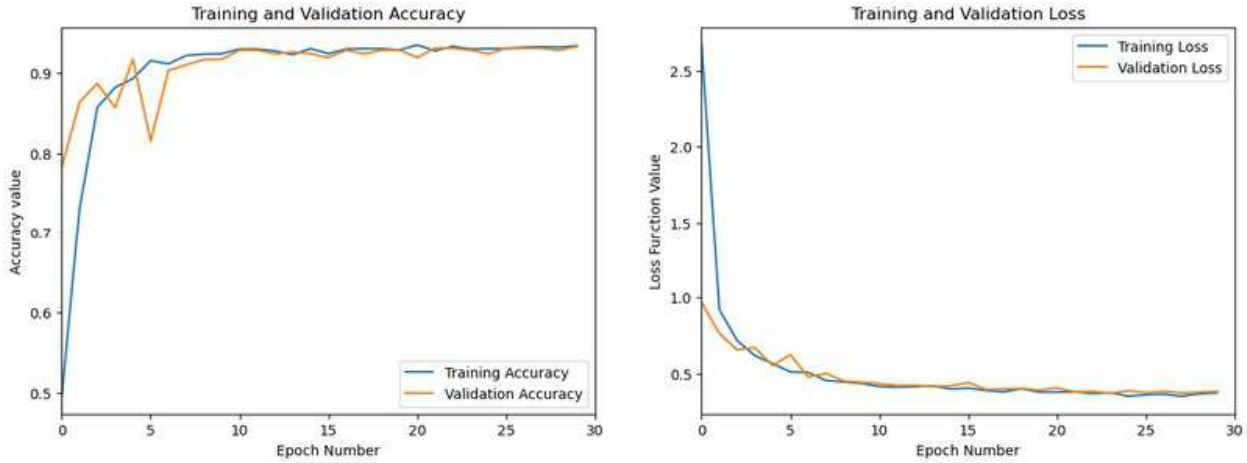


Figure 6.11: Accuracy and loss values over training iterations for training and validation datasets.

To analyze the results of a machine learning model, several metrics are often given: accuracy, precision, recall, and F score. The accuracy represents the proportion of correct classification the model made. Precision represents the proportion of correct multipath classification. Recall represents how well the model can detect multipath. The F score is the harmonic mean of precision and recall, it better highlights the impact of false negatives and false positive than accuracy [Powers, 2020]. The binary classification results can be represented as shown by the Table 6.4.

| **Actual** / **Truth** | Positive | Negative |
|---|---|---|
| Positive | True Positive | False Negative |
| Negative | False Positive | True Negative |

Table 6.4: Confusion Matrix

The accuracy, precision, recall, and F score are equal to [Dalianis, 2018]:

$$\begin{aligned} A_{cc} &= \frac{TP + TN}{TP + TN + FP + FN} \\ P_{rec} &= \frac{TP}{TP + FP} \\ R_{ec} &= \frac{TP}{TP + FN} \\ F_{score} &= \frac{2P_{rec} \times R_{ec}}{P_{rec} + R_{ec}} \end{aligned} \tag{6.11}$$

Where:

- $TP$ is a true positive
- $FP$ is a false positive
- $TN$ is a true negative
- $FN$ is a false negative

To better highlight the benefit of using CNNs for multipath prediction, these metrics will also be computed with the tracking bias method when the number of correlators is reduced. This means that the pseudorange bias will be estimated for each correlator configuration. Their accuracy, precision, recall, and F score for that configuration with respect to the multipath classification done with 101 correlators at 200 MHz will also be computed.

The differences between the time taken to predict the presence of multipath with the use of CNN with respect to the tracking bias method will also be presented. To have a fair comparison, both methods are compared when using the same number of correlators and are evaluated in the same coding language and on the same hardware.

### 6.7.1 Galileo E1-B

In Figure 6.12, the accuracy, precision, recall, and F score are given for all correlator combinations for both CNN and tracking bias based multipath classification. The tracking bias results are in blue while the CNN results are in orange. The CNN results presented here are from the test dataset; the data split was of 80% for the training data and of 20% for the test data. As there is just one model and the hyperparameters of the model are not varied, no validation data was used to generate the results presented.

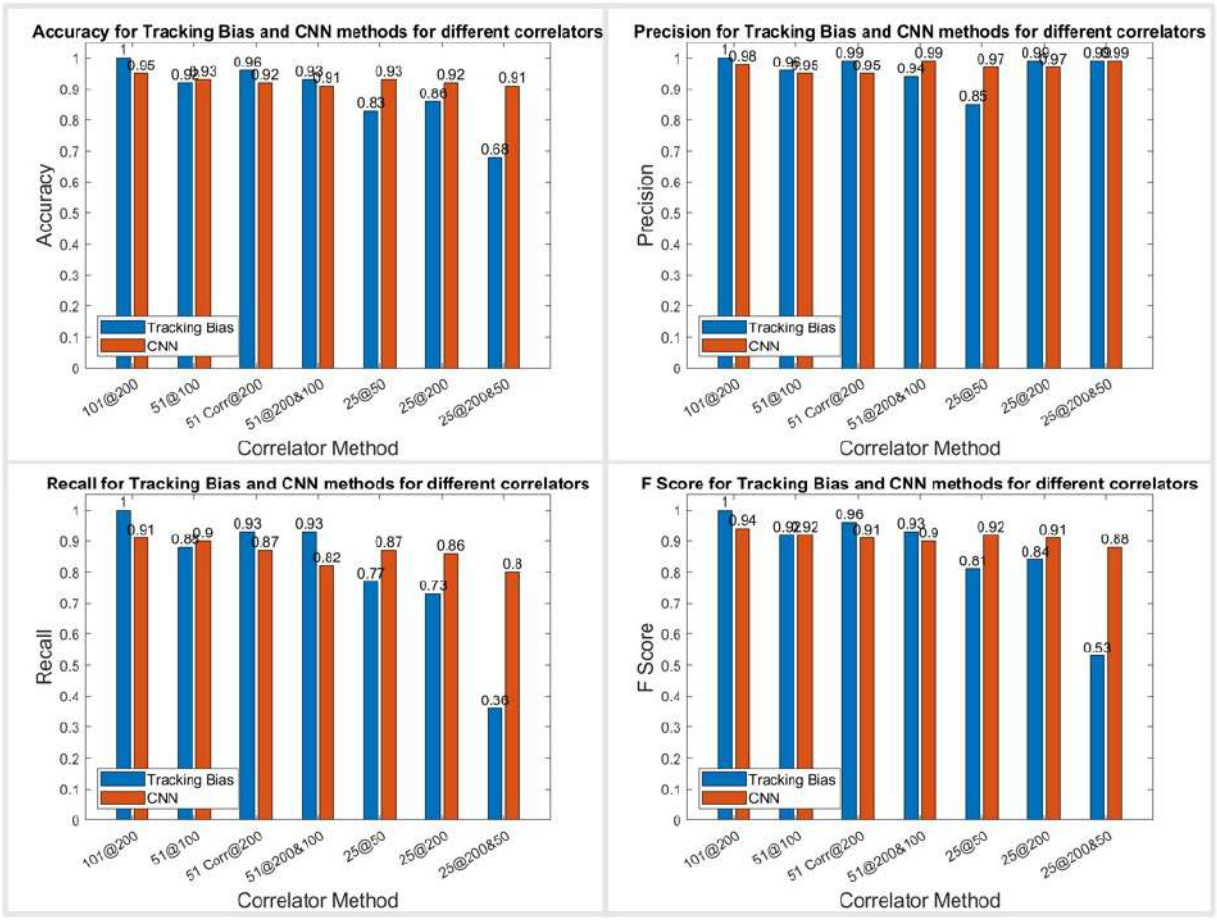


Figure 6.12: E1 Metrics of the CNN.

| | 101@200 | 51@100 | 51@200 | 51@200&100 | 25@50 | 25@200 | 25@200&50 |
|---|---|---|---|---|---|---|---|
| CNN | 0.17 | 0.18 | 0.17 | 0.18 | 0.16 | 0.16 | 0.16 |
| TB | 1 | 0.51 | 0.80 | 0.70 | 0.32 | 0.67 | 0.41 |

Table 6.5: Normalized Mean Execution time for all configurations.

#### 6.7.1.1 CNN vs Tracking Bias

The tracking bias method, when using 101@200, is the configuration used for determining the truth; so its metrics are all equal to one. When using 51 correlators – no matter the correlator sampling frequency – the tracking bias method performs better or at least equivalently to CNNs as conveyed by the higher F score and accuracy results. But when only 25 correlators are used, both F scores and accuracies plummet for the tracking bias approach. When the number of correlators is decreased, the tracking bias metrics quickly degrade in performance. This is shown by the 11%, 7.5%, and 36% F score differences between the 25@50, 25@200, and 25@200&50 configurations respectively.

To compute the tracking bias, the reconstructed discriminator function linearly interpolates the delay when there is a zero crossing. By reducing the number of correlators and

the correlator sampling frequency, the linear interpolation between two delays is greater and leads to a higher uncertainty on the resulting tracking bias based on the effect of multipath and noise at this time. On the other hand, reducing the number of correlators while maintaining a high correlator sampling frequency leads to a shortened discriminator function where the zero crossing may not be present. In that case, the tracking bias cannot be estimated. The metrics highlight that, when the number of correlators is reduced, the CNN method outperforms the tracking bias one.

#### 6.7.1.2 Computational load

Table 6.5 displays the time taken by each method, normalized by the most time consuming method. This table illustrates that all CNN based configurations are two to six times faster than their tracking bias counterpart. This results from the tracking bias being computed 250 times per second since the integration period used here was of 4 milliseconds. On the other hand, as images are generated every second to limit cross-correlation, the CNN only predicts once per second explaining the gain in time.

#### 6.7.1.3 CNN Correlator Configurations

The 101@200 variation is the benchmark configuration for the CNN approach, but the overall performance of other correlator configurations, when looking at accuracy and the F score, is very similar. This indicates that, even with a lot less correlators, the CNN used to detect multipath is still effective while using less computational power. This is depicted when the correlator sampling frequency is lowered from 200 MHz to 50 MHz, and the number of correlators is reduced by a factor of 4 for only a 2.8% reduction on the accuracy and 3% reduction on the F score. Thus, after this model is trained, the sampling frequency and number of correlators could be reduced to alleviate some of the computational load while keeping a high correct classification rate.

Despite the very high precision of the CNN from both correlator configurations of 51@200&100 and 25@200&50, their recall is lower with respect to the other variations. It illustrates that the two configurations are the worst at detecting multipath. Even though, this model is more precision oriented, the recall should not be fully sacrificed at the expense of precision. This is also reflected in the F scores of these variations as they have the worst F scores.

The two CNN variations of 25@200 and 51@200 perform worse than 25@50 and 51@100 by and 0.8% and 2% on their F score respectively while outperforming the two configurations using 51@200&100 and 25@200&50. Thus, the multipath effect is more observable on a larger delay range of correlator outputs.

### 6.7.2 GPS L1 C/A

In Figure 6.13, the accuracy, precision, recall, and F score are given for all correlator combinations for both CNN and tracking bias based multipath classification. The tracking bias results are in blue while the CNN results are in orange.

The data split used here was the same as Galileo E1-B, 80% for the training data and 20% for the test data and no validation data was used.

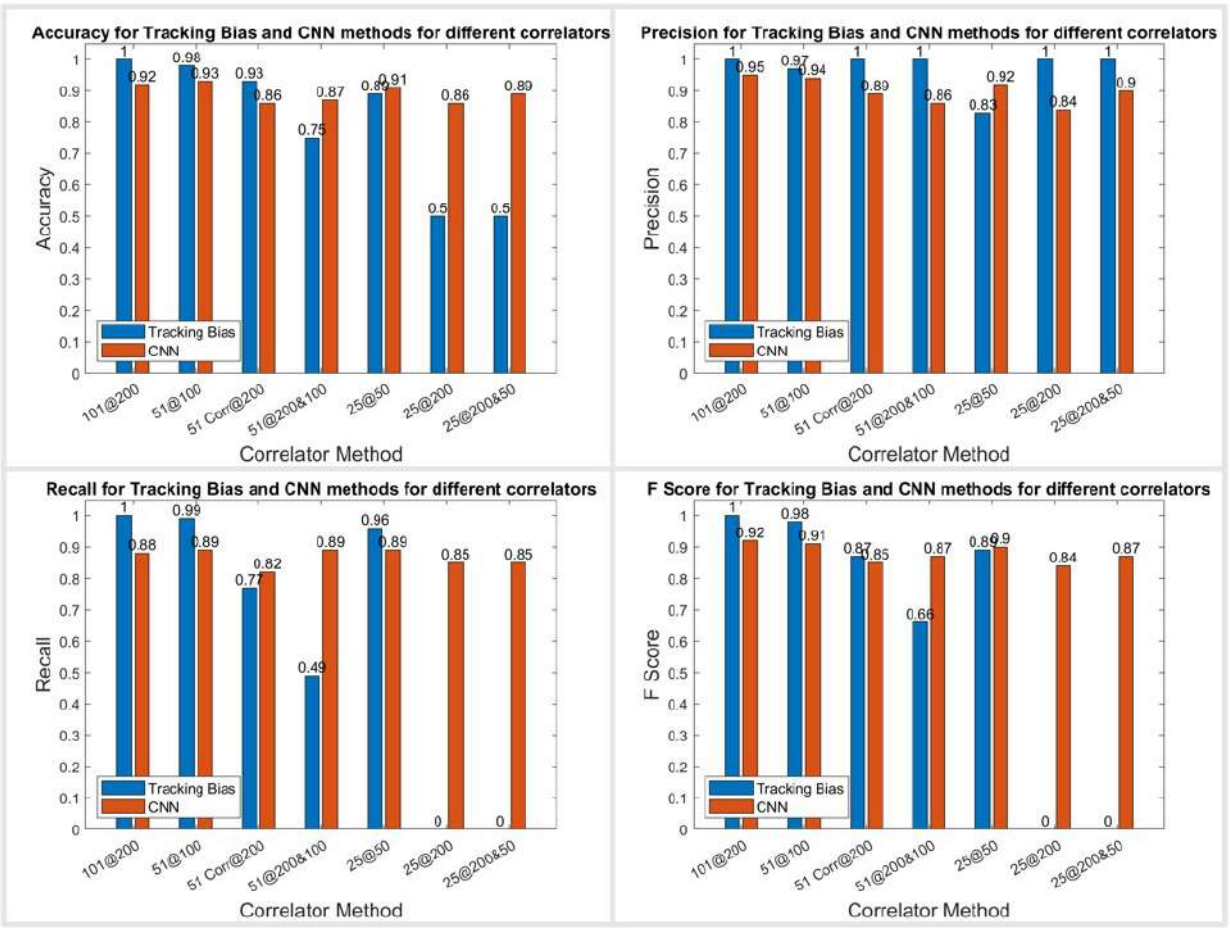


Figure 6.13: L1 Metrics of the CNN.

| | 101@200 | 51@100 | 51@200 | 51@200&100 | 25@50 | 25@200 | 25@200&50 |
|---|---|---|---|---|---|---|---|
| CNN | 0.0085 | 0.0090 | 0.0091 | 0.0092 | 0.0093 | 0.0093 | 0.0093 |
| TB | 1 | 0.40 | 0.29 | 0.27 | 0.20 | 0.047 | 0.073 |

Table 6.6: Mean Execution time for all configurations.

#### 6.7.2.1 CNN vs Tracking Bias

Like for Galileo E1-B, the tracking bias method using 101 correlators is used for determining the truth explaining the perfect metrics. When using 51@100 the tracking bias configuration performs better as the range of the delay of the ACF is kept while maintaining a narrow enough linear interpolation. For 51@200, the F Score is higher of 1.8% than for its CNN counterpart. Despite having a perfect precision, this variation has a much lower recall (77%) indicating that it is much worse at detecting multipath. This is further highlighted by the tracking bias results using 51@200&100. Indeed, the recall for this configuration is of 49.1% but has a perfect precision score.

When decreasing the number of correlators to 25, this becomes even more apparent as the 25@200 or 25@200&50 have recall scores of 0 meaning that these configurations classify every

measurement as not affected by multipath. This can be explained by the fact that the early late spacing is higher for GPS L1 C/A. Indeed, when the early late spacing is increased, the linear region of the discriminator function is also increased. On the downside, the number of correlators needed to cover the linear region to find the zero crossing of the discriminator function increases. The zero crossing can even be shifted further in the delay domain due to multipath and noise. Hence, the two variations using 25@200 and 25@200&50 do not cover enough of the ACF to get a correct estimate of the tracking bias.

Moreover, when using 25@50, the F score of the CNN based approach is 1.6% higher which is due to its much higher precision with respect to the tracking based approach despite its lower recall. Just like for Galileo E1-B, this conveys the fact that when the number of correlators is reduced, the CNN is a more viable solution to predict the presence of multipath than the tracking bias method.

#### 6.7.2.2 Computational load

Just like for Galileo E1-B, the CNNs are much faster than the tracking bias method. Indeed, the CNN is at 21 times faster on the 25@50 configuration as shown by Table 6.6. It is even more pronounced on GPS L1 C/A as the tracking bias is computed 1000 times per second. The time taken by the tracking bias method also depends on the number of correlators used whereas the CNN method execution time is almost identical. This can be explained by the fact that the CNN architecture is the same regardless of the correlator configuration. Therefore, the only change between correlator variations are the values of the filters and weights of the neurons. Since the time taken to generate an image is very low, this does not make a computational difference between the correlator inputs to the CNN.

#### 6.7.2.3 CNN Correlator Configurations

From all the correlator variations,he CNN configurations using less correlators at a reduced correlator sampling frequency (51@100 and 25@50) have only slightly lower metrics than the CNN benchmark variation of 101@200 with respect to other correlator configurations. This can be explained by the higher early-late spacing chosen for this signal with respect to E1-B. Hence, the distortion caused by the multipath on the tracking bias is larger for higher E-L spacing [Van Dierendonck et al., 1992]. So, maintaining the same delay range on the ACF is more important with larger early late spacing than having narrow correlators. Indeed, the decrease in F score from 101@200 to 51@100 is of 0.33 % and 1.14% from 101@200 to 25@50.

Furthermore, the two CNN configurations using 51@200 and 25@200 perform the worst. This is due to the fact that both of these variations do not display enough of the ACF – especially the 25 correlators configurations that display just enough the correlators between the early and late correlators. Hence, these variations combine too little information on the ACF while still overcrowding the image with narrow correlators that do not add enough benefit to the classification.

For the CNN based approach, the 51@200&100 to 25@200&50 variations have a performance that is in between the two previous configurations. As this configuration still requires a high sampling frequency of the signal yet does not improve the classification, a lower

correlator sampling frequency for correlators is favored for both computational and metric performances.

## 6.8 Conclusions

This chapter proposed a theoretical classification of multipath using GNSS tracking correlator outputs. This classification has been used to label data to train a convolutional neural network. Then, this CNN has been used to predict whether multipath was present or not on correlator outputs from GPS L1 C/A and Galileo E1-B leading to a publication [Guillard et al., 2023]. This CNN has been developed to be low in complexity so that it can be trained with a low amount of data while limiting the risk of under or overfitting.

The model uses tracking correlators values as a function of delay (with respect to the prompt correlator) and time in the form of images of 256x256 pixels. The computational cost of computing correlators was not investigated but the inputs of the model are lightweight, making the predictions yielded by the CNN to be swiftly computed. This is illustrated by the fact that it is around 21 times faster to compute than the tracking bias for GPS L1 C/A and two to six times faster for Galileo E1-B, when using 25@50. The metrics between the CNN approach and the tracking based approach were also compared, and the CNN performed better as the number of correlators was lowered. For example, when using 25@50, the F score was of 91.8% for the CNN and of 80.8% for the tracking bias on Galileo E1-B. The same conclusion applies to GPS L1 C/A as the F score was of 88.9% for the tracking bias and 90.5% for the CNN with 25@50.

Several variations have been tested in the CNN to test how the CNN performs when reducing the number of correlators to generate the input images. The favored configuration is to decrease the number of correlators along with the correlator sampling frequency. This result in an observation of the correlation function over the same delay range, but with a lower resolution. Indeed, it was shown that by reducing the number of correlators from 101 to 25 and reducing the resolution by a factor of 4, the F score only decreased from 94.7% to 91.8% on Galileo E1-B and from 91.6% to 90.5% on GPS L1 C/A. This highlights the robustness of this method to detect multipath and then exclude the corrupted measurement from the positioning solution.

Indeed, as mentioned earlier, recall is how well the model detects multipath, thus excluding it. So, if a model correctly detects all the multipath, the confidence in the positioning solution would be higher. On the other hand, solutions that satisfy a certain precision requirement would improve the solution availability, i.e having enough GNSS observations to have a full-rank system that can yield a GNSS Position Velocity Time (PVT) solution. As precision is how well the model classifies multipath, if there are very few false positives, then the model rarely falsely excludes useful measurements.

In an accurate model, precision and recall are inversely correlated with one another [Davis and Goadrich, 2006], meaning that increasing one would decrease the other. To increase the recall of the model, the threshold of the sigmoid function of the last neuron could be

changed. The proposed decision in this study is:

$$y = sigmoid(x)$$

$$\begin{cases} y \geq 0.5 \rightarrow \text{Multipath} \\ y < 0.5 \rightarrow \text{No Multipath} \end{cases} \tag{6.12}$$

By lowering the threshold to below 0.5, it would increase the recall at the expense of precision: it would only classify no multipath for very low values of $y$. In this research, availability was deemed more important than integrity as shown by the models having higher precision than recall. This is because additional integrity algorithms can be added on top of this solution such as fault detection and exclusion [Rakipi et al., 2015]. By orienting the model towards excluding all multipath, this could lead to less satellites than needed for a position computation.

The proposed CNN model could be implemented in the eHermes thanks to its lightweight nature, but further investigation needs to be done in the reduction of correlators. Indeed, using 25 correlators per satellite is still resource consuming. Therefore, the number of correlators needed for a given accuracy is still to evaluate.

The described model is re-used in Chapter 7 to detect pseudoranges that are affected by multipath and apply the proposed algorithm based on the combination of robust estimators and WLSE.

CHAPTER 7

# Positioning Performance of Adapted Robust Estimator

Chapter 6 detailed a method to detect multipath using CNNs. However, detecting multipath by itself does not lead to an improved positioning solution. To do so, the multipath affected observable(s) must either be excluded or their effect mitigated in the positioning solution. The approach taken in this PhD was to mitigate their effect but to keep the degraded observable in the positioning solution. Indeed, in challenging GNSS environments, multipath is unavoidable. Hence, excluding multipath-riddled measurements could lead to availability issues of the GNSS position.

It was seen in Chapter 5 that the positioning accuracy of the tightly coupled INS/GNSS EKF is dominated by the code pseudorange accuracy. Furthermore, multipath has the highest effect on code pseudoranges. Therefore, a study of snapshot positioning only using code pseudoranges is proposed here to study the mitigation of the multipath effect.

This chapter proposes a fusion of a robust estimator and a classical WLSE algorithm. This fusion is used to mitigate the effect of multipath by using the optimal WLSE algorithm when no multipath is detected and by using robust estimators when multipath is detected. This algorithm was published in an article at the ION GNSS+ 2023 (to be published). The variation of robust estimator efficiency as a function of the multipath environment will also be analyzed. The results and benefits of this method versus nominal robust estimators and WLSE algorithms will also be discussed.

## Chapter Contents


7.1 State of the Art Robust Estimation . . . . . 123
7.2 Algorithm Implementation . . . . . 124
7.3 Variation of Robust Estimator Parameters . . . . . 125
7.3.1 Variable efficiency . . . . . 125
7.3.2 Variable breakdown point . . . . . 127
7.4 Method and Scenarios . . . . . 128
7.5 Results . . . . . 129
7.5.1 No Parameter Variation . . . . . 130
7.5.2 Robust estimator parameter variation . . . . . 133

## 7.1 State of the Art Robust Estimation

Various authors have investigated the use of robust estimators for GNSS positioning whether it be to compare their performances with WLSE such as [Medina et al., 2019b] and [Garcia Crespillo et al., 2020] or to incorporate them in Kalman filter algorithms as [Ding et al., 2023]. They found that robust estimators provided better results than traditional positioning algorithms in challenging GNSS environments.

[Medina et al., 2019b] and [Garcia Crespillo et al., 2020] implement a traditional WLSE algorithm to compare with M, S, and MM robust estimators. They compared the different results on experimental data with all kinds of GNSS conditions - highways, urban, bridges etc. They showed that robust estimators based solutions performed better than the WLSE algorithm when the receiver was in harsh conditions.

[Chai et al., 2022] uses robust statistics to compute a positioning solution with carrier phase measurements and mitigate the effect of carrier phase measurements that either have converging ambiguity terms or poorly estimated ambiguity terms due to harsh conditions.

[Wei et al., 2021] aims at detecting outliers to reevaluate the variance of the outliers. Afterwards, they use the new weights to decide if the outlier is small, medium, or large. They use this information to increase the breakdown point of their robust estimator.

[Gaglione et al., 2017] compares the WLS and a receiver autonomous integrity monitoring (RAIM) algorithm (observation subset testing) with a M estimator. He compared these three algorithms in both static and kinematic environments. It was shown that robust estimators outperformed both the WLS and RAIM algorithms regardless of the dynamics of the receiver.

[Wang et al., 2022] implemented a tightly coupled cubature Kalman filter for an INS/GNSS system. They modified this KF to incorporate robust estimation in the measurement update. The proposed robust estimator uses a Geman McClure loss function to reduce the weight of outliers [Barron, 2019].

Some authors have also modified the parameters of robust estimators according to the estimated GNSS conditions such as [Ding et al., 2022] and [Ding et al., 2023]. Their approach is based on using several GNSS parameters such as number of satellites, PDOP etc to vary the weights used in the robust estimations. The weights values vary according to the estimated environment by the classifier. This was done with a neural network for [Ding et al., 2022] and with a SVM for [Ding et al., 2023].

The presented algorithm in this chapter uses a Machine Learning decision on the multipath conditions (presented in chapter 6) to decide if the WLSE algorithm should be used or the robust estimator. As mentioned previously, the WLSE is optimal when the noise can be assumed as Gaussian. Therefore, when no multipath is detected by the CNN, the optimal

estimator can be used. On the other hand, when multipath is detected, the MM estimator is used to limit the impact of outliers.

The proposed approach also uses the number of multipath measurements detected to adapt the robust estimator parameters. The mentioned articles do not explore the potential benefits of an adaptive positioning algorithm. Furthermore, the variation of the efficiency and/or breakdown point has only been investigated by authors in separate works with different classifying methods. A study of both parameters is performed to understand which parameter is more beneficial to vary for the positioning solution.

## 7.2 Algorithm Implementation

In section 3.4, several robust estimators were presented. As the MM estimator combines the good efficiency of M estimators and high breakdown point of S estimators, the MM estimator was chosen as the robust estimator to be fused with the WLSE.

Assuming the CNN model for each signal is already trained, correlators are then accumulated over a second, in this case the time window corresponding to the correlation of the loop filter. An image per tracked satellite is then generated resembling the inputs presented in Figure 6.7. Each image is passed onto the corresponding CNN model and yields a decision on whether multipath affects the input correlators or not.

This decision is then passed onto the positioning algorithm. This is a fusion of the robust MM estimator and the traditional WLSE. When multipath is detected by the CNN in at least one measurement, the position is computed using the MM estimator. On the other hand, when no multipath is detected, the WLSE algorithm is used for the position calculation. This algorithm is depicted in Figure 7.1 for one position computation.

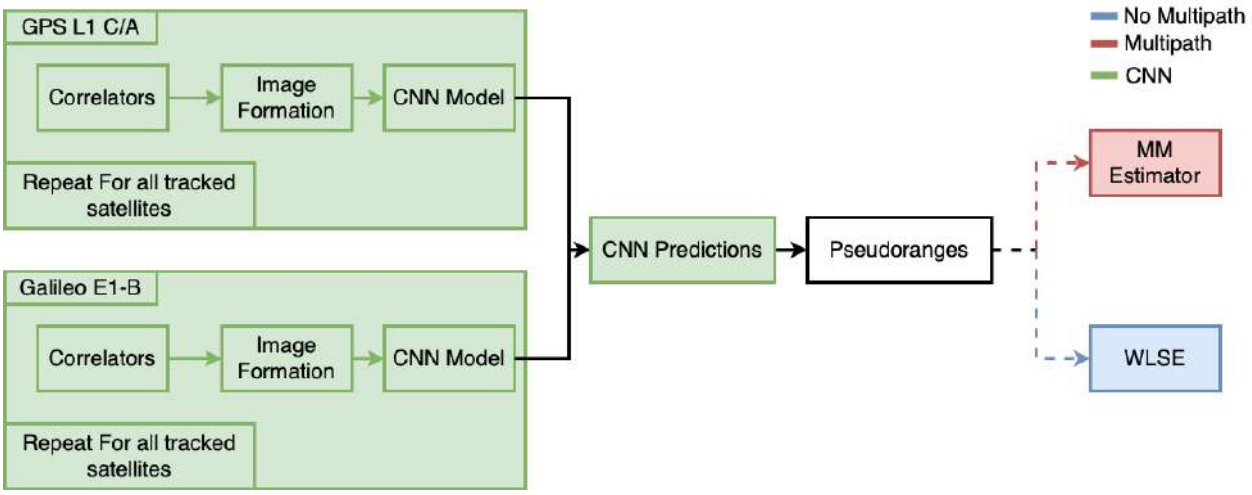


Figure 7.1: Robust Estimator Fusion Algorithm Diagram for one epoch.

WLSE algorithms break down with one outlier and are only optimal when the data error sources can be assumed as Gaussian. However, when multipath affects a code pseudorange, its error can not be assumed as Gaussian anymore [Matera et al., 2018]. On the other

hand, robust estimators are great for dealing with outliers in data but are not optimal when the errors follow a Gaussian distribution. Therefore, the idea was to design an algorithm that could perform ideally when multipath was not detected in the code pseudoranges while resisting when multipath affected the code pseudoranges. As the CNN model is designed to detect if multipath affects the code pseudoranges, the proposed algorithm was implemented and tested.

## 7.3 Variation of Robust Estimator Parameters

Unlike the LSE algorithm, robust estimators have parameters that can be adapted according to their targeted use. Indeed, the loss function, along with the breakdown point (if using an S or MM estimator), and the efficiency parameter can all be changed according to the application.

Since Chapter 6 introduced a method to detect multipath, the idea was to change the values of the breakdown point of the MM estimator and/or its efficiency for Gaussian distributions. With an estimate of the proportion of measurements that are affected by multipath, the breakdown point and efficiency can be varied. Varying the efficiency of the loss function was proposed in [Ding et al., 2022] and [Ding et al., 2023] with improved accuracy with respect to fixed parameters.

Even though the loss function is a design choice when implementing a robust estimator, changing loss functions according to multipath conditions was not implemented. Indeed, the bounds of the loss functions would constantly change, making the robust estimation not continuous. Therefore, this parameter was not investigated, and the Tukey biweight loss function was used in the presented algorithm. It was used because its corresponding weighting function brings large residuals towards zero. This implies that code pseudoranges that are largely affected by errors (including multipath) will not impact the positioning solution or only slightly. The Tukey biweight loss function expression is given in (7.1).

$$\rho_c(x) = \begin{cases} 1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3 \text{ if } |x| \leq c \\ 1 \text{ if } |x| > c \end{cases} \tag{7.1}$$

### 7.3.1 Variable efficiency

The efficiency of a robust estimator indicates how well the estimator performs when there are no outlier affecting the estimator. This is also referred to as the Gaussian efficiency as it is the performance ratio between a robust estimator and the LSE under the nominal noise model [Medina et al., 2019a].

Figure 7.2 illustrates the Tukey bi-weight loss and weight functions as a function of the residuals for different parameters of $c$ corresponding to different efficiencies.

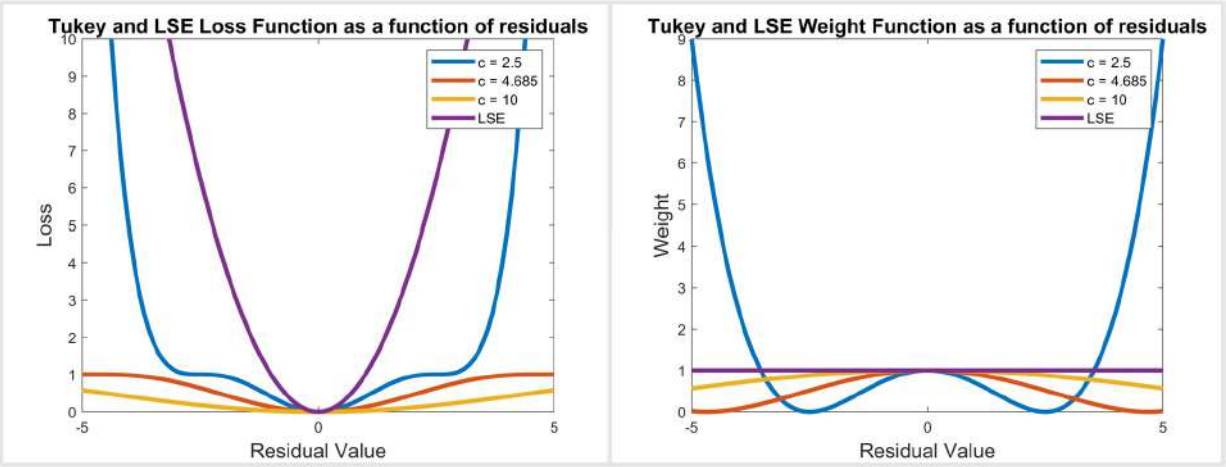


Figure 7.2: Tukey and LSE bi-weight loss functions (left) and weight functions (right) for different values of $c$.

It can be seen that as $c$ increases, the Tukey bi-weight weight function tend to the least squares function. Indeed, as the constant parameter $c$ increases, the efficiency of the robust estimator increases. On the other hand, this leads the measurements to affect the positioning solution equally. Therefore, there is a trade-off between the efficiency under Gaussian conditions and the weighting of measurements in degraded conditions.

Most algorithms set the parameter $c$ in (7.1) for an efficiency of 95%. Similarly to the breakdown point methods presented in section 7.3.2, two methods for varying the efficiency were designed and are described in (7.3) and (7.4). By varying the efficiency, the parameter $c$ of (7.1) is changed which has impacts on how the measurements are weighted. The efficiency can be defined as [Kafadar, 1983]:

$$\text{efficiency} = \frac{\text{minimum variance}}{\text{variance}(\rho_c(x))} \tag{7.2}$$

Therefore, for a given efficiency, the corresponding constant value $c$ can be computed by solving (7.2).

The first method consists in using the estimated proportion of multipath to decrease the efficiency of the estimator. The second method classifies the proportion of multipath measurements into one of four efficiency values depending on its magnitude.

- **Proportion of multipath to compute an efficiency**: By multiplying the efficiency at 95% with a decreasing exponential, this results in a smaller efficiency but more robustness towards outliers. This method is inspired by [Ding et al., 2023] which used exponential factors to impact the efficiency.

$$\epsilon = \epsilon_{0.95} \exp\left(-\frac{M_p}{N}\right) \tag{7.3}$$

  Where $\epsilon$ is the efficiency and $\epsilon_{0.95}$ is the efficiency at 95% under nominal conditions, $M_p$ is the number of multipath measurements, and $N$ the total number of measurements.

- **Discrete efficiencies**: This method uses the proportion of multipath to assign it to its corresponding breakdown point value. This method tries to limit the impact of erroneous estimation by the CNN which could negatively affect the efficiency of (7.3).

$$\epsilon = \begin{cases} \epsilon_{0.95} \exp(-1) \text{ if } \frac{M_p}{N} \geq 0.75 \\ \epsilon_{0.95} \exp(-0.75) \text{ if } 0.75 < \frac{M_p}{N} \leq 0.5 \\ \epsilon_{0.95} \exp(-0.5) \text{ if } 0.5 < \frac{M_p}{N} \leq 0.25 \\ \epsilon_{0.95} \exp(-0.25) \text{ if } \frac{M_p}{N} < 0.25 \end{cases} \tag{7.4}$$

### 7.3.2 Variable breakdown point

The breakdown point is the proportion of outlier measurements that would cause the estimator to produce large errors. Its expression is given in (7.5) from [Rousseeuw and Yohai, 1984].

$$\beta = \min_{m/N}\{\frac{m}{N} \text{ s.t } \|T(Y') - T(Y)\| \to \infty\} \tag{7.5}$$

Where:

- $\beta$ is the breakdown point
- $m$ is the number of corrupted measurements
- $N$ is the number of total measurements
- $Y$ is the vector of measurements
- $Y'$ is the vector of measurements where $m$ measurements of the total $N$ measurements have been corrupted
- $T$ is the estimator

Robust estimators aim to have higher breakdown points than LSE algorithms, which are of $1/N$. However, doing so induces a loss of efficiency when the model is not affected by outliers. Therefore, increasing the breakdown point comes at the expense of efficiency under nominal conditions.

Most implementations suggest a breakdown point with a fixed value of 0.5 indicating that half of the measurements can be outliers. Since the CNN predicts if each measurement is affected by multipath, then the proportion of outliers at each epoch can be known. Therefore, in the proposed algorithm, the breakdown point variability was tested with two different methods. One was to use the proportion of multipath affected measurements as the breakdown point. The other method was an adaptation of the one proposed in [Ding et al., 2023] using discrete values of the breakdown point as a function of the outlier proportion.

Similarly to the efficiency, varying the breakdown point influences the parameter $c$ of (7.1) which has impacts on how the measurements are weighted. The corresponding constant

parameter $c$ can be computed for a given breakdown point as given in (7.6) [Rousseeuw and Croux, 1993].

$$\beta = \frac{\mathbb{E}[\rho_c(x)]}{\rho_c(x)} \tag{7.6}$$

The breakdown point values for each method are given in (7.7) and (7.8):

- **Proportion of multipath as the breakdown point**: This method is straightforward and the breakdown point is given as:

$$\beta = \frac{M_p}{N} \tag{7.7}$$

  Where $\beta$ is the breakdown point.

- **Discrete breakdown points**: This method uses the proportion of multipath to assign it to one of four breakdown point values. This method is a way of limiting the impact of erroneous estimation by the CNN which could greatly impact the breakdown point of (7.7).

$$\beta = \begin{cases} 0.6 \text{ if } \frac{M_p}{N} \geq 0.75 \\ 0.5 \text{ if } 0.75 < \frac{M_p}{N} \leq 0.5 \\ 0.4 \text{ if } 0.5 < \frac{M_p}{N} \leq 0.25 \\ 0.3 \text{ if } \frac{M_p}{N} < 0.25 \end{cases} \tag{7.8}$$

## 7.4 Method and Scenarios

The data setup for these data collects were very similar to the one presented in section 6.3.2. The data was collected in both open-sky and urban conditions. The open-sky collect was used to have data that would not be greatly impacted by multipath. On the other hand, the urban data was used to obtain multipath errors to detect. Similarly to the dataset presented in Chapter 6, the data was gathered on GPS L1 C/A and Galileo E1-B, with a 200 MHz sampling frequency. The IFEN SX3 was still used as the software GNSS receiver [IFEN, 2023] and to gather correlator values. The parameters used on the IFEN are the same as in section 6.3.2 and are recalled in Table 7.1. The processed pseudoranges come from the RINEX observation files which are generated by the IFEN SX3 itself.

| | GPS L1 C/A | Galileo E1-B |
|---|---|---|
| E-L Spacing (chips) | 0.125 | 0.04 |
| DLL Bandwidth (Hz) | 1 | 1 |
| Integration Time (ms) | 1 | 4 |
| Number of correlators | 101 | 101 |
| Correlator Sampling Frequency (MHz) | 200 | 200 |

Table 7.1: IFEN SX3 Settings for GPS L1 C/A and Galileo E1-B.

To evaluate the position performance of the proposed algorithm, a reference position has to be used. The Novatel SPAN, presented in section 5.5.2.2, was used as the reference system.

The data setup is very simple as it requires two PCs, each connected to either the IFEN or the SPAN. The data setup is illustrated by the diagram on Figure 7.3.

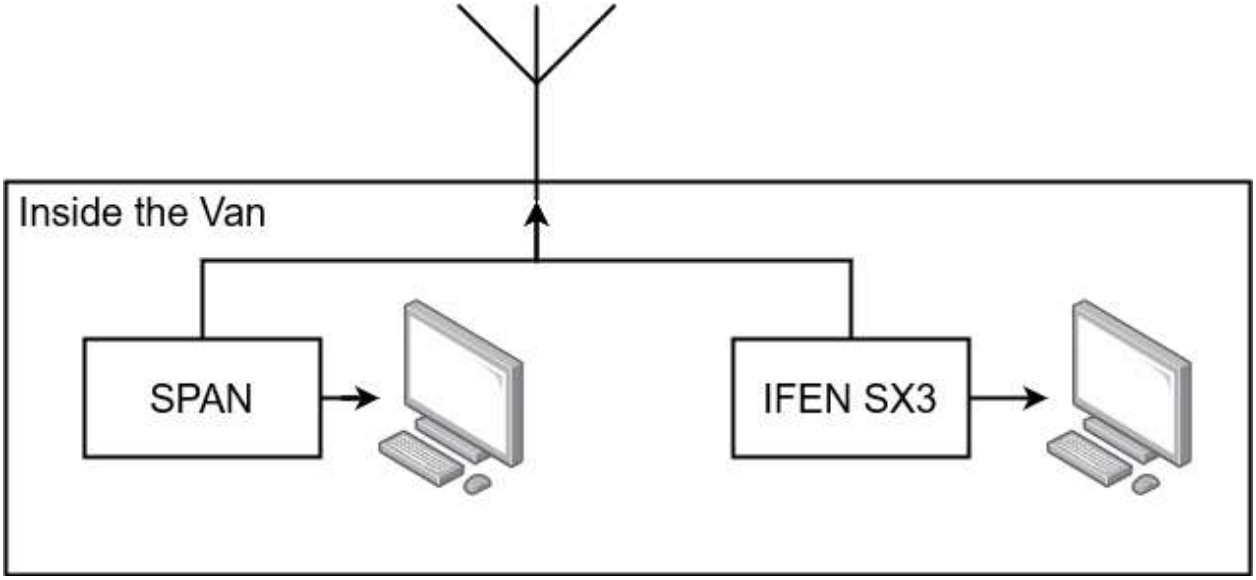


Figure 7.3: IFEN Data set-up to record data.

To quantify the benefit of the positioning solution, four data collects were performed to have different training data for the CNN. This also allowed for the CNN to be trained on three different independent data collects and to test the positioning method on an independent data collect:

- 12th of September 2022: Dynamic sub-urban and deep urban of 40 minutes
- 13th of January 2023: Static open-sky of 1 hour
- 20th of March 2023: Dynamic sub-urban and deep urban of 20 minutes
- 21st of March 2023: Dynamic sub-urban and deep urban of 45 minutes

## 7.5 Results

In order to demonstrate the benefit of the proposed algorithm, the results presented in this section will treat the four data collects as one whole data collect. This can be done because snapshot positioning is used in the investigated algorithms. Indeed, WLSE and MM algorithms do not use previous information but just the measurements at the current time index. This means that the CDF will group the different data collects into one curve.

The MM estimator is expected to perform better than the WLSE in challenging conditions due to its higher breakdown point and to its capacity to minimize the impact of outliers. On the other hand, the WLSE is expected to outperform the MM estimator in outlier free scenarios. Therefore, combining all data collects as one data collect theoretically allows for both algorithms to showcase their strengths while also demonstrating their weaknesses. The proposed combination of WLSE and MM estimator based on the CNN predicted environment should then maximize both estimators' strengths.

The results are presented for the combination of MM and WLSE algorithms along with their comparative algorithms: using WLSE only and using the MM estimator only. Firstly, the mix of WLSE and MM estimators to compute a positioning solution is investigated without varying the parameters of robust estimators. Then, the variation of the robust estimator parameters with the method presented in section 7.3 is studied and discussed.

The number of used satellites does not depend on the variation of parameters. The number of measurements detected as multipath affected are also independent from the parameters of the robust estimator. Therefore, these figures are given in Figure 7.4 and Figure 7.5. Since the data collects are used together as one data collect grouped together, sudden drops of used satellites can be observed in these figures.

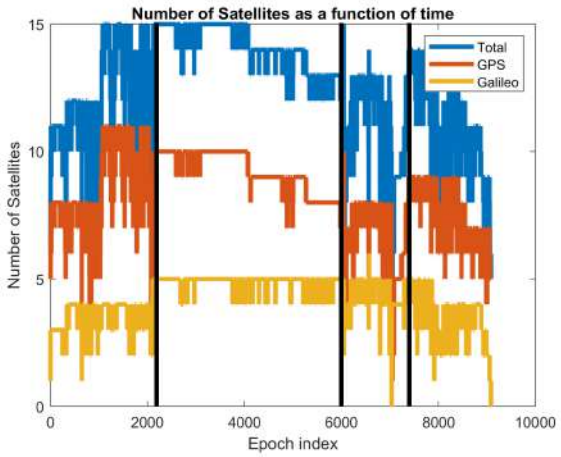


Figure 7.4: Number of satellites used over the concatenated data collects.

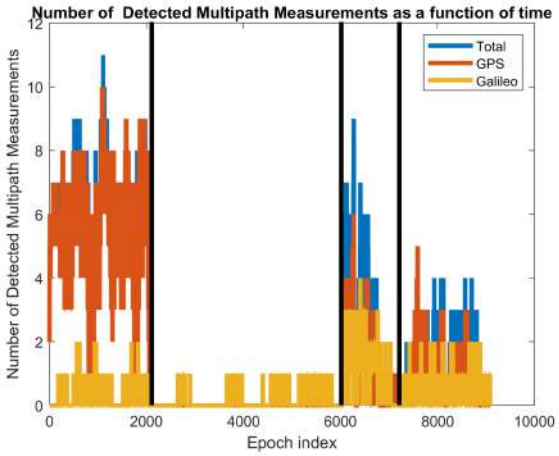


Figure 7.5: Number of detected multipath affected satellites over the concatenated data collects.

Figure 7.4 depicts the number of satellites used for the compounded data collect. The open-sky collect is contained between the 2000 and 6000 epoch indices. The other epochs corresponds to the urban collects where the number of satellites fluctuates more. Figure 7.5 illustrates the number of code pseudorange measurements that have been detected as affected by multipath. The number of affected code pseudoranges is quite high on urban environments indicating that the GNSS conditions are sub-optimal. On the other hand, there is a low frequency of multipath on the open-sky data collect.

### 7.5.1 No Parameter Variation

Figure 7.6 illustrates the CDF of the positioning solution over the different datasets when using the WLSE, MM estimator or the proposed mix of WLSE and MM estimator.

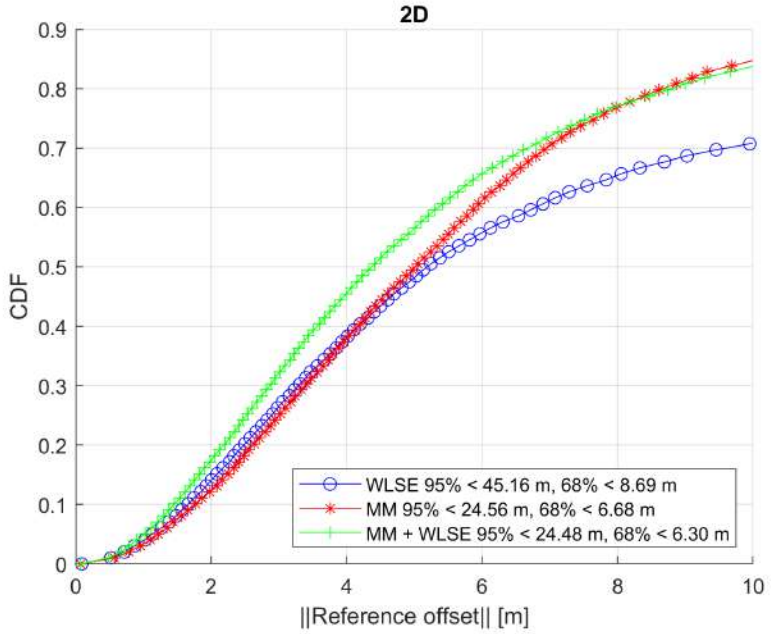


Figure 7.6: CDF of the WLSE, MM estimator, and proposed MM + WLSE mix.

It can be seen that the WLSE algorithm outperforms the MM estimator at the beginning of the CDF curve ($\sim 40\%$) as this is the proportion of data that is collected in open-sky data where the WLSE is barely affected by multipath. Therefore, in this portion the MM estimator is affected by a loss of efficiency due to the data being more Gaussian than it would be in multipath heavy conditions. However, as the position solution degrades, coinciding with higher multipath, the MM estimator positioning solution is much better. This confirms that MM estimators outperform the WLSE algorithm in the presence of outliers.

It can also be observed that the best performing algorithm is the proposed mix of WLSE and MM estimator. Even though the 68% and 95% horizontal errors are very similar to the one of the MM estimator, it can be seen that the mix of MM+WLSE outperforms the MM estimator significantly between 20% and 60% of the position outputs. Indeed, there is a 20.6% and 11.0% increase in positioning accuracy in the 20% and 60% horizontal error respectively. This can be explained by the fact that when multipath is correctly detected by the CNN, the mix of WLSE and MM algorithms uses the most optimal positioning solution for the conditions. Indeed, when the MM estimator computes a positioning solution with low error with respect to the reference by mitigating multipath, the mix of WLSE + MM uses the MM estimator as well. On the other hand, the WLSE algorithm is used when there is no multipath detected which also has low positioning error as it mostly used when in open-sky conditions.

The CDF of the WLSE + MM algorithm resembles more the one of the MM estimator as the horizontal error percentage increases. This is due to large multipath errors affecting the positioning solution which is detected by the CNN. Thus, the mixed algorithm only uses the MM estimator in these conditions.

Figure 7.7 represents the normalized 2D error of each discussed algorithm as a function

of the percentage of urban data used with respect to open-sky data. This graph is designed to highlight the benefit of the proposed method in urban and open-sky scenarios and was created with the steps given below.

1. Randomly select 100 epochs from both static and urban datasets to construct one randomly subsampled open-sky dataset and one for urban conditions with a chosen mix of epochs coming from either open-sky or urban environment.
2. Compute the positioning solutions using the 3 algorithms.
3. Take the x-th quantile of the horizontal error for each data mix.
4. For each percentage of urban data, repeat steps 1, 2, and 3 for N iterations for Monte-carlo simulation - chosen as 2000.
5. As there are a cloud of points (N) for each urban/open-sky ratio, take the median of each ratio.
6. Normalize by highest value of the median values to have a range from 0 to 1.

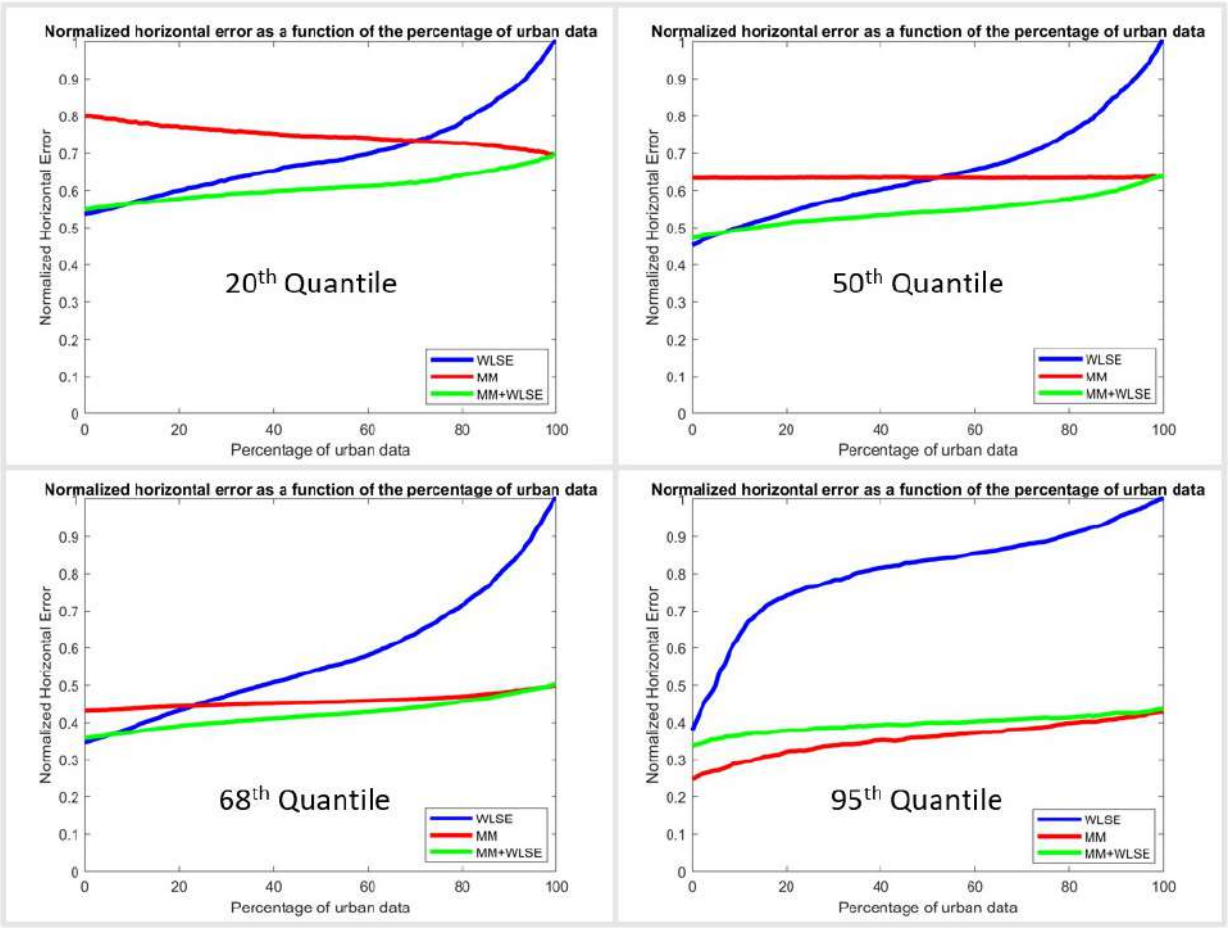


Figure 7.7: Normalized horizontal error as a function of the percentage of urban data used with respect to open-sky data for the WLSE, MM estimator, and MM+WLSE algorithms for the 20-th, 50-th, 68-th, and 95-th quantile.

The beginning of the x-axis of Figure 7.7 represents when mostly open-sky data is used and at the end of the x-axis represents when mostly urban data is used. Each sub-figure corresponds to a different horizontal position error quantile used to generate these plots as described in step 3. Different quantiles are shown to illustrate the benefit of the proposed algorithm according to the different GNSS environment.

Figure 7.7 illustrates that the MM estimator performs sub-optimally when no code pseudorange outliers are present with respect to the WLSE for all but the 95th quantile. The MM performance is steady and performs much better than the WLSE when the conditions tend to be more urban. For the 95th quantile of the normalized horizontal error, the MM estimator always outperforms the WLSE even in open-sky conditions. This can be explained by the fact that the open-sky data can still be affected by multipath which can correspond to the highest positioning errors. Furthermore, larger residual biases like the atmospheric errors etc can impact the positioning solution and be mitigated by the MM estimator and not the WLSE.

It can be observed that the proposed MM + WLSE algorithm performs similarly to the WLSE algorithm when mostly open-sky data is used for all quantiles. As more urban data is used, the normalized horizontal error of the WLSE rapidly increases and then performs worse than the MM estimator. On the other hand, the normalized horizontal error of the MM+WLSE algorithm increases steadily to eventually match the one of the MM estimator. This illustrates the fact that the proposed algorithm takes advantage of the optimality of the WLSE algorithm when the data is unaffected by multipath while limiting the impact of outliers when multipath affects pseudoranges.

On the other hand, the figure for the 95-th quantile shows that the proposed mix of the MM and WLSE algorithm performs worse than using only the MM estimator. This can be explained by the fact that the proposed algorithm uses the WLSE which, as mentioned earlier, can be still affected by multipath for the 95-th quantile of the horizontal positioning error. Furthermore, due to false detection of the multipath conditions by the CNN, the WLSE may sometimes be used instead of the MM estimator. However, the difference between the two normalized errors is small, and as the percentage of urban data increases, the proposed algorithm performs similarly to the MM estimator.

### 7.5.2 Robust estimator parameter variation

This section presents the results of the mix of WLSE and MM algorithm when the efficiency and/ or breakdown point are varied as presented in section 7.3. The algorithms discussed have the following notation:

- MM + WLSE $E_C$: Uses a mix of WLSE and MM estimator with a variable efficiency point computed in (7.3).
- MM + WLSE $E_D$: Uses a mix of WLSE and MM estimator with a variable efficiency point computed in (7.4).
- MM + WLSE $B_C$: Uses a mix of WLSE and MM estimator with a variable breakdown point computed in (7.7).

- MM + WLSE $B_D$: Uses a mix of WLSE and MM estimator with a variable breakdown point computed in (7.8).

Table 7.2 shows the results of the CDF for all parameter variation methods corresponding to their horizontal error.

| Algorithm | 68% Horizontal Error | 95% Horizontal Error |
|---|---|---|
| MM + WLSE Constant Parameter | 6.30 | 24.48 |
| MM + WLSE $E_C$ | 6.30 | 24.66 |
| MM + WLSE $E_D$ | 6.37 | 26.32 |
| MM + WLSE $B_C$ | 6.92 | 32.93 |
| MM + WLSE $B_D$ | 6.50 | 27.26 |

Table 7.2: CDF results for all algorithm parameter variations.

Table 7.2 shows that there is no parameter configuration that improves the MM+WLSE algorithm when the efficiency and breakdown point were kept constant for the MM estimator. Indeed, the best parameter variation was to vary the efficiency according to (7.3) which was worse than when the efficiency was constant.

The efficiency variations based on the multipath conditions do not add any benefit to the robust estimator. Both (7.3) and (7.4) decrease the efficiency according to the multipath environment. As highlighted by Figure 7.2, the weights of the Tukey biweight function, for $c = 2.5$, decreases as residual values increase. However, the value of weights increase again when $w > c$. Therefore, if the code pseudorange residuals are too high, which can occur in heavy multipath environments, these measurements have even more importance than non-multipath affected ones. However, increasing the efficiency to $c = 10$ for example, as shown in Figure 7.2, leads all measurements to almost be weighted equally. Hence, keeping the efficiency value for $95\%, c = 4.685$ is preferred as it reduces high residual weights and re-increases for a higher $c$ value.

The breakdown point variation did not improve the positioning accuracy of the WLSE+MM algorithm either. This can be explained by the fact [Rousseeuw and Yohai, 1984] does not recommend the breakdown point to be higher than 0.5 and lower than 0.25 which can occur when varying the breakdown point of (7.7) and (7.8). Furthermore, the varying breakdown point of (7.7) could be wrongly estimated due to a CNN error which would significantly impact the positioning solution. Even though the breakdown point computed by (7.8) reduces the impact of potential CNN errors which is conveyed by its higher positioning accuracy, the discrete mapping applied may be erroneous.

## 7.6 Conclusions

This chapter described an algorithm that uses either the weighted least squares estimation algorithm or the MM estimator. The decision to use one or the other is based on the estimated multipath environment. Each code pseudorange is classified as multipath affected or not. This classification is obtained by the CNN presented in Chapter 6. The MM estimator

is used as the positioning algorithm when one or more measurement is deemed as multipath affected and the WLSE is used when no multipath is detected. This algorithm was proposed to take advantage of the robustness of MM estimators when outliers are present and the optimal WLSE when there are no outliers. This work led to a publication in the ION GNSS+ 2023 (to be published).

The proposed algorithm showed that it overcomes the loss of efficiency of the MM estimator when no outliers are present and mitigates the positioning degradation of the WLSE in heavy multipath environments. Therefore, this algorithm fuses the benefits of the WLSE algorithm and MM estimator with minimal downsides. When the data is collected in open-sky and urban conditions, it was shown to improves the MM estimator accuracy by 20% and 11% on the 20% and 60% horizontal positioning accuracy. It was also shown that the proposed variation of the parameters of the robust estimator parameters -efficiency and breakdown point- as a function of the number of detected multipath could be detrimental to the positioning accuracy.

# CHAPTER 8

# Conclusions and Perspectives

This chapter summarizes the conclusions brought by the research presented in this work. It also provides insight on potential future research in the different investigated fields.

## 8.1 Conclusions

As presented in Chapter 1, the goal of this thesis was to develop light-weight algorithms, for a potential implementation on the eHermes navigation solution of 3D Aerospace, that would improve the positioning solution especially in challenging environments.

This was achieved in three phases. The first one consisted in improving the computational load of an ambiguity estimating EKF by using time differenced carrier phase measurements (TDCP) instead of raw carrier phase measurements. The second phase was based around detecting multipath affecting code pseudoranges by using CNNs applied to multicorrelator outputs. The last one was to use the multipath classification from the CNN to improve the positioning accuracy of weighted least squares and MM estimators. The different conclusions are arranged to their corresponding phases and given below.

- Inclusion of TDCP in an EKF

To better understand the functioning of Kalman filters, their theoretical equations were presented along with two EKFs using GNSS measurements in Chapter 3. One filter was based on using only GNSS measurements and the other used a INS/GNSS tight coupling EKF while estimating the ambiguity of carrier phase measurements. The latter served as the benchmark to compare the proposed TDCP filter presented in Chapter 5. The ambiguity estimating filter used carrier phase, Doppler, and code pseudoranges measurements which were presented in Chapter 2 to illustrate how they were generated and their differences.

Chapter 5 presented how to form time differenced carrier phase measurements. It was shown that using measurements from two different time epochs had an influence on the measurement update equations of the KF. The TDCP measurements were used to replace carrier phase measurements in the filter. This is because using raw carrier phase measurements leads to estimating the carrier phase ambiguities within the EKF state vector. The increase in the number of states to estimate inevitably leads to more calculations. Since the eHermes has limited computational resources, the additional state estimation would be too cumbersome. Therefore, the computational load of the ambiguity estimating filter was compared with the

TDCP filter. It was demonstrated that the TDCP filter was 57% faster than the ambiguity estimation filter in urban conditions.

Cycle slips also affect TDCPs as they are a combination of two carrier phase measurements. Hence, cycle slip detection methods were implemented. More specifically, two methods were tested to analyze the performance of each method. One method was based on rejecting TDCP measurements based on normalizing the innovations by the innovation covariance and pitting this against a threshold value. The other method involved comparing the innovation with the wavelength of the signal as cycle slips induce at least a change of one wavelength. It was proven that the EKF innovation-based method performed better especially when using ionosphere-free TDCPs as it rejected less useful measurements as the wavelength method is too stringent.

The GNSS observations used were collected in dual frequency. Thus, the differences in position accuracy and innovations between first order ionosphere free and uncombined TDCPs were also studied. When the filter was using uncombined TDCPs, it performed similar to the iono-free TDCP filter for 68% and 11% better for 95% of the horizontal error in urban conditions. This demonstrated that having more TDCP measurements is more beneficial to the positioning solution than fully removing the first order differential ionosphere effect.

It was shown that as the conditions degrade, the impact of TDCP measurements increases and even outperforms the ambiguity estimation filter. Indeed, the TDCP filter using ionosphere-free measurements with the EKF rejection based method performed 5% and 16.7% better than the ambiguity estimation filter in deep urban conditions for the 68% and 95% of the horizontal error respectively. In sub-urban conditions, the performance of the TDCP filter was similar for the 68% horizontal error to the carrier ambiguity estimation filter. However, the TDCP filter outperformed the carrier filter by 13.8% for the 95% horizontal error. This further shows the benefit of TDCPs as a light-weight and robust solution in challenging environments.

- Detecting Multipath using Convolutional Neural Networks

GNSS correlators were used as inputs to a convolutional neural network (CNN); thus, to give insight and knowledge on how correlators are computed in a GNSS tracking loop, the signal processing chain was detailed in Chapter 2. As the goal of this research was to detect GNSS multipath with CNNs, the theory behind CNNs and how to construct a model was described in Chapter 4.

Chapter 6 proposed a reference classification of multipath using GNSS tracking correlator outputs. The classification criterion revolved around computing the estimated filtered tracking bias, which is almost equivalent to the pseudorange error, by reconstructing the autocorrelation function of the corresponding signal. Then, the filtered tracking bias over one second was compared to a threshold to decide whether multipath affected the receiver over this one second span. The threshold was determined by using the multipath error envelope expected from GNSS signal propagation in urban environments taking into account frequently encountered materials.

This classification was used to label data to train a supervised convolutional neural network.

Then, this model was used to predict whether multipath was present or not on correlator outputs from GPS L1 C/A and Galileo E1-B. The CNN architecture was kept to a low complexity so that it could be trained with a limited amount of data while limiting the risk of under or overfitting.

The inputs of the proposed model were the correlators values from the DLL as a function of delay (with respect to the prompt correlator) and time in the form of images of 256x256 pixels in grey scale. The inputs of the model are lightweight, making the predictions yielded by the CNN to be swiftly computed. This is illustrated by the fact that predicting multipath using the CNN model is around 21 times faster than computing the tracking bias from a large number of correlator outputs for GPS L1 C/A and two to six times faster for Galileo E1-B.

Several variations have been tested in the CNN to test how the CNN performs when reducing the number of correlators to generate the input images. The favored configuration is to decrease the number of correlators along with the correlator sampling frequency. This results in an observation of the correlation function over the same delay range, but with a lower resolution. Indeed, it was shown that by reducing the number of correlators from 101 to 25 and reducing the resolution by a factor of 4, the F score only decreased from 94.7% to 91.8% on Galileo E1-B and from 91.6% to 90.5% on GPS L1 C/A.

- Robust Estimator Fusion Algorithm according to the Estimated Multipath Environment

Since the proposed algorithm used a combination of a robust estimator and the WLSE, these algorithms were theoretically presented in Chapter 3. The design of a MM estimator and WLSE for positioning purposes was discussed. The robustness of the MM estimator to outliers was also presented thanks to the loss function bounding the residual estimates. The impact that multipath can have on measurements was also illustrated in Chapter 2.

Chapter 7 proposed an algorithm that uses either a MM estimator or the WLSE based on the estimated multipath conditions. The MM estimator is used as the positioning algorithm when one or more measurement is deemed as multipath affected, and the WLSE is used when no multipath is detected. This multipath decision came from the CNN results presented in Chapter 6. It was shown that this algorithm took advantage of optimality of the WLSE in open-sky conditions and robustness of the MM estimator in urban environments. Indeed, when the data is collected in open-sky and urban conditions, it was shown to improves the MM estimator accuracy by 20% and 11% on the 20% and 60% horizontal positioning accuracy.

Varying the efficiency and breakdown points of the MM estimator according to the detected number of multipath-affected pseudoranges was also studied. The efficiency of the estimator under Gaussian distributions was reduced to reduce the weights of high residual measurements. This was done using the number of detected multipath in one epoch and based on [Ding et al., 2022] but did not yield promising results with respect to leaving the efficiency constant. The variation of the breakdown point was also tested. The breakdown point is the maximum proportion of corrupted measurements at which the robust estimator

breaks down. Therefore, it was thought that by varying it, the robust estimator would perform best. However, the variation of the breakdown point did not improve the positioning accuracy of the proposed algorithm.

## 8.2 Perspectives

Resulting from the presented research, several questions were induced that have not been answered in this work. These questions could lead to future research. The different research perspectives are given below.

- TDCP Filter EKF

As Global Navigation Satellite Systems (GNSS) continue to expand, there is an increase in the number of available signals. Indeed, Galileo is introducing Galileo E6 (High Accuracy Service), on top of the already existing signals (E1, E5a, and E5b). Meanwhile, GPS continues to transition its satellites to being L5 capable. Furthermore, Beidou has also become triple frequency capable [Chu and Yang, 2018]. In challenging GNSS conditions, such as urban canyons, the addition of new signals and constellations almost systematically improves the positioning accuracy, convergence or availability [Li, 2018] [Li, 2020].

The proposed filter could add triple frequency capabilities on the code and/or TDCP measurements. Triple frequency combinations could be used to remove the first order ionospheric effect while decreasing the noise of the iono-free measurement with respect to the dual frequency combination. To keep the filter computationally lightweight, a simple combination as the one proposed in [Deo and El-Mowafy, 2016] could be implemented. Having triple frequency measurements could also lead to more (uncombined) TDCP measurements being used. Indeed, as shown in Chapter 5, having more TDCP measurements lead to a better positioning solution in challenging GNSS conditions.

Longer data collects could also be performed using the MEMS IMU along with the rest of the dataset up presented in Chapter 5. This would better highlight the benefit of the TDCP filter when using a set-up similar to the eHermes receiver, especially in deep urban conditions. Moreover, it would also prove the assumption that the best performing TDCP filter is when using uncombined TDCPs over iono-free measurements even with a MEMS IMU.

- Detecting Multipath using Convolutional Neural Networks

In the future, the number of correlators used to build an input image to the CNN could be further reduced. The position of the correlators could also be varied to investigate what could be the minimum number of correlators to correctly classify multipath at a given performance. This could lead the method to be more implementable on real time receivers as the number of computed correlators would be more achievable. 3D images could also be built by taking into account the Doppler offset of the received signal as done in [Munin et al., 2020]. Furthermore, other factors could be added for the threshold determination such as the elevation of the satellite.

Excluding measurements as they are affected by multipath does not always lead to better positioning solution as it can lead too few measurements to compute a position solution, or sometimes this satellite was more important to the overall geometry than excluding the error was. Hence, the next step is to find a criterion to know if removing the multipath affected measurement is worth it. Then based on the results of the positioning accuracy, the multipath threshold can be redefined and optimized accordingly.

Another lead to explore would be to classify satellites into several groups based on their elevation or CN0. Then, a CNN model per group could be built. Evidently, each CNN would be trained with sufficient data and that data would correspond to the elevation or CN0 of the targeted group. As predicting with a CNN was shown to not be a computational bottleneck, the only downside of this solution would be storing the models. This method would most likely increase the detection of multipath for low elevations or CN0s.

- Robust Estimator Fusion according to the Estimated Multipath Environment

The proposed algorithm was implemented for snapshot positioning. However, snapshot positioning is not the most accurate positioning algorithm. For example, the EKF presented in Chapter 5 would be more robust than the presented algorithm. It is possible to adapt the EKF to use a robust estimator. This was done in [Garcia Crespillo et al., 2017]. Therefore, a future implementation of the proposed algorithm would be to use "nominal" EKF observation update equations when no multipath is detected and switch to the robust EKF when multipath is detected by the CNN.

Tukey biweight's function is a popular loss function for MM estimators and performed better than the other tested loss functions. However, as shown in section 7.3, the weight function can increase when the residual is higher than the constant $c$ causing problems for high residuals. Another loss function could be investigated when the breakdown point or efficiency of the MM estimator are varied. This loss function would need to have better mitigation of high residuals.

The proposed CNN models were designed to detect multipath on Galileo E1-B and GPS L1 C/A. A future lead could be to extend the multipath detection to other frequencies such as the L5 band (1176.45 MHz) to detect multipath on both Galileo E5a and GPS L5. This could be more resource demanding as a higher number of correlators would probably be needed for the same performance due to the higher chipping rate of PRN codes on both signals. An alternate solution to use dual frequency measurements, could be to assume that if a measurement on the L1 band is estimated as multipath affected then it is also affected on the L5 band. Even though it may not be the case as L5 and E5a signals are more robust to multipath [Circiu et al., 2017]; this would avoid extra models to train. It would also lead to less correlators computed for real-time receivers while only using estimated multipath-free measurements on dual frequencies.

# APPENDIX A

# EKF State and Observation Models

This appendix details the computation of the elements of the state transition matrix $\Phi_{k-1}$ and of the observation matrix $H_k$ for the GNSS/INS hybridization given in section 3.3.2. The computation for the GNSS only filter presented in section 3.3.1 are identical when removing the terms linked to the INS.

## A.1 State Transition Model

The state transition matrix is equal to:

$$\Phi_{k-1} = \begin{bmatrix} I_3 & \Delta_T I_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & 0_3 & 0_3 \\ 0_3 & -2\Omega_{ie}\Delta_T & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & \Delta_T[-R_b^e f^b\times] & -\Delta_T R_b^e & 0_3 \\ 0_{1\times3} & 0_{1\times3} & 1 & \Delta_T & 0 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 1 & 0 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 & 1 & 0 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 & 0 & 1 & 0_{1\times N} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{N\times3} & 0_{N\times3} & 0_{N\times1} & 0_{N\times1} & 0_{N\times1} & 0_{N\times1} & I_N & 0_{N\times3} & 0_{N\times3} & 0_{N\times3} \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & -\Delta_T\Omega_{ie} & 0_3 & -\Delta_T R_b^e \frac{\pi}{180} \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times N} & 0_3 & 0_3 & I_3 \end{bmatrix} \tag{A.1}$$

The state transition matrix of (A.1) can be obtained by $\Phi_{k-1} = \exp(F_{k-1}\Delta_T)$. The matrix $F_{k-1}$ is composed of the coefficients that express the derivative of a given state as a function of other states. (A.2)-(A.11) express all state derivatives as a function of other states. For state derivatives that cannot be expressed as a function of other states are expressed as zeros.

$$\frac{\partial \delta r^e}{\partial t} = \delta v^e \tag{A.2}$$

$$\begin{aligned} \frac{\partial \delta v^e}{\partial t} &\approx \hat{R}_b^e \hat{f}^b - R_b^e f^b \\ \frac{\partial \delta v^e}{\partial t} &\approx [-R_b^e f^b\times]\delta\psi^e - 2\Omega_{ie}\delta v^e - R_b^\delta b_{acc} \end{aligned} \tag{A.3}$$

In (A.3), the variation in gravity is assumed to be null.

$$\frac{\partial \delta dt_{Rx}}{\partial t} = \delta \dot{dt}_{Rx} \tag{A.4}$$

$$\frac{\partial \delta \dot{dt}_{Rx}}{\partial t} = 0 \tag{A.5}$$

$$\frac{\partial \delta dt_{GPS-GAL}}{\partial t} = 0 \tag{A.6}$$

$$\frac{\partial \delta ztd}{\partial t} = 0 \tag{A.7}$$

$$\frac{\partial \delta N^j}{\partial t} = 0 \tag{A.8}$$

$$\begin{aligned} \frac{\partial \delta \psi^e}{\partial t} &\approx \hat{R}_b^e(\hat{w^b} - w^b) \\ \frac{\partial \delta \psi^e}{\partial t} &\approx -\Omega_{ie}\delta\psi^e - R_b^e \delta b_{gyro} \end{aligned} \tag{A.9}$$

In (A.9), the small angle approximation was used.

$$\frac{\partial \delta b_{acc}}{\partial t} = [0;0;0] \tag{A.10}$$

$$\frac{\partial \delta b_{gyro}}{\partial t} = [0;0;0] \tag{A.11}$$

## A.2 Observation Model

The model for the three observables are detailed: code pseudorange, carrier phase, and Doppler measurement.

### A.2.1 Code Pseudorange and Carrier Phase

The observation model is recalled:

$$H_{k,\rho} = \begin{bmatrix} -\vec{e_{INS}}^1 & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 & -\vec{e_{INS}}^1 R_b^e(-\vec{p^b}_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ -\vec{e_{INS}}^N & 0_{1\times3} & 1 & 0 & C_{ind} & ztd & 0 & \dots & 0 & -\vec{e_{INS}}^N R_b^e(-\vec{p^b}_{Rx}\times) & 0_{1\times3} & 0_{1\times3} \end{bmatrix} \tag{A.12}$$

The matrix of (A.12) can be obtained by differentiating the code pseudorange with respect to the states of the state vector. The pseudorange is received at the GNSS antenna, but the EKF position is computed at the IMU's position. Therefore, the lever arm must be taken into account to transform the GNSS code pseudorange to the IMU's coordinates as given in (A.13).

$$\vec{r}^e_{INS} = \vec{r}^e_{Rx} + R_b^e \vec{p}^b_{Rx} \tag{A.13}$$

The code pseudorange model is given in (A.14).

$$\rho^j_{INS} = (\vec{r}^j - \vec{r}_{INS}) \cdot \vec{e_{INS}}^j + c(dt_{Rx} - dt^j) + T^j_{Rx} + I^j_{Rx} + \delta^j_{\rho,Rx} + \varepsilon^j_{\rho,Rx} \tag{A.14}$$

From the model of (A.14), we can simply differentiate the code pseudorange model with respect to each state of the EKF:

$$\frac{\partial \rho^j_{INS}}{\partial \delta x^e} = -\vec{e_{INS}}^j \tag{A.15}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta v^{e}} = [0;0;0] \tag{A.16}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta dt_{Rx}} = 1 \tag{A.17}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta \dot{dt}_{Rx}} = 0 \tag{A.18}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta dt_{GPS-GAL}} = 1 \text{ if Galileo measurement or 0 if GPS} \tag{A.19}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta ztd} = ztd \tag{A.20}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta N_i} = 0 \tag{A.21}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta \psi^{e}} = -\vec{e_{INS}}^{j} R_b^e(-\vec{p_{Rx}^b}\times) \tag{A.22}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta b_{acc}} = [0;0;0] \tag{A.23}$$

$$\frac{\partial \rho_{INS}^{j}}{\partial \delta b_{gyro}} = [0;0;0] \tag{A.24}$$

The carrier phase measurement observation matrix can be obtained with the same reasoning. The only difference is that $\frac{\partial \phi_{INS}^{j}}{\partial \delta N_i} = 1$ to account for the ambiguity term of the carrier phase measurement

### A.2.2 Doppler

The measurement matrix for the Doppler measurement is recalled:

$$H_{k,Doppler} = \begin{bmatrix} 0_{1\times3} & -\vec{e_{INS}}^{1} & 0 & 1 & 0 & 0 & 0 & \dots & 0 & -\vec{e_{INS}}^{1} R_b^e(\vec{v_{Rx}^b}\times) & 0_{1\times3} & -\vec{e_{INS}}^{1} R_b^e(\vec{p_{Rx}^b}\times)\frac{\pi}{180} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0_{1\times3} & -\vec{e_{INS}}^{N} & 0 & 1 & 0 & 0 & 0 & \dots & 0 & -\vec{e_{INS}}^{N} R_b^e(\vec{v_{Rx}^b}\times) & 0_{1\times3} & -\vec{e_{INS}}^{N} R_b^e(\vec{p_{Rx}^b}\times)\frac{\pi}{180} \end{bmatrix} \tag{A.25}$$

Similarly to the code pseudorange, the matrix of (A.25) can be obtained by differentiating the Doppler measurement with respect to the chosen EKF states. The Doppler is also received at the GNSS antenna; thus, the lever arm must be taken into account to express the Doppler measurement at the IMU's coordinates.

$$\vec{v_{INS}^e} = \vec{v_{Rx}^e} + R_b^e(\vec{\omega}_b \times \vec{p_{Rx}^b}) \tag{A.26}$$

The Doppler measurement model is recalled as:

$$D_{INS}^{j} = f_{c_j} \frac{(\vec{v_{INS}} - \vec{v_j}) \cdot \vec{e_{INS}}^{j}}{c + \vec{v_j} \cdot \vec{e^j}} \tag{A.27}$$

The differentiation for each state is given in (A.28)-(A.37).

$$\frac{\partial D_{INS}^{j}}{\partial \delta x^{e}} = [0;0;0] \tag{A.28}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta v^{e}} = -\vec{e_{INS}}^{j} \tag{A.29}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta dt_{Rx}} = 0 \tag{A.30}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta \dot{dt}_{Rx}} = 1 \tag{A.31}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta dt_{GPS-GAL}} = 0 \tag{A.32}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta ztd} = 0 \tag{A.33}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta N_{i}} = 0 \tag{A.34}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta \psi^{e}} = -\vec{e_{INS}}^{j} R_{b}^{e}(\vec{v}_{INS}^{b}\times) \tag{A.35}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta b_{acc}} = [0;0;0] \tag{A.36}$$

$$\frac{\partial D_{INS}^{j}}{\partial \delta b_{gyro}} = -\vec{e_{INS}}^{j} R_{b}^{e}(\vec{p}_{Rx}^{b}\times)\frac{\pi}{180} \tag{A.37}$$

# APPENDIX B

## Median Absolute Deviation Relationship with Standard Deviation for Gaussian Laws

The median absolute deviation (MAD) can be defined as:

$$MAD = median(|X - median(X)|) \tag{B.1}$$

Where:

- $MAD$ is the median absolute deviation
- $X$ is a set of variables following a given law

The MAD can be linked to the standard deviation for a given law as such:

$$\sigma = k \cdot MAD \tag{B.2}$$

Where:

- $\sigma$ is the standard deviation
- $k$ is a scale factor which depends on the distribution

For a Gaussian law, $k = 1.4826$ with the derivation shown in (B.3)-(B.8). Since the MAD covers 50% of the Gaussian CDF, it can be expressed as:

$$\frac{1}{2} = P(MAD \geq |X - \mu|) \tag{B.3}$$

We then divide both by the standard deviation to incorporate it.

$$\frac{1}{2} = P(\frac{MAD}{\sigma} \geq \frac{|X - \mu|}{\sigma}) \tag{B.4}$$

Using that $P(a < X \leq b) = F_X(b) - F_X(a)$ where $F_X$ is the CDF function:

$$\frac{1}{2} = \Phi(\frac{MAD}{\sigma}) - \Phi(-\frac{MAD}{\sigma}) \tag{B.5}$$

Where:

- $\Phi$ is the CDF of a Gaussian distribution

With the identity:

$$1 = \Phi(\frac{MAD}{\sigma}) + \Phi(-\frac{MAD}{\sigma}) \tag{B.6}$$

By plugging (B.6), it yields:

$$\frac{3}{4} = \Phi(\frac{MAD}{\sigma}) \tag{B.7}$$

By taking the inverse of the CDF funcion, we have:

$$\sigma = \frac{MAD}{\Phi^{-1}(3/4)} = 1.4826 \cdot MAD \tag{B.8}$$

# APPENDIX C

# Different CNN Architecture Results

Before settling on the CNN architecture presented in Chapter 6, different architectures were tested that did not perform as well or were much more computationally heavy without a clear benefit in metric performance. This section aims at showing that the chosen CNN architecture was the best performing amongst the tested ones. It is also meant to convey that the problems presented in Chapter 4 can occur if not accounted for.

## C.1 Overly Complex Model

Figure C.1 depicts the CNN architecture of a network that has two convolutional layers to detect multipath.

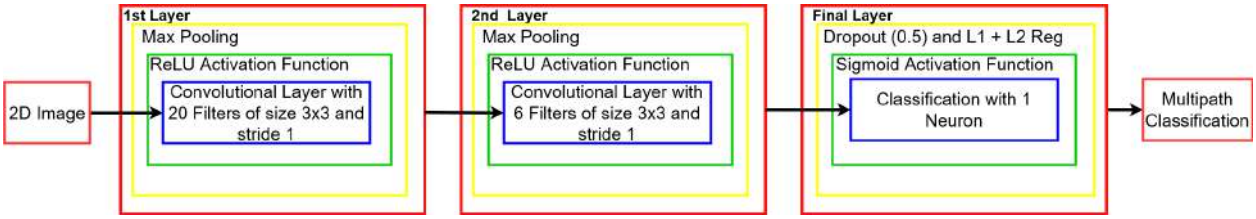


Figure C.1: Complex CNN Architecture for Multipath Detection.

Table C.1 displays the metrics results of the CNN architecture presented here. It can be seen that the model performs worse than the one presented in Chapter 6. It is due to the fact that the additional convolutional layer leads to more connections which need to be trained. The amount of data required to train this network is then much more substantial as conveyed by the difference in training and test results. Furthermore, this network is much slower due the increased number of convolutions performed.

| Metric | Training | Test |
|---|---|---|
| Accuracy | 0.94 | 0.91 |
| Recall | 0.90 | 0.83 |
| Precision | 0.97 | 0.97 |
| F Score | 6.50 | 27.26 |

Table C.1: CNN Performance Metric on Galileo E1-B for 101@200 with a complex CNN model.

## C.2 Overfitting model

Figure C.2 illustrates the CNN architecture of a network that causes overfitting when training to detect multipath. The difference between this network and the one presented in Chapter 6 is the absence of dropout and elliptic net regression.

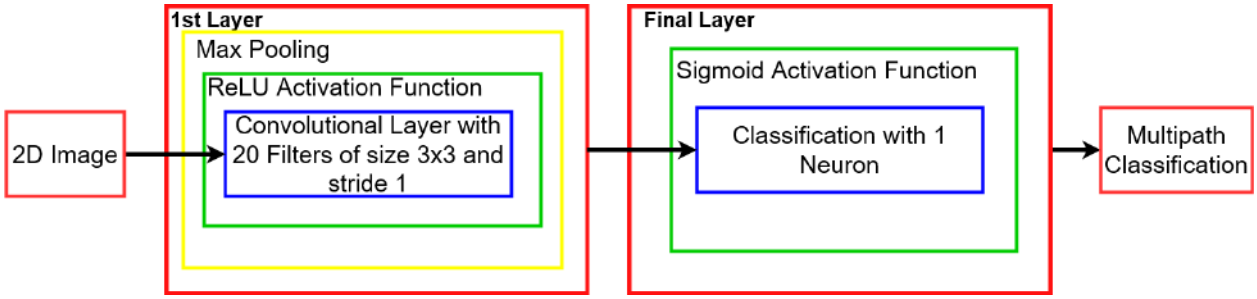


Figure C.2: Overfitting CNN Architecture for Multipath Detection.

Table C.2 displays the results of the network presented in Figure C.2. The metrics clearly show that the model performs better when using trained data than test data. This is because of the absence of dropout and regression. This causes the network to "learn" the training data and heavily adapt its connections based on it. However it has trouble generalizing as the test metrics are far off the training ones.

| Metric | Training | Test |
|---|---|---|
| Accuracy | 0.98 | 0.90 |
| Recall | 0.97 | 0.85 |
| Precision | 1.00 | 0.95 |
| F Score | 6.50 | 27.26 |

Table C.2: CNN Performance Metric on Galileo E1-B for 101@200 with an overfitting CNN model.

# Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

[Agarap, 2018] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375.

[Alzubaidi et al., 2021] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *J. Big Data*, 8(1):53.

[Barron, 2019] Barron, J. T. (2019). A general and adaptive robust loss function.

[Bellad and Petovello, 2013] Bellad, V. and Petovello, M. (2013). Indoor multipath characterization and separation using distortions in gps receiver correlation peaks. volume 4.

[Blais et al., 2022] Blais, A., Couellan, N., and Munin, E. (2022). A novel image representation of gnss correlation for deep learning multipath detection. *Array*, 14:100167.

[Blewitt, 2000] Blewitt, G. (2000). Basics of the gps technique : Observation equations.

[Boehm et al., 2006] Boehm, J., Niell, A., Tregoning, P., and Schuh, H. (2006). Global mapping function (gmf): A new empirical mapping function based on numerical weather model data. *Geophysical Research Letters*, 33(7).

[Borio and Lo Presti, 2008] Borio, D. and Lo Presti, L. (2008). Data and pilot combining for composite gnss signal acquisition. *International Journal of Navigation and Observation*, 2008.

[Borre et al., 2007] Borre, K., Akos, D., Bertelsen, N., Rinder, P., and Jensen, S. H. (2007). Chapter 5 - gnss receiver operation overview. In Benedetto, J. J., editor, *A Software Defined GPS and Galileo Receiver*, pages 75–80. Academic Press.

[Braasch, 2017] Braasch, M. (2017). Multipath. In *Springer Handbook of Global Navigation Satellite Systems*, chapter 15, pages 448–450. Springer.

[Braasch and Dierendonck, 1999] Braasch, M. and Dierendonck, A. (1999). Gps receiver architectures and measurements. *Proceedings of the IEEE*, 87:48 – 64.

[Brown, 2012] Brown, R. G. (2012). *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises, 4th Edition*. American Institute of Aeronautics and Astronautics.

[Buda et al., 2017] Buda, M., Maki, A., and Mazurowski, M. A. (2017). A systematic study of the class imbalance problem in convolutional neural networks. *CoRR*, abs/1710.05381.

[Bétaille et al., 2013] Bétaille, D., Peyret, F., Ortiz, M., Miquel, S., and Fontenay, L. (2013). A new modelling based on urban trenches to improve gnss positioning quality of service in cities. *Intelligent Transportation Systems Magazine, IEEE*, 5:59–70.

[Carneiro et al., 2017] Carneiro, G., Nascimento, J., and Bradley, A. P. (2017). Chapter 14 - deep learning models for classifying mammogram. In Zhou, S. K., Greenspan, H., and Shen, D., editors, *Deep Learning for Medical Image Analysis*, pages 321–339. Academic Press.

[Chai et al., 2022] Chai, D., Yipeng, N., Wang, S., Sang, W., Xing, J., and Bi, J. (2022). A robust algorithm for multi-gnss precise positioning and performance analysis in urban environments. *Remote Sensing*, 14:5155.

[Chollet, 2015] Chollet, F. (2015). keras. https://github.com/fchollet/keras.

[Chu and Yang, 2018] Chu, F.-Y. and Yang, M. (2018). Beidou system (bds) triple-frequency ambiguity resolution without code measurements. *Remote Sensing*, 10:675.

[Circiu et al., 2017] Circiu, M.-S., Meurer, M., Felux, M., Gerbeth, D., Thölert, S., Vergara, M., Enneking, C., Sgammini, M., and Antreich, F. (2017). Evaluation of gps l5 and galileo e1 and e5a performance for future multifrequency and multiconstellation gbas. *Navigation, Journal of the Institute of Navigation*, 64.

[Dalianis, 2018] Dalianis, H. (2018). *Evaluation Metrics and Evaluation*, pages 45–53. Springer International Publishing, Cham.

[Datta-Barua et al., 2008] Datta-Barua, S., Walter, T., Blanch, J., and Enge, P. (2008). Bounding higher-order ionosphere errors for the dual-frequency gps user. *Radio Science*, 43(5).

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. volume 06.

[de Menezes et al., 2021] de Menezes, D., Prata, D., Secchi, A., and Pinto, J. (2021). A review on robust m-estimators for regression analysis. *Computers & Chemical Engineering*, 147:107254.

[Deo and El-Mowafy, 2016] Deo, M. and El-Mowafy, A. (2016). Triple-frequency gnss models for ppp with float ambiguity estimation: performance comparison using gps. *Survey Review*, pages 1–13.

[Ding et al., 2022] Ding, Y., Chauchat, P., Pages, G., and Asseman, P. (2022). Learning-enhanced adaptive robust gnss navigation in challenging environments. *IEEE Robotics and Automation Letters*, 7(4):9905–9912.

[Ding et al., 2023] Ding, Y., Feriol, F., Watanabe, Y., Asseman, P., Pages, G., and Vivet, D. (2023). Adaptive robust-statistics gnss navigation based on environmental context detection. pages 138–152.

[Dong et al., 2020] Dong, Y., Wang, D., Zhang, L., Li, Q., and Wu, J. (2020). Tightly coupled gnss/ins integration with robust sequential kalman filter for accurate vehicular navigation. *Sensors*, 20(2).

[Du and Gao, 2012] Du, S. and Gao, Y. (2012). Inertial aided cycle slip detection and identification for integrated ppp gps and ins. *Sensors (Basel, Switzerland)*, 12:14344–62.

[Duong, 2019] Duong, V. (2019). An optimal linear combination model to accelerate ppp convergence using multi-frequency multi-gnss measurements. *GPS Solutions*, 23.

[El-naggar, 2011] El-naggar, A. M. (2011). Enhancing the accuracy of gps point positioning by converting the single frequency data to dual frequency data. *Alexandria Engineering Journal*, 50(3):237–243.

[Falco et al., 2017] Falco, G., Pini, M., and Marucco, G. (2017). Loose and tight gnss/ins integrations: Comparison of performance assessed in real urban scenarios. *Sensors*, 2017:27.

[Fan Liu, 2006] Fan Liu, Mats Brenner, C. Y. T. (2006). Signal deformation monitoring scheme implemented in a prototype local area augmentation system ground installation. *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, 19.

[Foucras et al., 2014] Foucras, M., Ekambi, B., Ngayap, U., Li, J., Julien, O., and Macabiau, C. (2014). Performance study of fll schemes for a successful acquisition-to-tracking transition. pages 529–540.

[Foucras et al., 2016] Foucras, M., Julien, O., Macabiau, C., Ekambi, B., and Bacard, F. (2016). Probability of detection for gnss signals with sign transitions. *IEEE Transactions on Aerospace and Electronic Systems*, 52:1296–1308.

[Gaglione, 2015] Gaglione, S. (2015). How does a gnss receiver estimate velocity? *Inside GNSS*, pages 38–41.

[Gaglione et al., 2017] Gaglione, S., Innac, A., Carbone, S., Troisi, S., and Angrisano, A. (2017). Robust estimation methods applied to gps in harsh environments.

[Gao et al., 2017] Gao, Z., Zhang, H., Ge, M., Niu, X., Shen, W., Wickert, J., and Schuh, H. (2017). Tightly coupled integration of multi-gnss ppp and mems inertial measurement unit data. *GPS Solutions*, 21:377–391.

[Garcia Crespillo et al., 2020] Garcia Crespillo, O., Andreetti, A., and Grosch, A. (2020). Design and evaluation of robust m-estimators for gnss positioning in urban environments. pages 750–762.

[Garcia Crespillo et al., 2017] Garcia Crespillo, O., Medina, D., Grosch, A., Skaloud, J., and Meurer, M. (2017). Robust tightly coupled gnss/ins estimation for navigation in challenging scenarios.

[Ghadrdan et al., 2012] Ghadrdan, M., Grimholt, C., Skogestad, S., and Halvorsen, I. J. (2012). Estimation of primary variables from combination of secondary measurements: Comparison of alternative methods for monitoring and control. In Karimi, I. A. and Srinivasan, R., editors, *11th International Symposium on Process Systems Engineering*, volume 31 of *Computer Aided Chemical Engineering*, pages 925–929. Elsevier.

[Groves, 2008] Groves, P. (2008). *Principles of GNSS, Inertial, Multisensor Integrated Navigation Systems*. Artech House.

[Groves et al., 2013] Groves, P., Jiang, Z., Rudi, M., and Strode, P. (2013). A portfolio approach to nlos and multipath mitigation in dense urban areas. volume 4.

[Guillard et al., 2022] Guillard, A., Thevenon, P., and Milner, C. (2022). Using tdcp measurements in a low-cost ppp-imu hybridized filter for real-time applications. In *ION GNSS + 2022, International Technical Meeting*, Denver, Colorado, United States. Institute of Navigation, ION.

[Guillard et al., 2023] Guillard, A., Thevenon, P., and Milner, C. (2023). Using convolutional neural networks to detect gnss multipath. *Frontiers in Robotics and AI*, 10.

[Hsu, 2017] Hsu, L.-T. (2017). Gnss multipath detection using a machine learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6.

[Hubel DH, 1968] Hubel DH, W. T. (1968). Receptive fields and functional architecture of monkey striate cortex. pages 215–243.

[Hui, 2020] Hui, R. (2020). Chapter 5 - optical amplifiers. In Hui, R., editor, *Introduction to Fiber-Optic Communications*, pages 155–207. Academic Press.

[IFEN, 2023] IFEN (2023). Gnss r&d software receiver portfolio. https://www.ifen.com/products/sx3-gnss-software-receiver/. Rev. 1.

[Irsigler et al., 2005] Irsigler, M., Avila-Rodriguez, J.-A., and Hein, G. (2005). Criteria for gnss multipath performance assessment.

[Jiang C., 2021] Jiang C., Xu B., H. L. (2021). Probabilistic approach to detect and correct gnss nlos signals using an augmented state vector in the extended kalman filter. *GPS Solut 25*, 72.

[Jong-Hoon Won, 2017] Jong-Hoon Won, T. P. (2017). Signal processing. In *Springer Handbook of Global Navigation Satellite Systems*, chapter 14, pages 428–433. Springer.

[Jovancevic et al., 2004] Jovancevic, A., Brown, A., Ganguly, S., Noronha, J., and Sirpatil, B. (2004). Ultra tight coupling implementation using real time software receiver.

[Julien, 2005] Julien, O. (2005). Impact of future gnss signals on carrier phase tracking. In *ENC GNSS05*, Munich Germany.

[Julien, 2006] Julien, O. (2006). Design of galileo l1f receiver tracking loops. *PhD Thesis, University of Calgary, Canada*.

[Kafadar, 1983] Kafadar, K. (1983). The efficiency of the biweight as a robust estimator of location. *Journal of research of the National Bureau of Standards*, 88 2:105–116.

[Kai et al., 2021] Kai, C., Chang, G., Chao, C., and Zhu, T. (2021). Improved tdcp-gnss/ins integration scheme considering small cycle slip for low-cost land vehicular applications. *Measurement Science and Technology*, 32.

[Kim et al., 2020] Kim, J., Park, M., Bae, Y., Kim, O.-J., Kim, D., Kim, B., and Kee, C. (2020). A low-cost, high-precision vehicle navigation system for deep urban multipath environment using tdcp measurements. *Sensors*, 20(11).

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Red Hook, NY, USA. Curran Associates Inc.

[Lampert, 2009] Lampert, C. H. (2009). Kernel methods in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 4(3):193–285.

[Langley, 1999] Langley, R. B. (1999). Dilution of precision.

[Lau, 2021] Lau, L. (2021). Chapter 4 - gnss multipath errors and mitigation techniques. In p. Petropoulos, G. and Srivastava, P. K., editors, *GPS and GNSS Technology in Geosciences*, pages 77–98. Elsevier.

[Leclère et al., 2014] Leclère, J., Botteron, C., and Farine, P.-A. (2014). Acquisition of modern gnss signals using a modified parallel code-phase search architecture. *Signal Processing*, 95:177–191.

[Legrand et al., 2000] Legrand, F., Macabiau, C., Issler, J.-L., Lestarquit, L., and Mehlen, C. (2000). Improvement of pseudorange measurements accuracy by using fast adaptive bandwidth lock loops.

[Li, 2018] Li, B. (2018). Review of triple-frequency gnss: ambiguity resolution, benefits and challenges. *The Journal of Global Positioning Systems*, 16.

[Li, 2019] Li, P. (2019). Kalman-filter-based undifferenced cycle slip estimation in real-time precise point positioning. *GPS Solutions*, 23.

[Li, 2020] Li, P. (2020). Gps, galileo, and beidou precise point positioning with triple-frequency ambiguity resolution. *GPS Solutions*, 24.

[Liu, 2018] Liu, Y. (2018). Feature extraction and image recognition with convolutional neural networks. *Journal of Physics: Conference Series*, 1087:062032.

[Macabiau et al., 2012] Macabiau, C., Deambrogio, L., Barreau, V., Vigneau, W., Valette, J.-J., Artaud, G., Thevenon, P., and Ries, L. (2012). Kalman filter based robust GNSS signal tracking algorithm in presence of ionospheric scintillations. In *ION GNSS 2012, 25th International Technical Meeting of The Satellite Division of the Institute of Navigation*, pages pp 3420–3434, Nashville, United States. http://www.ion.org/publications/abstract.cfm?articleID=10514.

[Marais et al., 2014] Marais, J., Meurie, C., Attia, D., Ruichek, Y., and Flancquart, A. (2014). Toward accurate localization in guided transport: Combining gnss data and imaging information. *Transportation Research Part C: Emerging Technologies*, 43:188–197. Special Issue with Selected Papers from Transport Research Arena.

[Marco Rao, 2012] Marco Rao, G. F. (2012). How can pseudorange measurements be generated from code tracking? *Inside GNSS*, pages 26–33.

[Masters and Luschi, 2018] Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks.

[Matera et al., 2018] Matera, E. R., Garcia Peña, A. J., Julien, O., and Ekambi, B. (2018). Characterization Of Pseudo Range Multipath Errors In An Urban Environment. In *ITSNT 2018, International Technical Symposium on Navigation and Timing*, Toulouse, France.

[Matera et al., 2019] Matera, E. R., Garcia Peña, A. J., Julien, O., Milner, C., and Ekambi, B. (2019). Characterization of Line-of-Sight and Non-Line-of-Sight Pseudorange Multipath Errors in Urban Environment for GPS and Galileo. In *ITM 2019, International Technical Meeting of The Institute of Navigation*, ITM 2019, International Technical Meeting of The Institute of Navigation, pages pp. 177–196., Reston, United States. ION.

[Medina et al., 2019a] Medina, D., Li, H., Vilà-Valls, J., and Closas, P. (2019a). On robust statistics for gnss single point positioning.

[Medina et al., 2019b] Medina, D., Li, H., Vilà-Valls, J., and Closas, P. (2019b). Robust statistics for gnss positioning under harsh conditions: A useful tool? *Sensors*, 19(24).

[Munin et al., 2020] Munin, E., Blais, A., and Couellan, N. (2020). Gnss multipath detection using embedded deep cnn on intel® neural compute stick. pages 2018–2029.

[Muthuraman et al., 2008] Muthuraman, K., Klukas, R., and Lachapelle, G. (2008). Performance evaluation of l2c data/pilot combined carrier tracking. 1.

[Neyman and Pearson, 1933] Neyman, J. and Pearson, E. S. (1933). On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337.

[Ng, 2004] Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA. Association for Computing Machinery.

[Pagot et al., 2017] Pagot, J.-B., Julien, O., and Selmi, I. (2017). Additional sqm metrics for generic tm detection. In *2017 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6.

[Pagot et al., 2015] Pagot, J.-B., Thevenon, P., Julien, O., Gregoire, Y., Amarillo-Fernandez, F., and Maillard, D. (2015). Estimation of gnss signals' nominal distortions from correlation and chip domain.

[Park and Kee, 2010] Park, B. and Kee, C. (2010). The compact network rtk method: An effective solution to reduce gnss temporal and spatial decorrelation error. *Journal of Navigation*, 63(2):343–362.

[Paziewski et al., 2019] Paziewski, J., Sieradzki, R., and Baryla, R. (2019). Signal characterization and assessment of code gnss positioning with low-power consumption smartphones. *GPS Solut.*, 23(4):1–12.

[Peyret et al., 2014] Peyret, F., Bétaille, D., Carolina, P., Toledo-Moreo, R., Gómez-Skarmeta, A. F., and Ortiz, M. (2014). Gnss autonomous localization: Nlos satellite detection based on 3-d maps. *IEEE Robotics & Automation Magazine*, 21(1):57–63.

[Powers, 2020] Powers, D. M. W. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR*, abs/2010.16061.

[Prasad, 2019] Prasad, S. (2019). Double block zero padding acquisition algorithm for gps software receiver. *Journal of Automation, Mobile Robotics and Intelligent Systems*, 12:58–63.

[Prochniewicz and Grzymala, 2021] Prochniewicz, D. and Grzymala, M. (2021). Analysis of the impact of multipath on galileo system measurements. *Remote Sensing*, 13(12).

[Quan et al., 2018] Quan, Y., Lau, L., Roberts, G., Meng, X., and Zhang, C. (2018). Convolutional neural network based multipath detection method for static and kinematic gps high precision positioning. *Remote Sensing*, 10:2052.

[Rahman et al., 2015] Rahman, G., Mubeen, S., and Rehman, A. (2015). Generalization of chi-square distribution. *Journal of Statistics Applications and Probability*, 4:119–126.

[Rakipi et al., 2015] Rakipi, A., Kamo, B., Cakaj, S., Kolici, V., Lala, A., and Shinko, I. (2015). Integrity monitoring in navigation systems: Fault detection and exclusion raim algorithm implementation. *Journal of Computer and Communications*, 03:25–33.

[Rousseeuw and Yohai, 1984] Rousseeuw, P. and Yohai, V. (1984). Robust regression by means of s-estimators. *Springer Lecture Notes in Statistics*, 26:256–272.

[Rousseeuw and Croux, 1993] Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283.

[Ruder, 2017] Ruder, S. (2017). An overview of gradient descent optimization algorithms.

[Ruppert and Wand, 1994] Ruppert, D. and Wand, M. P. (1994). Multivariate Locally Weighted Least Squares Regression. *The Annals of Statistics*, 22(3):1346 – 1370.

[Sadrieh et al., 2012] Sadrieh, S. N., Broumandan, A., and Lachapelle, G. (2012). Doppler characterization of a mobile gnss receiver in multipath fading channels. *Journal of Navigation*, 65:477.

[Salós et al., 2010] Salós, D., Macabiau, C., Martineau, A., Bonhoure, B., and Kubrak, D. (2010). Nominal GNSS pseudorange measurement model for vehicular urban applications. In *IEEE/ION PLANS 2010, Position Location and Navigation Symposium*, pages pp 806–815, Indian Wells, United States.

[Sanromà Sánchez et al., 2016] Sanromà Sánchez, J., Gerhmann, A., Thevenon, P., Brocard, P., Ben Afia, A., and Julien, O. (2016). Use of a fisheye camera for gnss nlos exclusion and characterization in urban environments. In *ION ITM 2016, International Technical Meeting*, Monterey, United States. Institute of Navigation, ION.

[Song et al., 2020] Song, J., Milner, C., and Selmi, I. (2020). Signal deformation fault monitors for dual-frequency gbas. *NAVIGATION: Journal of the Institute of Navigation*, 67(2):379–396.

[Soon et al., 2008] Soon, B., Scheding, S., Lee, H., Lee, H. K., and Durrant-Whyte, H. (2008). An approach to aid ins using time-differenced gps carrier phase (tdcp) measurements. *GPS Solutions*, 12:261–271.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[Stefanski and Boos, 2002] Stefanski, L. A. and Boos, D. D. (2002). The calculus of m-estimation. *The American Statistician*, 56(1):29–38.

[Suzuki and Amano, 2021] Suzuki, T. and Amano, Y. (2021). Nlos multipath classification of gnss signal correlation output using machine learning. *Sensors*, 21:2503.

[Suzuki et al., 2020] Suzuki, T., Matsuo, K., and Amano, Y. (2020). Rotating gnss antennas: Simultaneous los and nlos multipath mitigation. *GPS Solutions*, 24:86.

[Tanil, 2022] Tanil, C. (2022). Kalman filter partial innovation sequence monitor. pages 1263–1272.

[Teunissen, 1996] Teunissen, P. J. G. (1996). *GPS carrier phase ambiguity fixing concepts*, pages 263–335. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Toka and Cetin, 2010] Toka, O. and Cetin, M. (2010). The comparing of s-estimator and m-estimators in linear regression. *Gazi University Journal of Science*, 24:747–752.

[Tranquilla et al., 1994] Tranquilla, J., Carr, J., and Al-Rizzo, H. (1994). Analysis of a choke ring groundplane for multipath control in global positioning system (gps) applications. *IEEE Transactions on Antennas and Propagation*, 42(7):905–911.

[Van Dierendonck et al., 1992] Van Dierendonck, A. J., Fenton, P., and Ford, T. (1992). Theory and performance of narrow correlator spacing in a gps receiver. *NAVIGATION*, 39(3):265–283.

[Van Neem D.J.R, 1991] Van Neem D.J.R, C. A. (1991). New fast gps code-acquisition technique using fft. pages 158–160.

[Vana et al., 2019] Vana, S., Naciri, N., and Bisnath, S. (2019). Low-cost, dual-frequency ppp gnss and mems-imu integration performance in obstructed environments. pages 3005–3018.

[Vergara et al., 2009] Vergara, M., Antreich, F., and Meurer, M. (2009). Effect of multipath on code-tracking error jitter of a delay locked loop. Proceedings of Fourth European Workshop on GNSS signals and signal processing, GNSS SIGNALS 2009, Oberpfaffenhofen, Germany.

[Vezinet, 2014] Vezinet, J. (2014). *Study of Future On-board GNSS/INS Hybridization Architectures*. PhD thesis. Thèse de doctorat dirigée par Macabiau, Christophe et Escher, Anne-Christine Signal, Image, Acoustique et Optimisation Toulouse, INPT 2014.

[Wang et al., 2022] Wang, J., Chen, X., Shi, C., and Liu, J. (2022). An improved robust estimation method for gnss/sins under gnss-challenged environment. In *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 79–83.

[Wang et al., 2018] Wang, N., Li, Z., Li, M., Yuan, Y., and Huo, X. (2018). Gps, bds and galileo ionospheric correction models: An evaluation in range delay and position domain. *Journal of Atmospheric and Solar-Terrestrial Physics*, 170:83–91.

[Wei et al., 2021] Wei, Q., Hailu, C., Qin, Z., Yuan, G., Qingliang, W., and Ming, H. (2021). A robust estimation algorithm for the increasing breakdown point based on quasi-accurate detection and its application to parameter estimation of the gnss crustal deformation model. *Journal of Geodesy*, 95.

[Welch and Bishop, 1997] Welch, G. and Bishop, G. (1997). Scaat: Incremental tracking with incomplete information. pages 333–344.

[Won et al., 2012] Won, J.-H., Pany, T., and Eissfeller, B. (2012). Characteristics of kalman filters for gnss signal tracking loop. *IEEE Transactions on Aerospace Electronic Systems*, 48:3671–3681.

[Woodman, 2007] Woodman, O. J. (2007). An introduction to inertial navigation.

[Xu et al., 2020] Xu, H., Angrisano, A., Gaglione, S., and Hsu, L.-T. (2020). Machine learning based los/nlos classifier and robust estimator for gnss shadow matching. *Satellite Navigation*.

[Ying, 2019] Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022.

[Zafar et al., 2022] Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A. K., and Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, 12(17).

[Zhang and Sabuncu, 2018] Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836.

[Zhao et al., 2013] Zhao, H., Mayzus, R., Sun, S., Samimi, M., Schulz, J., Azar, Y., Wang, K., Wong, G., Gutierrez, F., and Rappaport, T. (2013). 28 ghz millimeter wave cellular communication measurements for reflection and penetration loss in and around buildings in new york city. In *2013 IEEE International Conference on Communications, ICC 2013*, IEEE International Conference on Communications, pages 5163–5167. Institute of Electrical and Electronics Engineers Inc. 2013 IEEE International Conference on Communications, ICC 2013 ; Conference date: 09-06-2013 Through 13-06-2013.

[Zhu, 1994] Zhu, J. (1994). Conversion of earth-centered earth-fixed coordinates to geodetic coordinates. *IEEE Transactions on Aerospace and Electronic Systems*, 30(3):957–961.

[Zou and Hastie, 2005] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320.

[Zoubir et al., 2018] Zoubir, A., Koivunen, V., Ollila, E., and Muma, M. (2018). *Robust Statistics for Signal Processing*. Cambridge University Press, United Kingdom.